

## MASTER

### Feasibility study on the reliability factor "system complexity" a case study at the Cardio Vascular X-ray scanner at Philips Healthcare

Albers, S.

*Award date:*  
2008

[Link to publication](#)

#### **Disclaimer**

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

#### **General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

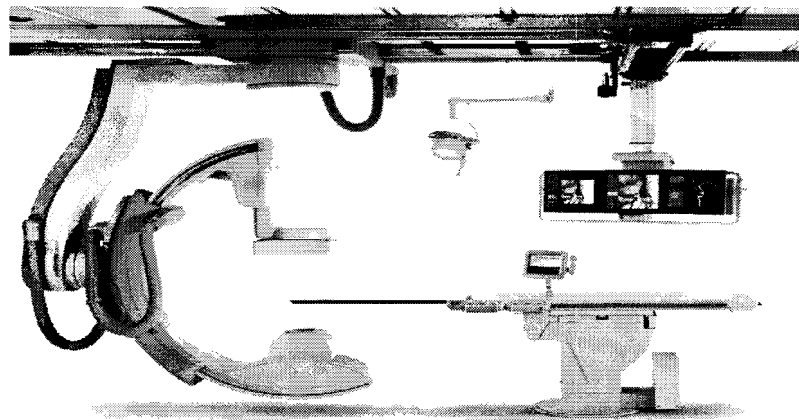
- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Eindhoven, July 2008

**Feasibility study on the reliability  
factor “system complexity”:**

**A case study at the Cardio Vascular X-ray  
scanner at Philips Healthcare**

By S. Albers



Author:

S. Albers  
0492372

College of Education:

Eindhoven University of Technology  
MSc Operations Management and Logistics

Supervisor Philips Healthcare:

Dr. G. Stollman, Reliability manager, CV development

Supervisors TU/e, QRE:

Dr. Ir. P.J.M. Sonnemans, TU/e, Industrial Design, BPD  
Dr. J.A. Keizer, TU/e, Technology Management, OSM

# Preface

This master thesis report is the last trial of my study Industrial Engineering and Management at Eindhoven University of Technology. The master thesis project is performed for the Cardio/Vascular development department of Philips Healthcare in Best and for the Ph.D. project led by Kostas Kevrekidis at the Quality & Reliability Engineering department of Eindhoven University of Technology.

The start of the project wasn't that fluent, it was hard to grasp the vague concept of "system complexity" and express it in concrete measurable terms. Besides, the availability of all the required data wasn't that straightforward and simple to collect as was assumed beforehand. Here I got great support from my direct colleagues: Kostas Kevrekidis, Noel Aben, Dirk in het Veldt, and Kenny van Uden, who faced similar problems within their promotion and graduation projects. Their companionship and sparring discussions provided a nice atmosphere to work in.

Besides, I would like to thank the employees of the Cardio/Vascular development department in Philips Healthcare in Best. My special thanks go out to: Riny van Asten, Klaas Tanis, Krzysztof Oborzynski, and Mark Loos, who provided me with great support to get the project on track in the beginning.

Further, I would like to thank my supervisor of Philips Healthcare, Guillaume Stollman, who made it possible to carry out this master thesis project within the Cardio/Vascular development department. Moreover, he not only provided me guidance and advice regarding my project, but also put trust in this rather theoretical and explorative study.

Next to the colleagues within Philips Healthcare, I would like to thank my direct supervisors of Eindhoven University of Technology: Peter Sonnemans and Jimme Keizer, for their support and constructive criticism on this thesis. In special I would like to thank my daily supervisor Kostas Kevrekidis, who kept me focused and critical all along the project. Moreover, our cooperation went fluently, which made me enjoy the project even more.

As a last remark I would like to notify to all readers that some paragraphs and tables are excluded from the appendices due to confidential information.

Stijn Albers

Eindhoven, July 2008

# Management Summary

## Research area

There are several reliability factors that are still relatively unknown or their impact on reliability is not known. Some examples are: supplier quality differences, environmental stress factors, system complexity and system age. (o'Connor, 1991) Hence, this master thesis will focus on one of these relatively unknown reliability affecting factors; system complexity.

Extended knowledge on system complexity should enable a classification of the systems according to this factor and reduce the variance in the reliability performance of each classified group. It is expected that this classification according to complexity enables better understanding of the problems/issues that cause the high variance in reliability performance.

This relatively “unknown” reliability affecting factor is selected based on findings by the Ph.D. research by K. Kevrekidis who applied the (social sciences) “Grounded theory” by Glaser & Strauss (1967) to select relevant reliability influencing factors. This master thesis assumes that these reliability factors found with this theory are truly relevant factors.

### Problem definition:

*More quantitative insight in the relationship between system complexity and the system’s reliability performance level is required in order to determine if system complexity can actually explain the variance within the reliability performance of professional systems.*

## Research question

The aim of this Master thesis is to investigate the relation between the factor “system complexity” and the systems reliability performance. Therefore, this research should provide a methodology how to measure/quantify the reliability factor “system complexity”. Finally, this research should establish if a system classification according to “system complexity” can be made and if it actually reduces the variance of reliability performance within subgroups among the Allura Xper Cardio Vascular (CV) X-ray scanners at Philips Healthcare (Philips Medical Systems).

This led to the following central research question for this master thesis:

*What is the impact of “system complexity” on the reliability performance of the professional repairable systems under study and how can this gained knowledge be assimilated into decision rules for product improvement?*

## Assignment

This master thesis will come up with the following deliverables:

- This research should define the way in which the reliability performance of the professional system should be measured
- This master thesis provides a clear definition of “system complexity”. This definition is expressed in measurable indicators which are representative for the actual system complexity.
- The expected relation between system complexity and reliability has been defined and are tested in practice by a case study on the CV X-ray scanner of Philips Medical Systems (PMS).
- An attempt to create a classification model for the professional repairable systems according to the reliability factor “system complexity” has been made.
- This master thesis investigated if the variance of the reliability performance level within a classified group is smaller than the variance of the reliability performance for the group without a classification according to complexity.
- Higher aim of this research was to provide the Ph.D. project with relevant and available data on the reliability factor “system complexity”.

## Theoretical and practical relevance

The main contribution to the scientific research is the overview of the complexity and reliability metrics that are required in order to investigate the relation between these two variables. Moreover, a data set and its limitations are provided, so other researchers can use and interpret this data set in a correct way as well. This is important as this master thesis is a contribution to the Ph.D. research of K. Kevrekidis at the Quality & Reliability Engineering department at Eindhoven University of Technology. Furthermore, this research combines theoretical definitions and approaches of system complexity and tries to apply them in practice. It shows that many theoretically important complexity concepts (or software code metrics) are not applicable in practice or are not relevant for the relation with system reliability for this CV X-ray scanner at Philips Medical Systems.

## Research design

This research applies the Kumar's (1999) research methodology. This methodology is specifically suitable for a theoretical research, which is the case in this Master thesis project.

The methodology divides a research in several research phases and provides a format of the content and activities for each phase. These phases are: formulation of the research problem, designing the research concept, construction of the data collection instrument, selection of the sample, create the research proposal, collect data and finally process the data.

## Results

### Software complexity versus reliability:

#### **Software reliability and complexity definitions & availability and quality of required data**

The following available reliability performance indicators were selected: Problem reports (PR) during testing, Field programming errors (FPE) derived from the system log-files, Software failures (SWF), derived from the failure classification by the FMEA developed by the PhD researcher K. Kevrekidis. Furthermore, over 40 separate complexity metrics were derived and transformed into two common complexity factors "size" and "structure", for which composite factor scores were defined. For 17 observations (SW units) all complexity metrics and reliability metrics PR and FPE were available and for 19 observations all complexity metrics and reliability metric SWF are available.

#### **Established relations between complexity factors and reliability metrics**

The established relations between the developed complexity factors and the reliability metrics FPE and SWF are expressed in the following three ways:

##### *- Correlation between reliability and complexity metrics:*

Based on the Kendall's tau correlations test between the two complexity factor scores and the reliability metrics FPE, MTBF, Entropy, it can be stated that there is a significant correlation between the complexity factor "size" and the reliability metric FPE. The same conclusion can be derived for the complexity factor size and the reliability metric SWF (based on the Pearson correlation test).

##### *- Explaining the variance within FPE and SWF based on the defined complexity factors:*

Several regression models are developed in chapter 6 and can be used in order to explain the variance within the reliability metrics FPE and SWF rather well. The variance in the FPE and SWF scores among the observed SW units could be explained best by the complexity factor "size".

##### *- Prediction of the FPE and SWF based on the SW unit's complexity factor scores:*

The regression models based on one or two complexity factors "size" and "structure" are over-rated, due to the small number of observations within the training group, and can therefore not be used in order to predict reliability metrics FPE or SWF for the SW units in the control group accurately.

#### **Classification of SW units based on the complexity factors "size" and "structure"**

The SW units were classified into groups with similar complexity scores. This resulted in three complexity classes: low, middle, and high. Subsequently some analyses for categorical data were performed and the conclusions follow in the next paragraph.

### Hardware complexity versus reliability:

#### **Hardware reliability and complexity definitions & availability and quality of required data**

There were no real hardware reliability metrics available for the Allura Xper systems within PMS. Only the number of component replacements (during “corrective” maintenance) is available as an indicator for the hardware reliability performance. Although a component replacement can also be caused by a software failure, it is assumed that the replacements are all caused by hardware failures. This assumption reduces the trustworthiness of the conclusions on the relationship between hardware reliability and complexity. Nevertheless, the results still may present an indication of how the hardware complexity and reliability are interrelated for the PMS Allura Xper systems.

#### **Established relations between hardware complexity and reliability**

No significant correlations have been found among the HW complexity attributes or classes and HW failures. Therefore the complexity metrics cannot be used in order to explain the variance within the HW failures, let alone predict the number of HW failures.

#### **Categorizing the HW building blocks based on its complexity scores**

The HW building blocks were categorized into two groups: low complexity and high complexity scores. The average number of HW failures (and its variance) was larger for the group which included the “low complex” building blocks. This result contradicts with the expectation that more complex HW building blocks will on average have a higher number of HW failures. The only logical explanation for this remarkable result is that the reliability metrics or the complexity scores are not relevant or representative for the actual situation.

## **Conclusions**

#### **Reflection on the research question**

This master thesis came up with some conclusions on this relationship for the software units as well as the hardware building blocks based on the Allura Xper systems at PMS. Although, the relationship between hardware complexity and reliability has been investigated less thoroughly, this thesis came up with an overview of the required and available data at PMS for the reliability factor “system complexity”.

All research questions are answered except for the last sub question: “*How can extended knowledge about system complexity and its relation with reliability performance assist the decision process in product development*”. Reason for this, was the lack of time to investigate the decision process in product development. The other research questions and corresponding findings are as follows:

#### *1. How should the reliability performance of the professional system be defined or measured?*

This research defined several software related reliability indicators that could be used in order to express the reliability performance of the Allura Xper system. These indicators were: test problem reports (PR), field programming errors (FPE), mean time between programming errors (MTBF), entropy and classified software failures (SWF). For hardware only one reliability indicator was available, which was the number of component replacements from the panel analyses.

#### *2. How should “system complexity” be defined in such a way that there will arise no misconceptions in interpreting these factors?*

First the required system complexity metrics were defined for both hardware and software. Subsequently the available complexity metrics were defined / measured. For software complexity, around the 40 code metrics were collected, these metrics were transformed into two complexity factors “size” and “structure”. Furthermore, the metrics from different sources were compared in order to validate if the complexity metrics scores were representing the software complexity consistently. There were 8 hardware complexity attributes defined and quantified by a small survey among system architects. These eight complexity attributes were transformed into an aggregated complexity measure.

3. *How should the relationship between “system complexity” and the reliability performance of the system be established?*

Software complexity versus reliability: The relationship between system complexity and reliability has been researched in several ways. First of all, several metrics are used to express system complexity and the reliability performance and they are independently investigated. Secondly, there are several types of tests performed: correlation tests among the metrics, regression analysis and chi-square tests. These analyses were performed independently for the system’s hardware and software.

It shows that the complexity factor size does correlate with the number of FPE and SWF, and factor “structure” does not. The correlation can be explained by the assumption that larger software code leads to more FPEs or SWFs due to higher chance of mistakes by the programmer. However, the absence of correlation between the factor structure and FPE and SWF can not be explained with certainty, but probably the “structure” related metrics show less discriminative scores among the software units and therefore cannot explain the variance in FPE or SWF.

Furthermore, the developed regression models are based on only a small number of observations. Therefore there is a serious risk that the regression models are over-fitting the data and cannot be used in order to predict the reliability performance in FPE or SWF accurately.

Hardware complexity versus reliability: There were no significant correlations found between the hardware complexity attributes and the reliability indicator “component replacements”. Therefore no conclusions on this relationship are found.

4. *How to create a product classification process based on system complexity?*

The software units and the hardware building blocks were classified according to its complexity scores as well as its reliability performance level. Based on these classifications it can be concluded that the means and variance of the reliability performance (expressed in SWF) increases when the group of software units are classified as “high” complex. For the hardware building blocks, the variance within reliability performance could not be explained by the classification of the building blocks according to complexity.

## **Recommendations**

### **Software complexity versus software reliability**

The following topics and directions are recommended for further research within investigation of the relationship between SW complexity and SW reliability:

- Find more discriminative complexity code metrics for “structure” related complexity.
- Repeat the regression analysis for a larger training group and investigate if the accuracy of the reliability predictions based on the complexity scores improves.
- Add the (in literature) recommended complexity metric “fan in / fan out” to the code complexity model and investigate if more complexity factors arise or if the ability to explain the variance in the reliability scores among SW units by the complexity factors improves.
- Investigate which underlying reasons cause the increased variance in software errors and failures for systems with high scores on the complexity factor “size”.
- As the complexity factor “size” seems to be related to reliability, the number of SWF could be normalized by for example the Lines of Code (LOC) in order to provide more representative and fair measures of the actual reliability performance of the SW units.
- Perform more case studies on other complex professional repairable systems that include (embedded) software, and discover if there are similar findings and conclusions. Subsequently, one can investigate if these findings can be combined into a general applicable regression model that fits the SW of complex professional repairable systems in general.
- Find out how the established relations between complexity and reliability can provide more focus/insight for the development team and support them in making product improvement decisions.

### **Hardware complexity versus hardware reliability**

The following topics and directions are recommended for further research within investigation of the relationship between HW complexity and HW reliability:

- This master thesis investigated this relationship quickly and not very thoroughly, therefore more extended investigation of this relationship is recommended. Especially because not much quantitative research on this relationship has been carried out in literature yet.
- Within PMS, effort should be spent on getting insight in the actual failure causes that have led to the component replacements (during “corrective” maintenance). The required reliability measures should be defined upfront the process of data collection and filtering, in order to make sure that all required reliability data is available.

### **General recommendations for further research**

Here follow some general recommendations for further research on the research topics system complexity and system reliability:

- A challenging research would be to quantify the interactions between hardware and software reliability, and see how system complexity influences this interaction.
- Carry out more quantitative case studies on several types of complex professional repairable systems in order to see if the conclusions and regression models from this research can be generalized.
- Investigate other factors (for example the in this master thesis called extraneous variables: “environment” and “user profile”) that could explain the variance within the reliability performance.



# Table of contents

<b>Abstract</b> .....	<b>I</b>
<b>Preface</b> .....	<b>II</b>
<b>Management summary</b> .....	<b>III</b>
<b>1. Company profile</b> .....	<b>1</b>
1.1 Introduction .....	1
1.2 Royal Philips Electronics .....	1
1.3 Philips Medical Systems .....	1
1.4 The Allura Xper Cardio/Vascular X-ray scanner.....	2
<b>2. Research outline</b> .....	<b>5</b>
2.1 Formulation of the research area .....	5
2.2 Problem situation.....	6
2.3 Research objective.....	6
2.4 Research questions .....	8
2.5 Research approach.....	9
2.6 Research boundaries.....	9
<b>3. Designing the research concept</b> .....	<b>11</b>
3.1 Defining the expected causal relationship.....	11
3.2 Decomposition of “system complexity & reliability” .....	12
3.2.1 System complexity .....	13
3.2.2 Decomposition into software and hardware .....	14
3.2.3 System reliability and reliability metrics.....	14
3.3 Sample selection & Data collection .....	15
3.3.1 Selecting a study design and sample .....	15
3.3.2 Selecting a method of data collection.....	15
<b>4. Software reliability</b> .....	<b>16</b>
4.1 Separate software into subsystems and units.....	16
4.2 How to measure software reliability?.....	16
4.3 Evaluation and selection of software reliability data .....	17
4.4 Data sources for reliability indicators.....	18
4.4.1 Data source: system Log-files .....	18
4.4.2 Data source: Problem reports .....	19
4.5 Findings and conclusions on available reliability data.....	20
4.5.1 Source: System log-files.....	20
4.5.2 Source: Problem reports .....	23
4.5.3 Correlation among reliability metrics.....	23

<b>5. Software complexity</b> .....	<b>25</b>
5.1 How to measure software complexity? .....	25
5.1.1 Quantitative physical software attributes (metrics).....	25
5.1.2 Qualitative software attributes (metrics) .....	26
5.1.3 Software complexity in embedded system software .....	26
5.2 Evaluation and selection of software complexity metrics .....	27
5.3 Data sources for software complexity metrics .....	28
5.3.1 Data source: Telelogic Logiscope audit .....	29
5.3.2 Data source: Audit by CV software development team .....	30
5.4 Findings and conclusions on available complexity data .....	32
5.4.1 Available code metrics for the code complexity model .....	32
5.4.2 Reducing the number of software complexity metrics .....	33
5.5 Validation of the used audit programs Logiscope and JAH.....	36
<b>6. Software complexity versus software reliability</b> .....	<b>38</b>
6.1 Expected relation between SW complexity and reliability .....	38
6.2 Correlations with reliability metrics.....	38
6.3 Regression Models .....	40
6.3.1 Explaining the variance in reliability metrics by regression analysis.....	40
6.3.2 Predicting reliability metrics by the regression model .....	41
6.4 Explaining variance within reliability by complexity classes .....	42
6.5 Conclusions on software complexity versus reliability.....	44
<b>7. Hardware complexity versus reliability</b> .....	<b>45</b>
7.1 Hardware reliability performance indicators.....	45
7.2 Hardware complexity and selected complexity metrics.....	45
7.3 Relationship between hardware complexity and reliability .....	48
7.3.1 Expected relations between HW complexity attributes and reliability .....	48
7.3.2 Verify assumed relations between HW complexity and reliability .....	49
7.3.3 Variance within reliability after complexity classification.....	50
7.4 Conclusions on hardware complexity versus reliability.....	51
<b>8. Conclusions &amp; recommendations</b> .....	<b>52</b>
8.1 Conclusions .....	52
8.2 Recommendations .....	54
<b>References</b> .....	<b>55</b>
<b>List of appendices</b> .....	<b>60</b>

# 1. Company profile

This chapter starts with a brief introduction of this master thesis, followed by a short description of the company Philips Healthcare and the investigated product within this company.

## 1.1 Introduction

This master thesis is a feasibility study on the reliability factor “system complexity” for the Cardio Vascular (CV) X-ray scanner of Philips Healthcare (i.e. Philips Medical Systems). It investigates how to define the system complexity for this system and if it actually influences the reliability performance of this system.

The first chapter of this thesis describes the company and the product that is investigated. The second chapter describes the research outline of the thesis. Chapter three describes the research variables and the design of the research concept; it also separates the research into software complexity and hardware complexity.

Chapter 4 describes how to define software reliability and chapter 5 describes how to define and measure software complexity. In chapter 6 the relation between software complexity and reliability is tested in several ways.

Chapter 7 is subdivided into three parts, the first paragraph describes how to express the Hardware reliability performance, the second paragraph describes how to define and measure hardware complexity. The last paragraph investigated the relationship between hardware reliability and complexity.

Finally, the last chapter contains answers to the research questions and recommendations for the CV development department and further research.

## 1.2 Royal Philips Electronics

Royal Philips Electronics is one of the world leading electronics companies. Philips started as a lamp factory in 1891 in the centre of Eindhoven and has developed into a multinational with annual sales around € 27 billion in 2007. Philips is divided into four business divisions which are; Consumer Lifestyle, Healthcare, Lighting, and Innovation & Emerging Business Group. Together these four divisions employ around 125,000 people over 60 countries.

The corporate mission is to “Improve the quality of people’s lives through timely introduction of meaningful innovations”, assisted by the following corporate vision “In a world where complexity increasingly touches every aspect of our daily lives, we will lead in bringing sense and simplicity to people”.

## 1.3 Philips Medical Systems

Philips first medical activities started in 1918, when it started to develop its first X-ray tube. In 1929 they sold their first Medical X-ray tube; called the Rotalix tube. Since 1993 Philips also started to manufacture other medical equipment. Nowadays Philips Medical Systems (PMS) is next to GE and Siemens a global leader in: diagnostic imaging systems, patient monitoring and cardiac devices, and healthcare information technology solutions. In specific PMS is a leader in imaging systems which also represents a major part in the total annual sales as shown in Figure 1.

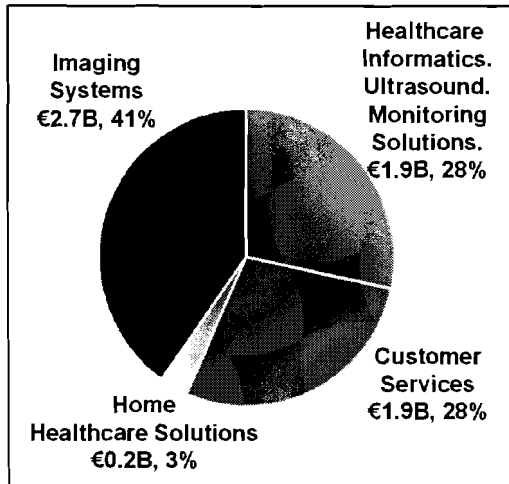


Figure 1 Annual sales per product group

PMS employs around 30,000 people over 60 countries and has a total sale of around € 6.7 billion in 2007. PMS headquarter is located in Andover, United States of America. The business structure of PMS is shown in figure 2. This master thesis takes place at the Cardio Vascular business line which is highlighted in Figure 2. Within the Business Line Cardio Vascular (CV) this master thesis is positioned in the sub-department “Development”.

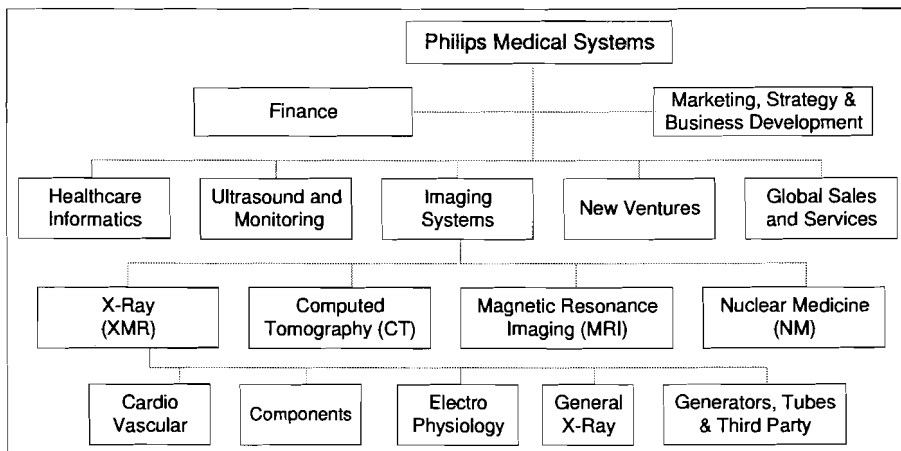


Figure 2 Organization chart Philips Medical Systems

## 1.4 The Allura Xper Cardio/Vascular X-ray scanner

The CV Business Line makes the Allura Xper product line which is the successor of the Integris Allura product line. The Allura Xper product line can be subdivided into cardio and vascular x-ray systems and into Mono and Bi-plane systems (Table 1).

Name	Mono / Bi plane	Usage
Xper FD10	Mono-Plane	Vascular imaging system
Xper FD10/10	Bi-Plane	Cardio imaging system
Xper FD20	Mono-Plane	Cardio imaging system
Xper FD20/10	Bi Plane	Vascular / Cardio imaging system

Table 1 Systems within product line Allura Xper at PMS

These four types of systems are used in the following clinical areas:

- Interventional Cardiology
- Pediatric Cardiology
- Electro Physiology
- Interventional Neuro-radiology
- Interventional Radiology
- Diagnostic Radiology and Neuro-radiology

These systems are selected for the case study because of the following reasons:

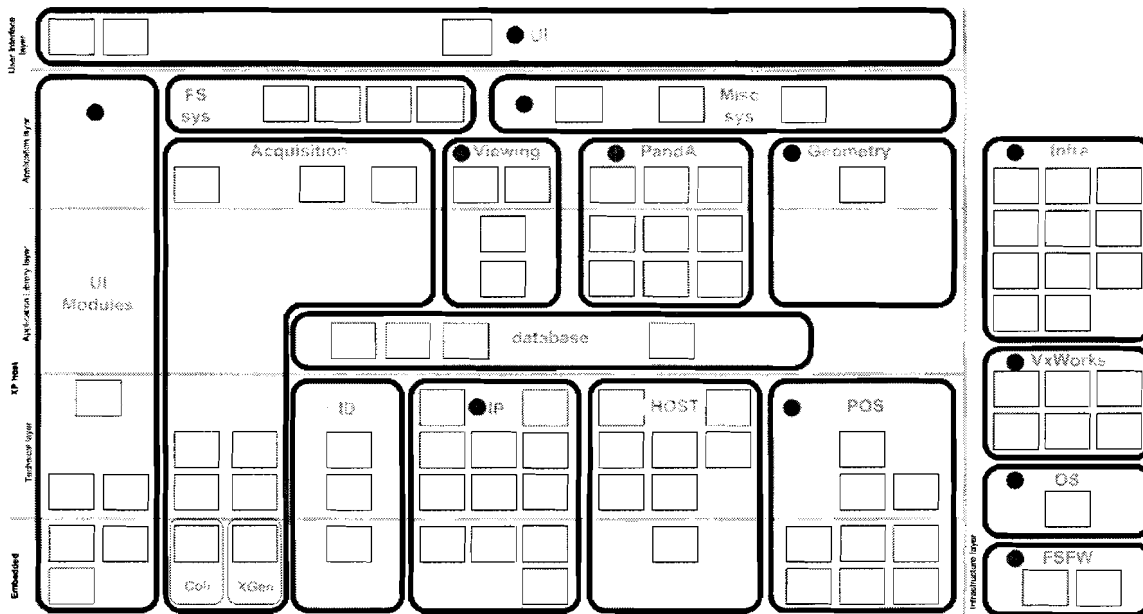
- These systems apply highly innovative and complex technologies and are used in a professional environment by professional operators (Interventional radiologists).
- These systems are capital investments and therefore are repaired instead of replaced when they are failing.
- PMS offered its time, experience and support to investigate this research concept.
- PMS is a large enterprise with developed and monitored business processes which makes generating the required data more easily.
- This Master thesis is part of a PhD project which focuses on professional repairable systems.

**System architecture of the Allura Xper product line**

All systems within the Allura Xper product line follow the same subdivision into subsystems (modules). However, there are two division methods applied; one division in Hardware subsystems and another division into Software subsystems. This difference in architecture between Hardware and Software is required because the Hardware subsystems cannot be mapped “one on one” with the Software subsystems. Both subdivisions are briefly described in the following subparagraphs.

*Software subsystems:*

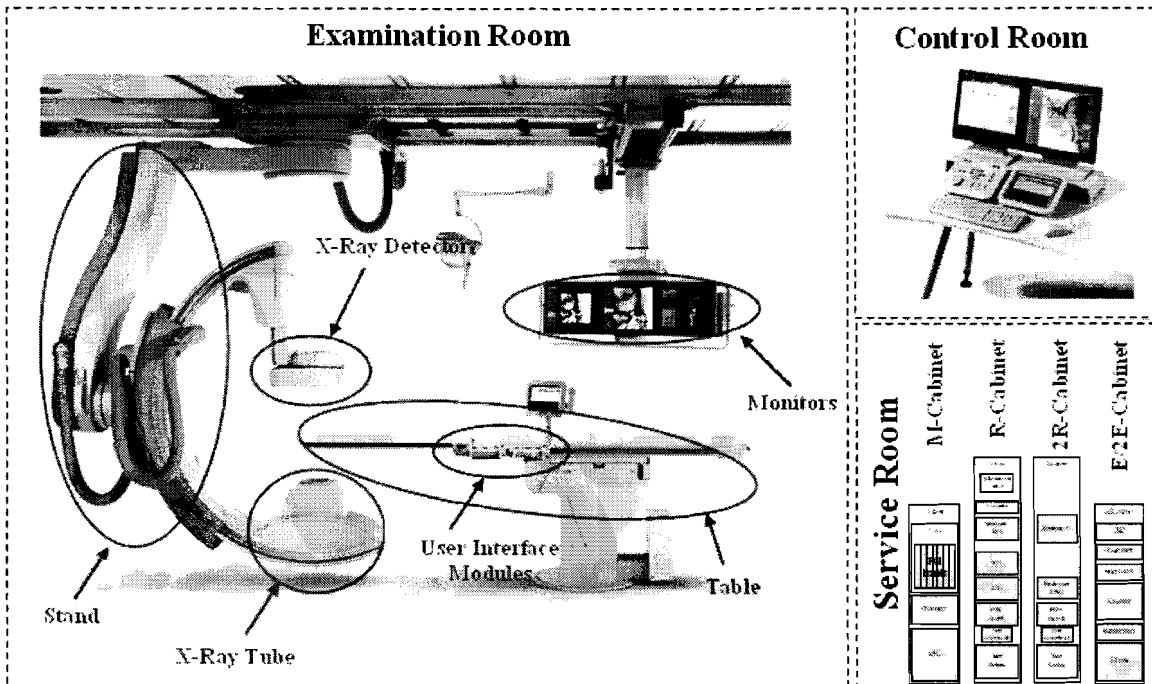
Figure 3 shows the subdivision in Software subsystems, each red encircled block represents one subsystem that consists out of one ore more software units. These subsystems and their function within the system are explained in appendix 1.



**Figure 3 Software subsystems and units (Source: System Design Specification Rocket C2)**

**Hardware subsystems:**

From a hardware perspective the system can be subdivided into three groups of subsystems which together form the Allura Xper CV scanner, these are: the Examination room, Control room and Service room. (Figure 4). The Examination room is where the actual CV X-Ray system is placed and where the patient is examined by the Interventional radiologist. In the Control room the pictures from the examinations can be reviewed by the radiologists and other hospital employees like for example surgeons. The Service room contains Software cabinets that contain all control and operating software for the CV X-ray scanner in the Examination room.



**Figure 4 Hardware building blocks (Source: System Design Specification Rocket C2)**

The Examination room can be divided in the following “building blocks” on which the system is build up:

- **Stand:** the stand holds the C-arm and the connected X-ray detector and X-ray tube. The stand can be placed on the Floor or hung on the Ceiling. (Floor stand and Ceiling stand) The selected stand depends on if the system will be used for cardio or vascular applications.
- **X-ray tube and X-ray Detector:** the X-ray tube creates the X-ray beam which is detected and converted into a digital signal by the X-ray detector. The data is sent to the Cabinets in the Service room, which further process this data into graphs and pictures that are sent back to the Examination room and the Control room.
- **Table:** the table holds the patient and can be turned in all kind of positions in such a way that the patient is correctly placed between the X-ray tube and the X-ray detector.
- **Monitors and User Interface modules:** The pictures that are retrieved from the scan are shown on the monitors in the Examination and Control room. The User Interface modules are the control tools like hand and foot switches with which the Interventional Radiologist can direct the system.

A list of all the Hardware subsystems is presented in appendix 1.

## 2. Research outline

The research outline starts with a description of the research area followed by the description of the problem situation. These two paragraphs explain the relevance and necessity of this Master Thesis. Subsequently, the research objective is established and the accompanying research questions are defined in the paragraphs 2.3 and 2.4. Paragraph 2.5 research approach, describes the followed methodology. Finally, some general research limitations and boundaries of this Master thesis are described in the last paragraph.

### 2.1 Formulation of the research area

This Master thesis project is an addition to the Ph.D. project “Monitoring and Predicting the Reliability Behavior of Early Failures in the Field based on Decision Rules” (Kevrekidis, 2007). This project will focus on the first two modules of the Ph.D. project solely.

To determine system reliability of a high complex technology system there are several factors that have to be taken into account. Some of these factors are already investigated thoroughly and their impact on reliability is well known, like physical processes and maintenance. (Wong, 1990) Generally, these “reliability factors” and their impact on reliability are evidenced by experiments in which operating conditions are simulated. However, these factors are not solely responsible for the reliability performance of the system and in many situations (like in the case of professional complex systems) carrying out an experiment in which factors can be kept constant in order to test the relation between a single factor and reliability is impossible. Today, the actual reliability performance level still cannot totally be explained due to unknown reliability influencing factors. Therefore accurate predictions regarding system reliability are still not available. (Wong, 1990; o’Connor, 1991; Ascher and Feingold, 1984)

Especially reliability prediction for the early use phase becomes more complex due to “hidden zero-hour” failures (type 1) and “Early wear-out” (type 2). There are a few trends which cause an increase in these failures (Brombacher et al., 2005):

- A strong pressure on “time-to-market” requires reliability assurance in an early phase of the product development to approve the product launch. This early reliability prediction has to be based on the field performance of preceding product versions. Due to the short time-to-market of the new version there is little time to generate (reliable) information to correct/improve the product design.
- Increasing customer demands on product quality and reliability, results in more customization and more diversity in end products. This increased diversity due to complex product customization complicates the reduction of failure types 1 and 2.
- Another trend in current product development is an increased number of product features and options, which results in more complex products (Calvano and John, 2003; Prasad, 2001; Doty, 1989; Brombacher, *et al.*, 2005). It is plausible that this increase in “system complexity” has a (negative) impact on system reliability.

A more accurate explanation for the actual reliability for the early life phase will have the following main benefits (o’Connor, 1991):

- Advance knowledge about reliability can improve the predictions of warranty costs, spare parts requirements, support costs, and marketability of the new product (version).
- Reliability prediction can be used for comparing options and highlighting critical reliability features of designs. This can improve the product design process.
- Might eliminate early life failures. A reduction of these early life failures is important for the manufacturer because it will reduce the repair costs and time due to early life failures within the warranty period.

## 2.2 Problem situation

There are several reliability factors that are still relatively unknown or their impact on reliability is not known. Some examples are: supplier quality differences, environmental stress factors, system complexity and system age. (o'Connor, 1991)

The following phenomenon supports the statement that these relatively “unknown” reliability affecting factors lead to an increased impact on the system’s reliability performance. This phenomenon is “**high variance in the reliability performance for the same types of systems**”. (Communications with Ir. Kevrekidis, Ph.D. student at TU/e) The systems selected for this research are the Philips Allura Xper CV X-ray scanners, which are complex and professional repairable systems. This large variance in the reliability performance of these Allura Xper systems cannot be explained by the traditional reliability affecting factors; physical processes and maintenance. Extended knowledge on the “unknown reliability factors” should enable a classification of the systems according to these factors in order to gain less variance in the reliability performance of each classified group. It is expected that this classification enables better understanding of the problems/issues that cause the high variance in reliability performance.

The phenomenon “high variance in reliability performance” is tested based on field reliability data of four panels of Allura Xper CV X-ray systems. This problem analysis shows that there is indeed a high variance among the systems reliability performance. The data and analysis on which these conclusions are based can be found in appendix 2.

The relatively “unknown” reliability affecting factors that could explain this variance in reliability performance are identified in the Ph.D. research by K. Kevrekidis. This PhD project applies the (social sciences) “Grounded theory” by Glaser & Strauss (1967) to select the relevant factors that are taken into account. This master thesis assumes that these reliability factors found with this theory are truly relevant factors. One of the reliability factors found by this method is “**System complexity**” and is selected for further investigation in this Master thesis project. A literature study and a research proposal on the factor “system complexity” are carried out as a preparation on this Master thesis. (Albers, 2008)

The **aim of this Master thesis** is to investigate the relation between the factor “system complexity” and the systems reliability performance. Therefore, this research should provide a methodology how to measure/quantify the reliability factor “system complexity”. Finally, this research should establish if a system classification according to “system complexity” can be made and if it actually reduces the variance of reliability performance within subgroups among the **Allura Xper CV X-ray systems at PMS**.

### Problem definition:

*More quantitative insight in the relationship between system complexity and the system’s reliability performance level is required in order to determine if system complexity can actually explain the variance within the reliability performance of professional systems.*

## 2.3 Research objective

The type of research that is carried out depends on the chosen research objective and corresponding research questions. According to Verschuren en Doorewaard (1995) there are two main research types: theoretical research and practical research. The difference between these two approaches is as follows (Verschuren en Doorewaard, 1995):

- *Theoretical research:* also called fundamental research, its aim is to solve problems in the formulation of theories.
  - Theory development: motive for this type of research are hiatuses or “blind spots or gaps” in current theories or that current theories can not be generalized.
  - Theory testing: this type of research tests existing theories and if necessary adapts or optimizes the existing theory.



- *Practical research*: this type of research contributes to an intervention to solve a practical situation or problem. To construct practical research efficiently, an intervention cycle should be (partially) carried out:
  1. Problem recognition: The problem itself should be clearly defined.
  2. Diagnosis: The main causes and effects of the problem should be clearly defined.
  3. Development: development of a realistic solution that will solve the problem as defined in stage 1 and 2.
  4. Implementation: Implementing the proposed solution.
  5. Evaluative: Does the intervention really solve the defined problem?

This Master Thesis project will focus on theoretical research in order to develop a theory that describes the relationship between system complexity and reliability in a quantitative way. It focuses on the blind spot of the incompetence of reliability engineering theories to come up with reliable and accurate prediction of system performance because of a lack of understanding in the reliability affecting factors (Wong, 1990; o'Connor, 1991; Ascher and Feingold, 1984). Based on the PhD project by K. Kevrekidis, this Master Thesis assumes that theory development on the reliability factor “system complexity” will increase insight in complexity and its effect on the reliability performance.

The theoretical objective of this research is to establish if there is a relation between system complexity and system reliability. The practical objective of this research is to explore how system complexity can be expressed in the case study at Philips Medical Systems (PMS) and which data is available or can be gained.

These two objectives are combined into a desired situation to which this research should lead (Figure 5). In this desired situation the reliability performance of a new product version can be:

- improved by controlling the reliability influencing system complexity characteristics (**focus**)
- **predicted** based on its already known system complexity characteristics during development.

In order to come to this desired situation the relationship between complexity and reliability must be evidenced first, which means that differences in the reliability can be **explained** by the reliability factor “system complexity”. Therefore, complexity should be expressed by a set of characteristics (metrics) that show significant correlation with system reliability performance.

The initial Master Thesis research proposal was to:

1. Create a methodology to quantify the reliability factor “system complexity”,
2. Carry out a feasibility study at PMS for the “CV X-ray scanner” in which the required data was collected for several system versions in the field, and
3. Establish if this data can be used to fulfill the desired situation as expressed in Figure 5.

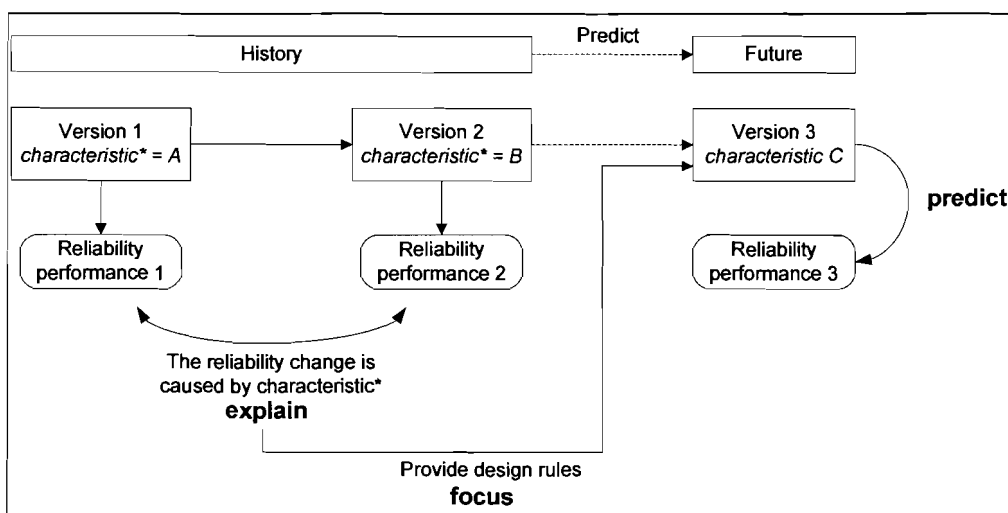


Figure 5 The desired objective of the Master Thesis Project

However, due to a time limitation and restrictions in the available data only the variables reliability and complexity are expressed in quantitative measures and the relationship between these variables are tested. So the main objective of this Master thesis is to **explain** reliability changes by complexity characteristics. Furthermore, this Master thesis concludes if and how these complexity characteristics can be manipulated and controlled in order to improve the next system version (**focus**). Unfortunately, this Master thesis cannot comply with the desired objective to **predict** the field reliability performance based on the system's complexity characteristics. The reason for this is the small number available observations which will be further explained in chapters 4 till 7.

## 2.4 Research questions

The following central research question should be answered in order to get better understanding of the relationship between system complexity and system reliability:

*What is the impact of “system complexity” on the reliability performance of the professional repairable systems under study and how can this gained knowledge be assimilated into decision rules for product improvement?*

In order to fulfill the objective of this research, as described in paragraph 2.3, this research question should be specified in more detailed sub-questions.

*Sub-questions and deliverables:*

1. How should the reliability performance of the professional system be defined or measured?
  - This research should define the way in which the reliability performance is measured.
2. How should “system complexity” be defined in such a way that there will arise no misconceptions in interpreting these factors?
  - The research should provide clear definitions of “system complexity”. This definition should enable a quantification of this reliability factor. Furthermore, the quantification should be representative for the actual system complexity.
3. How should the relationship between “system complexity” and the reliability performance of the system be established?
  - The actual included independent and dependent research variables (selection of complexity and reliability metrics) for establishing the relationship between system complexity and the reliability performance need to be defined.
  - The expected relations between “system complexity” and “system reliability performance” should be based on theoretical investigation (literature study) first before it's quantitatively investigated.
  - The established relationship should be based on the case study at PMS.
4. How to create a product classification process based on system complexity?
  - A classification model for the professional repairable systems should be designed based on the investigated reliability factor “system complexity”.
  - Investigate if the variance of the reliability performance level within a classified group is smaller than the variance of the reliability performance for the group without a classification according to complexity.

*This classification process highly depends on the actual relationship between system complexity and system reliability. When no relationship is found, a classification of the systems according to complexity won't result in more homogenous system groups.*
5. How can extended knowledge about “system complexity” and its relation with reliability performance assist the decision process in product development?
  - The impact of “system complexity” on the system's reliability performance should be translated into design decision rules for the development process of each classified group.
  - This research question is related to the PhD project and represents the higher aim of this research project.

## 2.5 Research approach

This research applies the Kumar's (1999) research methodology. This methodology is specifically suitable for a theoretical research, which is the case in this Master thesis project.

The methodology divides a research in several research phases and provides a format of the content and activities for each phase. These phases are visualized in Figure 6, on the next page, and each of them is briefly explained within this research structure. A more detailed and chronological description of each research phase is described in appendix 3.

Figure 6 should be interpreted as follows:

- The column "projects" contains the three subprojects within the Master thesis. The first two projects "Preparation Master thesis Part 1 and Part 2" are carried out at the university, the last project "Master thesis" is carried out at PMS within the development department of the Business Line "Cardio Vascular".
- The column "project phases" contains the just described seven main phases of the general research methodology provided by Kumar (1999). The column "Actions" contains the concrete actions that were carried out or were required during each research phase.
- The last column "Chapter" shows in which chapters the research phases are described. The project phases 3 till 6 are carried out and described separately for the two main areas within "system complexity": "software complexity" (Ch. 4, 5 6) and "hardware complexity" (Ch. 7).

## 2.6 Research boundaries

The case study restricts itself to the Cardio/Vascular department of PMS, and therefore also to the available data and knowledge within this department. The initial plan was to develop a methodology to express/measure the complexity of the system for the system's hardware as well as the software. Due to the limited available time, this thesis elaborates much more on the methodology to measure "software complexity" than "hardware complexity". The concept of measuring "qualitative structure complexity" hasn't been touched at all, due to difficulties with measuring and gaining the required quantitative data.

The main reason for selecting software complexity above hardware complexity was the availability of reliability related metrics. The reliability of the systems could initially only be expressed in component replacements and the initial root cause (hardware or software failure) was unknown. The PhD project started on retrieving reliability metrics for the system software by analyzing system log-files. Furthermore, software complexity metrics were easier to collect than hardware complexity metrics. The research on software complexity versus software reliability would provide quick results that might endorse further research into system complexity.

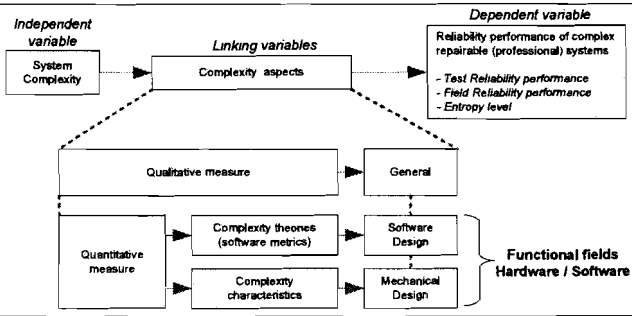
Projects	Project phases	Content / Actions	Chapter
Preparation on Master Thesis (Part 1 Literature review and Part 2 Research proposal)	1. Formulation of the research problem	<p><b>Problem definition:</b> The influence of several factors on the system reliability performance are unknown. To gain better insight in the causes of reliability performance, one of these factors (system complexity) and its effects on reliability performance is investigated.</p>	CH 1
	2. Designing the research concept	<p><b>Research question:</b> How can the impact of "system complexity" on reliability performance of professional repairable systems be measured and how can this knowledge be translated or assimilated into decision rules for product improvement?</p> <p><b>Identifying variables and constructing hypotheses:</b>  <i>Independent variable:</i> complexity in perspective of hardware &amp; software  <i>Dependent variable:</i> reliability performance in test &amp; field failures, and system entropy (seperated into hardware and software failures)</p> 	CH 2  CH 3
Master Thesis Project (Case study at PMS)	3. Construction of the data collection instrument	<p><b>Definition of required data regarding:</b>  - <i>Reliability:</i> establish the reliability performance and its variance  - <i>Complexity indicators:</i> with respect to hardware and software</p>	CH 4 + CH 5 + CH 7
	4. Selection of sample	<p><b>Sample:</b> Cardio/Vascular X-ray scanner at Philips Medical Systems (PMS)  - Selecting the group of system types with the required data available</p>	
	5. Create the research proposal	<p><b>Definition of sub questions for the selected samples</b>  <b>Definition of variables and its indicators that are desired in theory</b>  <b>Expected relations to be found between the indicators</b></p>	
	6. Collect data	<p><b>Data collection at the PMS department Cardio/Vascular (CV)</b>  - Selection of the system types and systems for which the required data is available (separately for Hardware and Software)  - Specify (for each variable or indicator) which or what kind of data is required and how this data can be retrieved  - Selection of the required data sources: Databases &amp; Interviews</p>	
	7. Process data	<p><b>Data processing and testing the hypotheses</b>  - Selecting the required statistical procedures  - Define/monitor the reliability/validity of the tests and results  - Results: answer the research question and sub questions  - Conclusions and remarks on the results</p>	CH 6 + CH 7
	8. Construct the report	<p><b>Conclusions and recommendations</b> based on the research findings</p>	CH 8

Figure 6 Research structure Master Thesis

### 3. Designing the research concept

Generally, one assumes that there is a logical relationship between system complexity and system reliability. Much research has already been carried out in the investigation if “system complexity” affects “system reliability” and if complexity can even be used to predict system reliability. In literature (Albers, 2008), over 30 researches were found that discuss or are related to investigation of “system complexity”. When reviewing the literature for software complexity in chapter 4, even more researches have been found.

However, for “hardware” related complexity not much quantitative research on the relationship “hardware complexity and reliability” has been carried out. In contrary, much quantitative research on the relationship “software related complexity and reliability” has been carried out and many types of software complexity metrics are defined. However, these researches come to different and sometimes contradicting conclusions about which metrics are important and how these metrics influence reliability.

In order to come to good research hypotheses, the relevant variables and the assumed causal relation should be defined first (paragraph 3.1). To reduce the bias in the conclusions about the relationship between complexity and reliability, the other “reliability influencing factors” should be kept constant (paragraph 3.1). Furthermore, the vague variable “system complexity” needs to be decomposed into several subareas within this research topic like “hardware and software complexity” (paragraph 3.2). Finally, a short introduction on the selected sample and applied data collection methods is given (paragraph 3.3).

#### 3.1 Defining the expected causal relationship

One of the first research steps (Kumar, 1999), is to identify the variables of interest and quantify them in metrics that represent the level of these variables. However, the perspectives from which these variables should be analyzed need to be defined first. Kumar (1999) describes these perspectives as followed:

- The causal relationship;
- The design of the study (paragraph 3.3.1);
- The unit of measurement (chapters 4, 5 and paragraphs 7.1 and 7.2).

The causal relationship under investigation in this thesis is the relation between the assumed relevant *independent variable* “system complexity” and the *dependent variable* “system reliability performance”. There are several possible *extraneous variables* that could moderate the research outcome as well. When selecting the research design and/or the method of data collection, it is extremely important that these extraneous (/moderating) variables are controlled and do not influence the results from the tested hypotheses.

Based on the literature study and communications with: Kevrekidis, Sonnemans and Stollman, it is assumed that the following actions need to be carried out in order to establish a causal relation between the variables complexity and reliability:

- Translate the vague term “system complexity” in measurable characteristics (metrics).
- Translate the term reliability in measurable indicators.
- Establish which extraneous variables might influence the relation between complexity and reliability and define how these can be controlled.

Figure 7 visualizes the causal relation that is under investigation. The vague term “system complexity” is an aggregated research variable that can be decomposed into several sub variables. This decomposition is explained further on in paragraph 3.2.

Translation of the vague terms “system complexity” and “system reliability” into measurable characteristics or metrics are carried out separately for Software complexity in chapters 4, 5 and 6 and Hardware complexity in chapter 7.

The last requirement “keeping the extraneous variables constant” is tackled as good as possible upfront in this paragraph. Based on communications with G. Stollman and K. Kevrekidis, the next two extraneous variables are identified as interfering variables and were attempted to keep constant in the following way:

- *Constant system environment:* the complexity characteristics are defined on subsystem (module) level which means that all subsystems are used in the same system environment. Each system is placed within a hospital; however this does not automatically mean that each system interacts with similar environmental stresses. The environmental stresses in Asia might differ from Europe because of differences in the climate. Also governmental regulations on how to use or maintain CV systems are different for each continent or even each country. To reduce the influence of this extraneous variable “system environment” as low as possible all analyzed systems are placed within the United States of America.
- *Constant user profile:* the systems are stressed in a different way among different users. The clinical areas in which the system is used were already mentioned in paragraph 1.3, the CV systems are stressed differently in each clinical area and users sometime use systems for a combination of clinical areas. Although it is not possible yet to separate the systems according to these clinical areas or user profiles. The Allura Xper systems can be divided into the two main groups of cardio and vascular systems. At the moment this is the only subdivision of systems that can keep the variable “user profiles” as constant as possible.

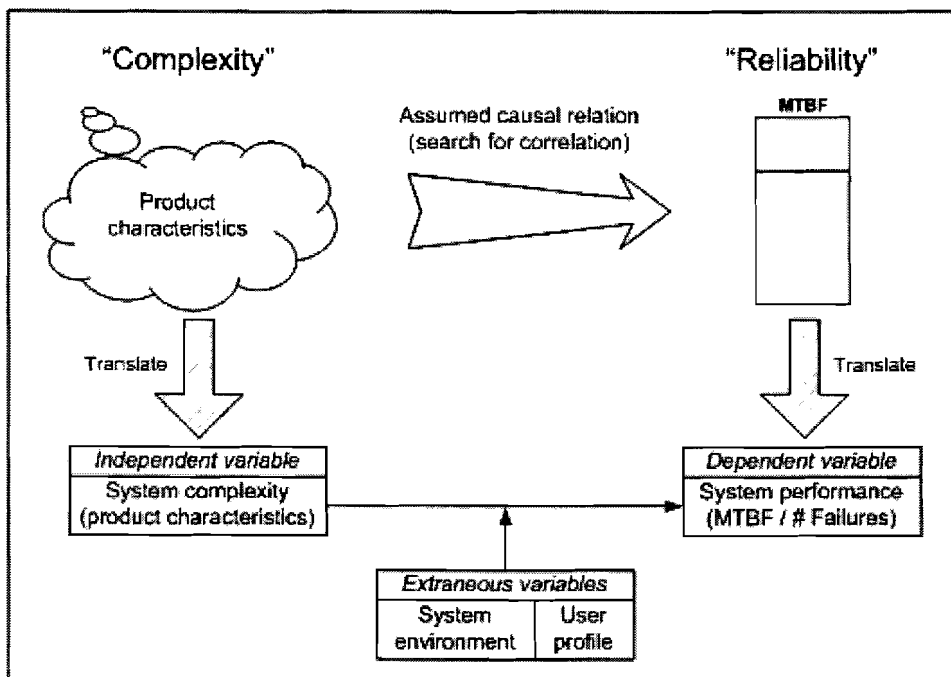
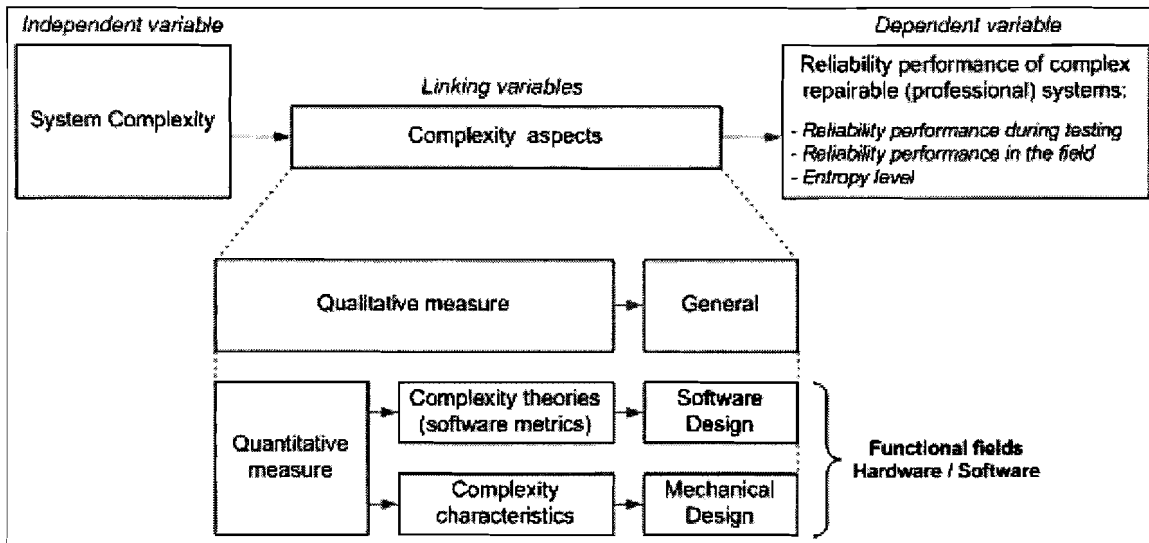


Figure 7 Research variables and extraneous variables within the research

### 3.2 Decomposition of “system complexity & reliability”

In order to establish that there is a causal relation between system complexity and the reliability performance, both variables have to be defined first. The reliability influencing factor “complexity” cannot be expressed in a single measurable parameter (or variable). Therefore, some linking variables are used to establish the relation between the independent and dependent variable. The linking variables are subdivided into qualitative and quantitative complexity concepts. These concepts are selected in the Master Thesis preparation (Albers, 2008).

Figure 8 shows both complexity concepts as well as the division of reliability performance into sub measures. Both complexity concepts and the reliability sub measures will now briefly be described in the following paragraphs.



**Figure 8** Decomposition of research variables

### 3.2.1 System complexity

*Qualitative complexity concepts:* system complexity is expressed in (Flood and Carson, 1990):

- The *system symmetry*: structure of control system;
- The *non-holonomic constraint*: the system's ratio of the total number of degrees of freedom against the number of controllable degrees of freedom;
- The *perception of engineers/users*: combining the level of education and the diversity in required specialists.

*Quantitative complexity concepts:*

- *Complexity theories*: one of these types of complexity measures lends itself for expressing the complexity of the applied/used software in the systems. Because of the fact that software plays a crucial role into the development and design of current highly innovative professional systems (Hobday, 1998), this will be an important measure of complexity.
  - *Kolmogorov complexity*: this measure uses the length of a description in a certain language to express complexity. (Löfgren, 1977; Edmonds, 1999) The length/size of the software program/codes could be used to express the complexity of the system from the “software engineering” point of view. When software codes are unavailable or not measurable, other measures of system size can be used like the number of software systems or programming languages used.
  - *Software metrics*: these are metrics that express the complexity / size or quality of the software code. There are approximately 200 types of software metrics (Zuse, 1992), it is task to select appropriate metrics to express software complexity for a complex professional system.
- *Complexity characteristics*: there are several researchers that used this concept to express complexity (Calvano and John, 2003; Hobday, 1998; Ren and Yeo, 2006; Barclay and Dann, 2000; Lucas, 2000; Rodriguez-Toro, *et al.*, 2003). In general, these researchers defined complexity in the sense of manufacturing or assembly complexity (Rodriguez-Toro, *et al.*, 2003) or as the complexity of new innovative projects or products (Hobday, 1998). These complexity characteristics are suitable to express the system complexity from the “mechanical engineering” point of view.

The software, hardware (mechanical engineering) and their interactions are the main contributors to the development of a new product and most likely also the reliability performance of that new product. The decomposition into “hardware” and “software” is described in the next paragraph.

### 3.2.2 Decomposition into software and hardware

The hardware is separated from the software, as the building blocks differ for hardware and software. The subdivision, i.e. building blocks, of the system is described in chapter, for software as well as hardware. Due to the inconsistency in the subdivision and due to the area depended technology within software and hardware, these two technology areas should be analyzed separately. The selection of the hardware complexity characteristics and software complexity metrics (mentioned in Figure 8) should be made correctly. The following criteria will be used in order to select the correct and representative hardware complexity characteristics and software metrics:

- The characteristics must be measurable
  - Required data must be available at the CV department of PMS
  - Processing the data must be feasible within the Master thesis time limitation
- Characteristics must be derived from relevant (academic) literature
- The assumption that there will be a relationship between the complexity characteristic and reliability performance must be realistic which means that it should be based on some theories or logical/straightforward assumptions.

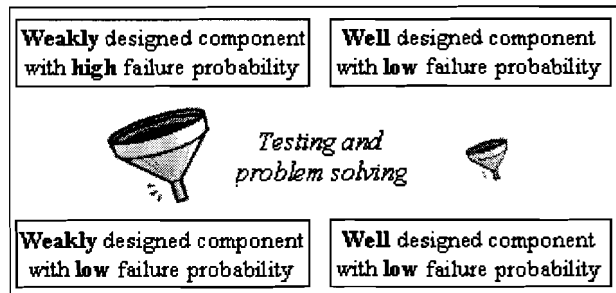
The next problem that should be tackled is to identify categories of complexity. The number of complexity levels (for example: “high, middle, low” or “continuous”) has to be determined. Furthermore, weight factors for each contributing metric to the aggregated measure of complexity should be defined.

### 3.2.3 System reliability and reliability metrics

The system’s reliability performance should be expressed in quantitative metrics in order to find correlations between the complexity metrics and reliability metrics. The following reliability metrics are desired in order to test the relations with complexity:

- *Failure occurrence in the field:* The reliability performance of the system in the field should be expressed in the number of failures found during a certain period of time and if possible the Mean Time Between Failures (MTBF) of the systems should be derived. The MTBF provides a more fair comparison between systems, because not all systems are observed for the same period of time and are operated in the same extent within a certain time period. Therefore the MTBF expresses the reliability field performance of the system in the best way.

- *System performance before testing:* All new developed systems are tested before they are launched for field use. The extent and thoroughness with which the system or subsystem is tested influences the reliability performance in a great way. Very complex or weakly designed subsystems may perform in the same way as “well designed” subsystems because they all have the same expected failure



- *Entropy:* The entropy measures the chaos in the system, which for this master thesis is expressed into the number of system failure types (states) and the distribution of failure over these failure types. The more failure types and the more homogeneous the failure distribution over these failure times, the higher the entropy level and the less reliable the system will be. However, this chaos in the system can also be seen as a form of complexity as well. Therefore entropy is the perfect reliability measure in order to verify if the found relationship between complexity metrics and reliability are logical. More information about entropy can be found in appendix 4.



### **3.3 Sample selection & Data collection**

The first paragraph describes the selected study design and sample, the next paragraph describes the applied methods in order to collect the required data.

#### **3.3.1 Selecting a study design and sample**

The study design in this Master thesis can best be described as a fundamental research in which the effect of system complexity on system reliability is investigated according to a comparison of this theoretical concept (assumption) with empirical data (*Primary empirical comparison*) that is generated from existing data (log)files (*Desk research*). This study will be carried out as a *single case study* on the CV X-ray scanner at PMS, the established relationship between complexity and reliability will be based on this case solely. Therefore, the developed model is solely applicable for the Allura Xper systems at PMS.

#### **3.3.2 Selecting a method of data collection**

In order to establish the relationship between “system complexity” and reliability, quantitative data will be preferred. However, the concept complexity is rather vague and can be interpreted in many ways (Albers, 2008). Besides, the “technical system” knowledge of the researcher is limited. Therefore, primary sources in the form of interviews with specialists are used to create/define realistic measures for “system complexity” and “reliability performance”. For “hardware complexity”, interviews were the only source of information in contrast to “software complexity” where more quantitative data was available for the Allura Xper systems.

## 4. Software reliability

This chapter starts with an introduction into the software version for the Allura Xper systems at PMS in paragraph 4.1. Subsequently, a short description of software quality and reliability follows in paragraph 4.2. Paragraph 4.3 describes the general process of defining and collecting reliability data. In paragraph 4.4 all available reliability data sources are described and finally the available reliability data and interpretation of the data is described in paragraph 4.5.

### 4.1 Separate software into subsystems and units

First of all some more understanding of the division of the software system into subsystems, units, Product Base Lines (PBLs) and Service Packs (SPs) is required in order to understand how to measure the reliability. The Allura Xper CV systems are subdivided into PBLs, there are two major types of PBLs which are cardio CV scanners (FD10) and vascular scanners (FD20).

The first PBL for cardio scanners are named PBL10s and are succeeded by the PBL30s, PBL50s and so on. The Vascular scanners are called PBL20s followed by the PBL 40s and PBL60s. Within each Product Base Line some versions are made, for example the PBL40 consists out of the released PBL41 and PBL43 software versions, which can be seen as upgrades and are called 4.0.0 or 4.3.0. Again for each of this software version some Service Packs (“bug solving”) were launched, these are called PBL 4.3.1, PBL4.3.2, and so on. An overview of all software versions and service packs can be found in appendix 5. An overview of the selected scanners (FD20-PBL40 scanners) can be seen in Table 2.

PBL/ Project	Commercial Name	Software: New Version	Service Packs				
			SP-1	SP-2	SP-3	SP-4	SP-5
PBL_23/ Rocket-A	Allura Xper FD20	PBL_2.0.0/ PDC_23.25.0	PBL_2.0.1/ PDC_23.25.1	PBL_2.0.2/ PDC_23.25.2	PBL_2.0.3/ PDC_23.25.3	PBL_2.0.4/ PDC_23.25.47	PBL_2.0.5/ PDC_23.25.53
PBL_41/ Rocket-B2	Allura Xper FD20	PBL_4.0.0/ PDC_41.12.0	PBL_4.0.1/ PDC_41.13.4	-	-	-	-
PBL_43/ Rocket-B2+	Allura Xper FD20	PBL_4.3.0/ PDC_43.10.0	PBL_4.3.1/ PDC_43.10.14	PBL_4.3.2/ PDC_43.10.22	PBL_4.3.3/ PDC_43.10.39	PBL_4.3.4/ PDC_43.10.42	-
PBL_61/ Rocket-C2	Allura Xper FD20	PBL_6.0.0/ PDC_61.18.0	PBL_6.0.1/ PDC_61.18.12	-	-	-	-

Table 2 FD20-PBL40 software releases (Source: JAH intranet site at PMS)

For each new PBL, new or extended functionality is added to the software while the SPs are improvement changes for problems and bugs in the initial Software version or previous SPs.

### 4.2 How to measure software reliability?

First of all software reliability should be expressed in a quantitative measure. However, software reliability is rather a qualitative than a quantitative variable. There are many ways in which software reliability can be expressed and there are more methodologies to use. One way to express software reliability is the methodology provided by the ISO quality handbook (Figure 10Figure 10).

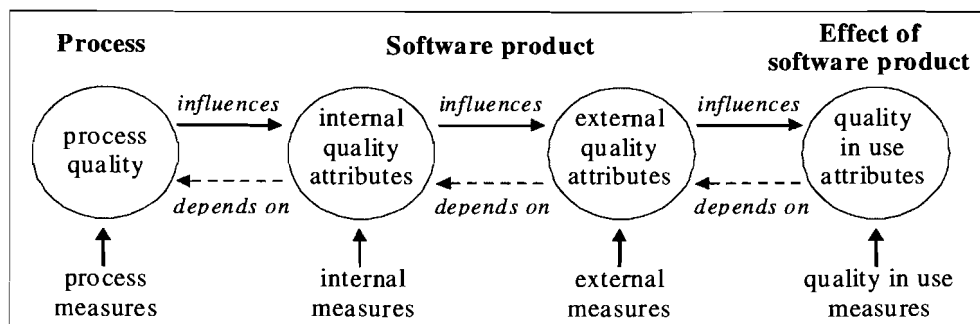


Figure 10 Quality model framework (ISO/IEC 9126-1:2001(E))

The model is built up out of process quality (the quality of the development process), internal quality attributes (static measures of internal software characteristics of the actual source code), external

The system errors need to be **filtered** first in order to create a correct representation of the software reliability performance. The following filtering is applied:

- Only programming errors are taken into account, because it is assumed that these errors are an indicator for the software reliability performance.
- Only programming errors during “normal operation” are taken into account, because it is assumed that errors outside this system state are not real errors and are caused due to system start-up or system closure and do not influence the external quality attributes of the system.
- All software errors that occur with a time to failure of zero seconds are marked as “*collateral damage*” and are excluded from the data. Assumption is that the first software error causes all the succeeding errors with a time-to-failure of zero seconds, and these failures are solely caused by the initiating programming error.

After the programming errors are filtered based on these assumptions and knowledge, the following steps in the data collection process are carried out. The filtered programming errors should be analyzed and their actual failure causes should be identified. Identifying the actual failure causes is done by constructing an Failure Mode and Effect Analysis (FMEA) based on the knowledge of system experts. The failure causes of this FMEA are mapped with the errors which are derived from the system Log-files. Based on the refined programming errors derived from the log-files and the FMEA a relation between the programming errors and their probability that they result in a system failure or system crash is known.

#### Limitations

However, this data analysis is very time consuming and extended programming skills are required. Although this thesis has supported the PhD research by Kevrekidis in order to gain this data, this Master thesis is solely responsible for the analysis of the data. Therefore only the system errors are used within this research, they are filtered and then used to calculate the MTBFs based on these programming errors. The FMEA and mapping of SWFs and crashes with the logged programming errors took too long and therefore the results from this analysis were available very late. The only reliability measures that could be used for data reduction by factor analysis (paragraph 5.4.2) are the indicators “MTBF based on programming errors” and “the amount of Programming errors” instead of the actual SW failures or crashes. However, for investigating the relation between SW complexity and reliability in chapter 6, also the indicator “software failure” (derived by translating the programming errors with the FMEA) was available.

#### 4.4.2 Data source: Problem reports

Next to the SW errors derived from the Log-files, the SW failures found during testing will be used as well. Figure 14 visualizes the relationship between complexity and reliability with “testing” as a moderator.

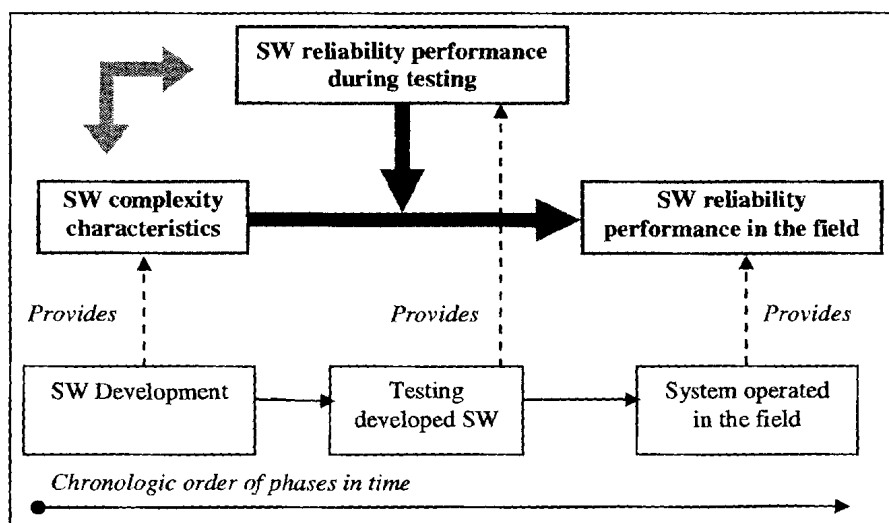


Figure 14 Development phases and corresponding “complexity” and “reliability” measures

Critical software with many problem reports (PR) during testing will probably get more extended tests and this will influence the impact on the SW reliability performance in the field. As a result the relation between SW complexity and SW reliability might decrease due to extensive testing. To verify if the impact of testing truly influences the relationship between SW complexity and reliability, it is important to also express the dependent variable reliability in PRs during testing because these SW failures are less influenced by the testing filter.

The **Problem Reports** are available through the JAH intranet site at PMS, there is an overview of all the PRs and Change Requests (CRs) for each specific subsystem within the system SW. The CRs are not taken into account because these are changes that have to be made due to changed (customer) requirements. These CRs are no failures but are customization projects.

The PRs however, are SW failures that were detected during testing and are categorized according to their severity. These **severity categories** are: **critical, major, average, minor, and enhancement**. Critical and major PRs would affect the external quality attributes of the SW, while the minor and enhancement PRs would only lead to illogical structure or unimportant programming errors.

#### *Limitations:*

Based on communications with M. Bouts (Project manager within CV department at PMS) it can be concluded that there are standards for categorizing the PRs according to their severity. However, in practice SW developers are biased in their decisions because they tend to categorize the problems they detected as very important. Although they are important to the SW developer itself, the impact on the external attributes of the SW systems are negligible. As it is assumed that this is a general habit of all SW developers within the CV development team, the PR data will not be further processed in order to exclude this bias.

Another remark should be made about the ability to allocate a problem during testing. The level of testing and the easiness with which “problems” can be found are most likely not equal for each SW subsystem or unit. Unfortunately this noise in the PR data has not been excluded due to the scope of the thesis and time restrictions.

## **4.5 Findings and conclusions on available reliability data**

The test PRs and Field programming errors are not available on each hierarchic level and/or for each unit within that level. The SW architecture and the way it is developed are already described in paragraph 4.1. The following groups are made:

- Architecture: systems, subsystems and units
- Development approach: new PBLs (20, 41, 43 and 61) which contain “added functionality” and new service packs which contain “problem solutions”.

### **4.5.1 Source: System log-files**

The analyzed log-files are solely from PBL40 systems. There were two reasons for selecting PBL40 systems in order to start our analysis. Based on communications with SW developers at PMS it was concluded that the *log-files from the earlier PBL systems were inconsistent* and therefore less reliable for our analysis. However, the *later PBL systems have shorter log-files* because they are less long in the field, which results in a shorter observation. The PBL40 systems are the best option considered these two conditions.

#### *Available direct measures for PBL 40 systems:*

- Normal operating time during the observation period (length of the log-file)
- Operating time and errors separately for each software version (or service pack)
- Error events logged on software unit level

A sample of 30 PBL40 systems were analyzed, for all these 30 systems the log-files were filtered on the total normal operating time per system and all the error events that were logged. The error events are logged on Software unit level, there were 20 software units found. After filtering on programming errors and filtering out collateral damage, as described in paragraph 4.4.1, only 12 units with

programming errors remained. It is shown that these other 8 units do log errors. However, they only contain “collateral damage” errors or errors that are not found during “normal operation”. It is assumed that these 8 units do not contain any programming error, and therefore the Mean Time Between Failures (MTBF) will be equal to the observed time period, as will be described further on under MTBF.

### Filtered field programming errors during normal operation

After analyzing the results from the log-files for these 30 systems, it immediately stroke that a small number of systems caused the major part of all programming errors. However, not all systems were reviewed for the same time period; therefore the number of programming errors should be normalized first in order to proof this conclusion. These “normalized” programming errors are calculated by dividing the number of programming errors by the normal operation time of the system and multiplying this by 1500 hours. Extended data can be found in Appendix 7. The Pareto of the “normalized” programming errors is shown in Figure 15. Nine systems (30% of the sample) cause around 80% of the total amount of “normalized” programming errors.

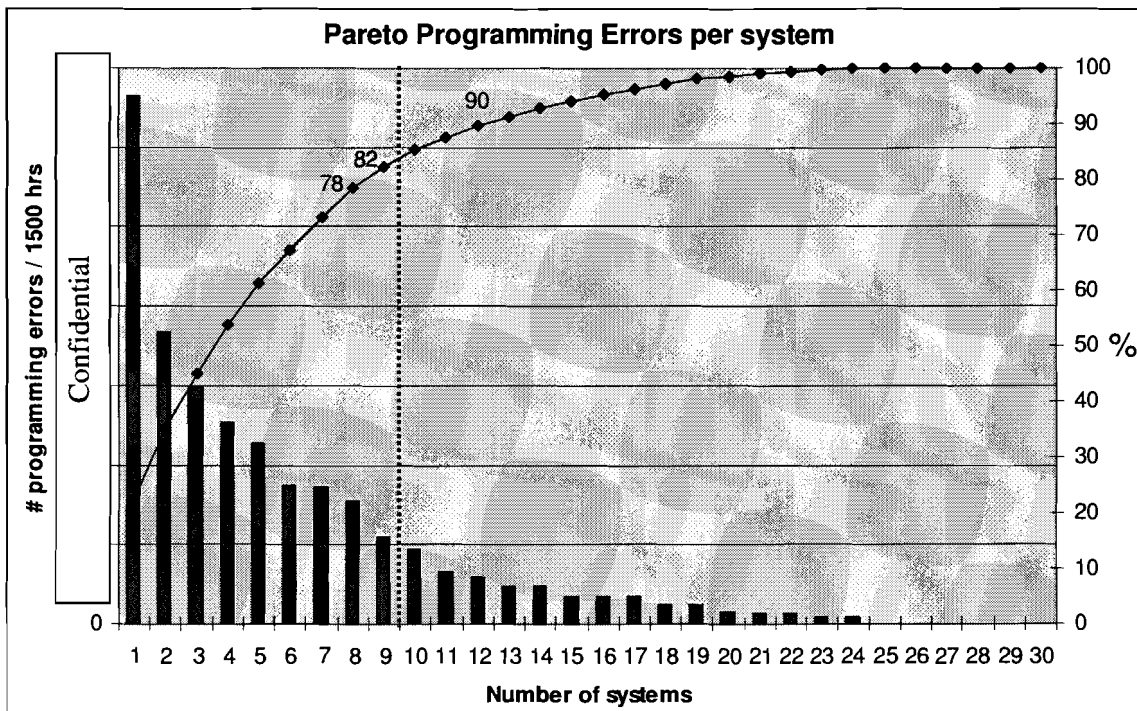


Figure 15 Pareto on “normalized” programming errors per system

Besides, a small group of software units also produces considerably more programming errors than the other units. Four software units (20% of the sample) cause around 80 percent of all field programming errors. The Pareto analysis for the software units can be found in Appendix 7.

### Mean Time Between Failures (MTBF)

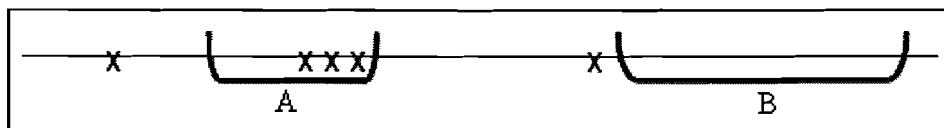
The MTBFs are calculated for each system and also for each software unit separately. The MTBF for each system (i) is calculated by dividing the “Normal Operating Time” (NOT) for each system by the number of filtered programming errors (FPE) plus one.

$$\text{System MTBF}_i = \frac{NOT_i}{FPE_i + 1}$$

The MTBF for the software unit (u) is calculated by dividing the aggregated NOT for all systems by the number of FPE within this unit plus one.

$$\text{Unit MTBF}_u = \frac{\sum_{i=1}^{30} NOT_i}{\sum_{u=1}^{17} FPE_u + 1}$$

In this approach, the data is censored by the assumption that if no failures are found the MTBF is equal to the observation time, which might not be the case in practice (Figure 16-B). From the other side, if failures are found within a short observation period, this will result to a short MTBF while in fact the MTBF is much longer (Figure 16-A).



**Figure 16 Censored data**

This type of censored data is called “interval censored”, it can be the case that units didn’t fail within the time interval and it can also be that failures occurred within the observation interval but the previous and succeeding failures were outside the time interval (ReliaSoft, 2007). There are four approaches in order to establish that the censored data still follows an exponential distribution (i.e. constant failure rate), these are (Lewis, 1996):

- The exponential distribution might be assumed based on experience with similar research; Kitchenham (1987), Khoshgoftaar and Munson (1990A) and Ebert (1996) did research the impact of some software metrics on software reliability by the same dataset of Kitchenham (1987). This dataset does describe the software errors as changes in the software (which are assumed to be made due to software errors or failures), there are no descriptions of how they tackled this problem with censored data. Software changes can also be compared with the Test Problem Reports within this thesis, the indicator of the field reliability performance in programming errors is more different. Therefore no experience with similar datasets can be used in order to verify if constant failure rate can be assumed.
- It may be argued from the failure mode whether the failures are random as opposed to early or aging failures; the failure modes and their underlying distributions are not known.
- The exponential distribution might be identified by using one of the standard statistical goodness-of-fit criteria;
- Visualize the failure rate by probability plotting (QQ plots) to examine if it is a straight line;

The last two methods should be applied for all units where programming errors are found. However, this requires the “time between failures” for all the errors and therefore the log-files should be analyzed for each software unit separately. As this is a time consuming task, these separate filters are not applied for every software unit and “constant failure rate” is assumed.

### Entropy measures

The entropy is measured for each software unit. The unique descriptions of the “programming errors” are taken as the possible failure states in which the software unit can be. The frequency with which this unique programming error is logged is used in order to estimate the probability is in this failure state. The entropy value is calculated by the following formula:

$$H(p) = -\sum_{i=1}^m p_i \log p_i ,$$

with  $H(p)$  being the level of entropy,  $m$  being the maximum number of failure states and  $p_i$  being the probability that the system is in state  $i$ . The entropy value is calculated by the sum of all the probabilities for each possible system state (Heylighen, 2002). Table 3 shows an example how the entropy is measured for the software unit “Graphical UI”. An overview of all entropy values for the 17 investigated software units can be found in appendix 7.

Unit	Failure states	Failure description (programming error)	Frequency	p	H(p) /state	H(p)
<b>Graphical UI</b>			<b>101</b>			<b>0,320</b>
	1	"GSC"	2	0,02	0,034	
	2	"COM server not available."	80	0,79	0,080	
	3	"process terminated due to a fatal exception."	1	0,01	0,020	
	4	"starting Application failed"	10	0,10	0,099	
	5	"unspecified exception."	8	0,08	0,087	

Table 3 Entropy measure for software unit "Graphical UI"

#### 4.5.2 Source: Problem reports

The PRs are known for 10 subsystems and all units within these subsystems. (The subsystems "Image detection" and "Image processing" are not available) Besides, these PRs are also available for all PBLs, Software Versions and SPs. However, for this research only the PBL40 systems and related Software Versions will be taken into account in order to compare and verify the "field programming errors" with the "problem reports" during testing. The data is shown in appendix 7.

A Pareto analysis has been performed on the problem reports during testing. Again, a minor group of software units (3 units which is 10% of the sample of 30 units that were tested) cause almost 90% of the total number of problem reports found during testing. (Figure 17)

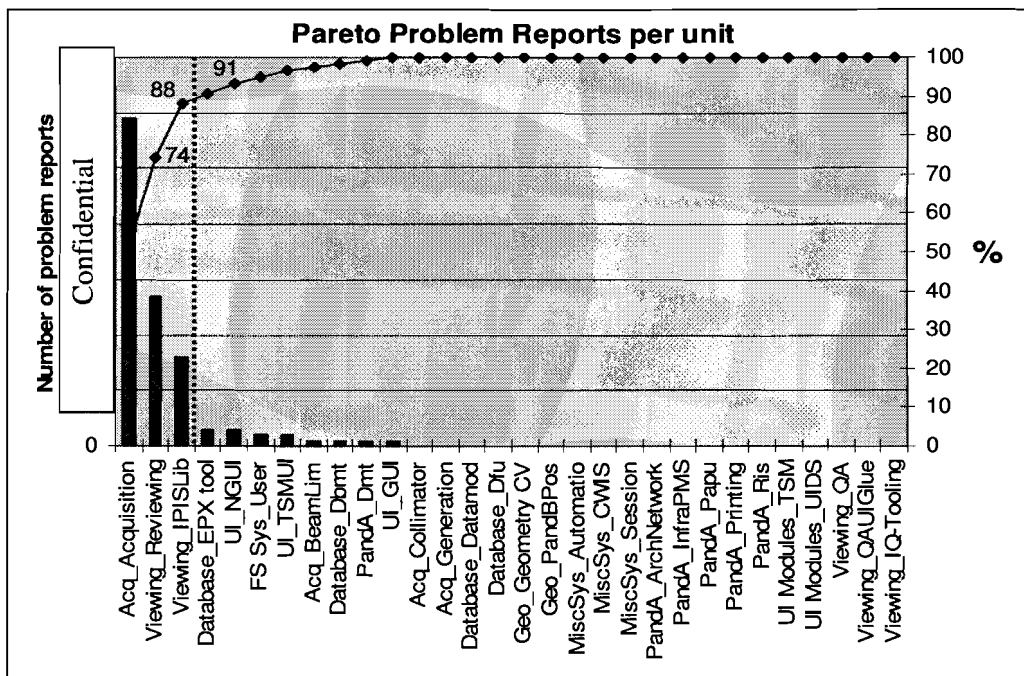


Figure 17 Pareto on Problem reports during testing per software unit

#### 4.5.3 Correlation among reliability metrics

In order to establish if the reliability metrics do correlate among each other, the Kendall's tau correlation test is applied in SPSS15.0. This correlation test is selected because of the following supportive circumstances for a Kendall's tau correlation test (Field, 2005):

- The test applies a bivariate correlation between all the reliability metrics separately
- The test is applicable on non-parametric metrics, which is the case for the reliability metrics because their underlying distributions are definitely not normally distributed (Figure 18).
- The available data set is very small (only 16 observations) with tied ranks.

The results from Kendall's tau correlation test are shown in Table 4. Based on the observed 16 software units for which all reliability metrics were available, it can be concluded that the Field Programming Errors (FPE) do correlate with the test Problem Reports (PR). In other words, software units that contain many FPEs generally also contain many PRs during testing.

Besides, the problem reports were separated into **total** PRs and **severe** PRs (only the categories "catastrophic" and "major"). Both metrics show high correlation, reason is that most of the PRs are "severe" PRs (Table 4).

Also the MTBF, which is calculated based on the FPEs and the Normal Operating Time (NOT), shows high correlation with the FPE metric. Reason for this is that the NOT is equal for all observed Software units.

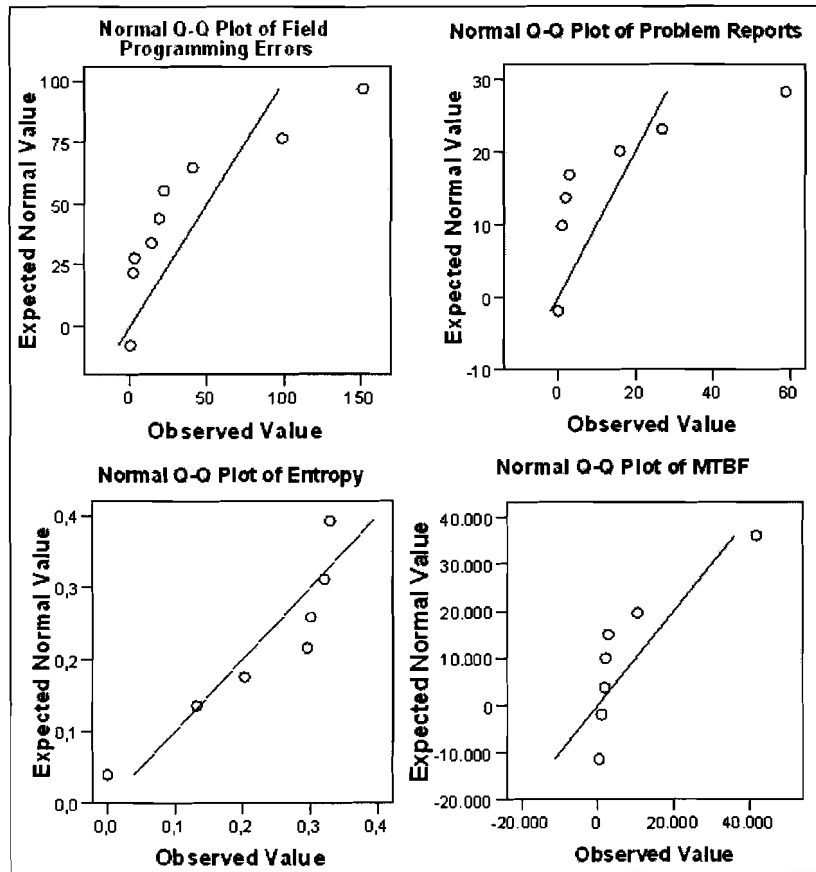


Figure 18 Normal Q-Q plots for reliability metrics

Kendall's tau Correlation test						
		Problem Reports (Total)	Problem Reports (Severe)	Programming Errors	Entropy	MTBF
Problem Reports (Total)	Corr. Coefficient	1,000	1,000(**)	,528(*)	0,127	-,453(*)
	Sig. (2-tailed)			0,014	0,655	0,038
	N	16	16	16	9	16
Problem Reports (Severe)	Corr. Coefficient	1,000(**)	1,000	,528(*)	0,127	-,453(*)
	Sig. (2-tailed)			0,014	0,655	0,038
	N	16	16	16	9	16
Programming Errors	Corr. Coefficient	,528(*)	,528(*)	1,000	0,147	-,820(**)
	Sig. (2-tailed)	0,014	0,014		0,593	0,000
	N	16	16	17	9	17
Entropy	Corr. Coefficient	0,127	0,127	0,147	1,000	0,119
	Sig. (2-tailed)	0,655	0,655	0,593		0,667
	N	9	9	9	9	9
MTBF	Corr. Coefficient	-,453(*)	-,453(*)	-,820(**)	0,119	1,000
	Sig. (2-tailed)	0,038	0,038	0,000	0,667	
	N	16	16	17	9	17

\*\* . Correlation is significant at the 0.01 level (2-tailed).  
 \* . Correlation is significant at the 0.05 level (2-tailed).

Table 4 Kendall's Tau correlation test on reliability metrics



## 5. Software complexity

Chapter 5 starts to describe several ways in which software complexity can be measured. The second paragraph evaluates which metrics should be selected for which purposes. Paragraph 5.3 describes the available data sources, paragraph 5.4 describes which software complexity metrics are selected. Finally, paragraph 5.5 validates if the metrics scores from the two sources are representative.

### 5.1 How to measure software complexity?

As part of a study after system complexity, software complexity is a narrower focus within system complexity and is less vague than the term “system complexity” itself. However, software complexity is still a broad topic within software engineering. Since the 70s there are several researchers, among others: Curtis , McCabe and Halstead, who tried to establish a definition for software complexity (McQuaid, 1996). They came up with some directly measurable attributes that express software complexity. These attributes are called software metrics and are defined and developed ever since. This evolution resulted in many software metrics, which in some cases even contradict each other. (McQuaid, 1996)

So even after thorough investigation by several researchers, it seems that there is not going to be one unambiguous method or concept that will perfectly establish the relationship between software complexity and reliability. Therefore the aim of this quantitative study between software complexity and software reliability at PMS is to establish which metrics are relevant and applicable for the professional system “Allura Xper CV scanner”.

In general, there are two ways to assess the software complexity level, which are a quantitative and a qualitative approach. (McQuaid, 1996) Paragraph 5.1.1 provides an overview of the available quantitative software complexity metrics developed over time. The more qualitative software complexity attributes are briefly discussed in paragraph 5.1.2. The quantitative approach is the most frequently applied methodology, because qualitative attributes are much less precise and are an indirect measure (McQuaid, 1996).

#### 5.1.1 Quantitative physical software attributes (metrics)

According to Baker (1991) the use of applying software metrics is to provide engineers with tangible evidence that the software programs are meeting their expectations and requirements. The use of metrics reduces the subjectivity in establishing the software’s performance by providing objective, visible data that reduces the guesswork in software development. (Baker, 1991) Which metrics are required depends on the basic uses of software metrics, which can be:

- Project estimation
- Quality control
- Process analysis
- Product engineering

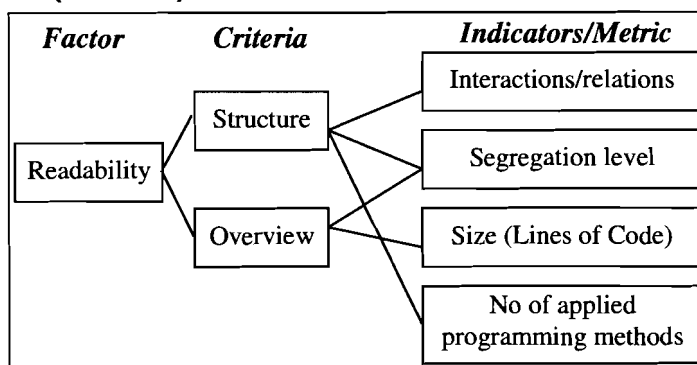
In this thesis only “*quality control*” and “*product engineering*” are fields of interest. Metrics that contribute to these fields of interest should be selected as metrics for software complexity.

Quantitative physical software attributes are metrics, which means that they are precise and can be measured directly. There are over 200 metrics developed over time (Zuse, 1990) and many of them are also derived from each other. Some basic metrics that are frequently applied are (McQuaid, 1996): no. of bytes, no. of lines of code (LOC), Halstead Fundamental Attributes, McCabe Fundamental Attributes (cyclomatic complexity), and depth of nesting.

All metrics can generally be classified in the following categories (McQuaid, 1996): Software science parameters, Control-flow metrics, Data-flow metrics, Information-flow metrics, Hybrid metrics, Special-purpose metrics. These categories and related software complexity metrics are described in appendix 8.

### 5.1.2 Qualitative software attributes (metrics)

Quality software attributes are software properties like; *reliability, maintainability, readability, testability or understandability* (McQuaid, 1996; Lange, 2005). These properties are in general factors that cannot be measured directly and therefore are expressed in related direct measurable indicators. For example: readability in Figure 19.



The principle behind these factors is that this taxonomy of indicators will provide

**Figure 19 Example readability [Source: Fenton, 1996]**

a general picture of the readability of the software program. Within this master thesis the factor is “software complexity” and should be expressed in relevant criteria and accompanying metrics. This qualitative software complexity model will be defined in the following paragraph.

### 5.1.3 Software complexity in embedded system software

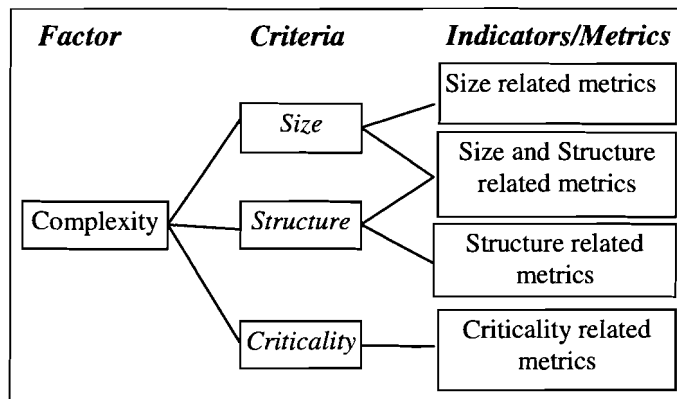
In order to define software complexity for a complex repairable system like a cardio vascular X-ray scanner, the software “should be” subdivided into system operating software and embedded system software. Embedded software is software that directs the hardware components and therefore is a subpart of a larger host (Coskun & Grabowski, 2005).

However, the discussion whether a software unit is a general system operating software or embedded software is not that important. The defined model by Coskun and Grabowski (2005) to define software complexity for embedded intelligent real-time systems could also partly be applied in order to define “software complexity” for the Allura Xper systems as well. Table 5 provides an overview of the software complexity types and corresponding related domains and metrics. This master thesis will restrict itself to the “code structure” complexity, which is within the impact area “software performance”. Reason for this restriction is the direct relation with the internal product characteristics (software performance) and not interference with other impact areas like operator performance as is shown in Table 5. This choice results into analysis of code structure related software complexity metrics like; LOC, Software science metrics, and cyclomatic complexity.

Software complexity type	Impact area	Related domains	Metrics and measurements
Code structure complexity	<ul style="list-style-type: none"> <li>Software performance</li> </ul>	<ul style="list-style-type: none"> <li>Mathematics</li> <li>Economics</li> <li>Software engineering</li> <li>Computer science</li> </ul>	<ul style="list-style-type: none"> <li>Source lines of code</li> <li>Software science metrics</li> <li>Cyclomatic complexity</li> <li>Language and design specific complexity metrics</li> </ul>
Data processing and reasoning complexity	<ul style="list-style-type: none"> <li>Software performance</li> <li>Operator performance</li> </ul>	<ul style="list-style-type: none"> <li>Decision support systems</li> <li>Intelligent systems</li> <li>Psychology and cognitive science</li> <li>System science</li> <li>Computer science/software engineering</li> <li>Control systems</li> </ul>	<ul style="list-style-type: none"> <li>Task level metrics</li> <li>Representation level metrics</li> <li>Implementation level metrics</li> </ul>
Decision support and explanation complexity	<ul style="list-style-type: none"> <li>Operator performance</li> </ul>	<ul style="list-style-type: none"> <li>Decision support systems</li> <li>Computer-aided decision-making</li> <li>Cognitive sciences</li> <li>Psychology</li> <li>Human-computer interaction</li> </ul>	<ul style="list-style-type: none"> <li>Quality and usability of decision support</li> <li>User perception of decision support provided</li> </ul>
User interface complexity of information on user interface	<ul style="list-style-type: none"> <li>Operator performance</li> </ul>	<ul style="list-style-type: none"> <li>Computer science</li> <li>Human-computer interaction</li> <li>Software engineering</li> <li>Psychology</li> <li>Cognitive sciences</li> <li>Human factors</li> </ul>	<ul style="list-style-type: none"> <li>Amount</li> <li>Quality of user interface</li> <li>Usability of user interface</li> <li>Understandability of user interface</li> </ul>

**Table 5 EIRTS complexity; domains, metrics and measurements [Coskun and Grabowski, 2005]**

However, this factor “code structure complexity” needs to be expressed in criteria and related metrics if you want to quantify this factor (Fenton, 1996) in a complexity index value. Based on literature review (Lew, *et al.*, 1988; Khoshgoftaar and Munson, 1990(A); Khoshgoftaar and Munson., 1990(B); 1991; Yin, *et al.*, 2004), the following main criteria for software complexity returned frequently in other researches: Size, Structure and Criticality of the software program. Figure 20 shows the composition of complexity and its criteria and metrics. The actual selection of metrics for the 3 code complexity criteria will follow in paragraph 5.2.



**Figure 20 Code Complexity Model**

## 5.2 Evaluation and selection of software complexity metrics

The best metrics to express software complexity depends on some criteria. First of all the purpose for which this metric is used has major impact on which metrics are best representing software complexity. For this master thesis, in which the relationship between software complexity and software reliability is investigated, the metrics should be logically linked to software reliability. Much research on the relation between software metrics and the occurrence of software failures has been done (Lew, *et al.*, 1988; Khoshgoftaar and Munson, 1990(A); Khoshgoftaar and Munson., 1990(B); 1991; Yin, *et al.*, 2004). Paragraph 4.4 describes the expected relations between metrics and reliability and also discusses how to measure this in practice for the CV X-ray systems at PMS.

A second criterion that determines which software metrics to apply is the available data at PMS that is required to measure these software metrics. This is an essential criterion, while generally you will not be able to find all the required data to measure all these metrics mentioned in the paragraphs before. Also the required time to gain the required data differs among the metrics. For example the required data for McCabe’s metric is much harder to gain than the Halstead science parameters or Lines of Code.

These two criteria should be taken into account when software metrics are selected. For example Lines of Code is a simple metric which is easy to access and it’s relationship with software reliability has already been proved in the past frequently (Munson and Khoshgoftaar, 1991; Yin *et al.*, 2004; Coskun and Grabowski, 2005). These researchers mention that Lines of Code has a significant **correlation** with the number of software errors, but they also mention that this metric **cannot** be used easily for **predicting** the number of software errors.

The aim of this thesis is to **explain** the occurrence of software failures and its variance among systems and provide **focus** for decision rules for product improvement. When feasible, a model/tool that can **predict** the software reliability performance based on the software code metrics scores will be developed. For each of these aims, different complexity data is required as described below. These requirements form the criteria to select code metrics.

### Research aims and requirements for software complexity code metrics:

**Aim:** *Explain the software reliability performance based on complexity metrics*

#### **Requirements:**

- There should be correlations between some of the software metrics and the software reliability performance.
- At least some differences in the metrics scores among the software versions, SPs, software subsystems or units should exist in order to establish a relationship between both variables “complexity” and “reliability”.

- Also within the reliability performance some differences in the scores among software versions, SPs, software subsystems or units are required. It can be stated that there is variance within the SW reliability performance as this was proven in paragraph 4.5.

**Aim:** *Creating a regression model for software units or subsystems that predicts their reliability performance based on the known software metrics scores that are included in the model.*

**Requirements:**

- All mentioned requirements for the previous aim.
- The minimum ratio of observations (of software versions, SPs, subsystems and units) and variables should at least be 5:1. This means that a regression model that includes 2 metrics should at least require 10 observations. However, for a single metric (simple regression) at least 20 observations are required. (Hair, *et al.*, 2006) Unfortunately, there are only 12 SW subsystems and around 40 SW units within a Software version and for the Allura Xper systems less than 5 SPs or Software versions do exist. Furthermore, not for every single SW version, SP, subsystem or unit is data available which will even further decrease the number of observations. Therefore, designing powerful regression models will be difficult.

**Aim:** *Gain focus for design decision rules based on “historic information about the relation between software code metrics and reliability”.*

**Requirements:**

- All mentioned requirements for the previous aims.
- The correlations between the SW metrics and reliability should represent causal instead of coincidental relationships.
- The designer should be able to manipulate these metrics. (Example; you can't just delete code in order to reduce the LOC, because the program won't perform its functionalities anymore.)

**Aim:** *Predicting software reliability based on “historic information on the relation between the metrics and reliability” and “the current software metric scores of the new developed Software”.*

**Requirements:**

- All mentioned requirements for the previous aims.
- Reliability measures and software code metrics for more than one SW version or SP.
- No radical changes should occur in the underlying relationships between the SW code metrics and the occurrence of SW failures or errors. This requirement is impossible to satisfy because one can only predict the future but almost never know it for sure.

The next paragraph describes the available data and concludes for which aims all requirements can be fulfilled with this available data set.

### 5.3 Data sources for software complexity metrics

The available SW code metrics differ for each layer within the SW architecture. This SW architecture and its division in subsystems and units is described in appendix 1. However, not all subsystems and/or units are taken into account. The reasoning behind the exclusion of the following subsystems and units is:

- Excluded subsystems: The subsystems *Infra*, *VwWorks*, *OS*, *FSFW* are excluded because they are within the infrastructural layer and contain generally standardized SW applications that have been purchased. Subsystem *Host* is excluded because similar reasons and subsystems *Geometry* and *Positioning* are combined into one subsystem.
- Excluded units: All the *Field Service Code* units (FSC) are excluded because they are code files for the engineers themselves and not for running the software, this code is generally written fast and unstructured and doesn't represent the general standard of coding within PMS. Furthermore, the units: *FS UIspecific* and *Storage Hardcopy* are excluded because they did not contain any C++ source code.

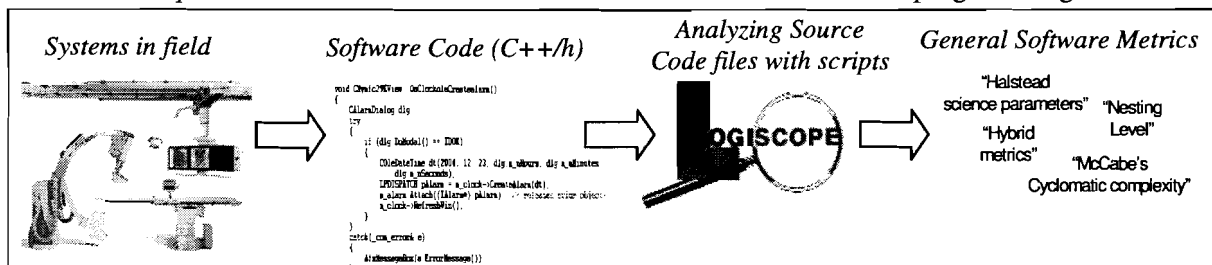
There were two main sources for collecting SW complexity metrics. The first source is an audit of the system source code by the Telelogic Logiscope tool and the second source is an audit of the source

code by the CV Software development team within PMS, who process data from Clear Case into a set of software metrics.

### 5.3.1 Data source: Telelogic Logiscope audit

The Logiscope tool from Telelogic can audit source code files with a set of scripts that calculate some standard SW code metrics. Figure 21 visualizes how the SW code metrics were gained by the Logiscope Tool.

First, the **Source Code** files are loaded in the program and the **Data file** is built. During this building process, Logiscope attempts to read and understand all the source code (C++ code and Header files), therefore it requires a **dialect**. This dialect can read and understand the code programming standards.



**Figure 21 Data collection and transformation into information**

In this Master thesis the **dialect** “*Rocket\_logiscopeFilter*” is applied, this filter has been designed for the CV Software application Xtravision. Because there is a general programming methodology (TICS) within PMS, this dialect will be able to read more than the general C++ dialect standard provided by Logiscope. However, not all programming code can be standardized and after the “source file” has been built with the dialect “*Rocket\_logiscopeFilter*” some errors remain due to inability of the dialect to understand this code. The source code files that contain errors are excluded entirely from the analysis.

It is assumed that the proportions of excluded source code files are equal for each software subsystem / unit. This assumption has been made because it was too much work to include all the errors as exceptions in the dialect “*Rocket\_logiscopeFilter*” or measure how many errors or types of errors occur.

Secondly, the **data file** is analyzed by programmed scripts that can calculate a specific metric. These scripts are combined in one “**Quality model**” that can be selected in order to audit the built source file. Two available standard Quality models provided by Telelogic (Logiscope) are used because of the inability of the researcher to program software scripts in C++ code. These two Quality models were “*Logiscope.ref*” and “*Halstead.ref*”.

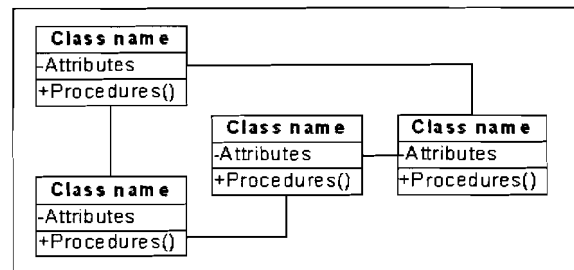
Table 6 shows for each quality model which metrics are measured. Definitions of the metrics (descriptions / interpretation) can be found in appendix 9.

In the past the Logiscope audit was performed on the Xtravision software (an additional software application for Allura Xper). This audit concluded that most metrics were not able to grasp the difference between the software versions for the Xtravision program. This master thesis will verify if the Logiscope audit can be successful within other software subsystems or units and if it can grasp the changes applied in new software versions and Service Packs. Another conclusion from the Xtravision audit was that the “Fan in – Fan out” (FI/FO) metric was one of the few metrics that was able to grasp the development changes. Also Kitchenham and Linkman (1990) mention that the FI/FO (appendix 8) is an important metric with significant impact on reliability. Unfortunately this metric was not available in the quality models “*Logiscope.ref*” and “*Halstead.ref*”. However, it is highly **recommended** to measure this metric and gain the required quality model from Telelogic.

"Logiscope.ref" (Class metrics)	"Logiscope.ref" (Application metrics)	"Halstead.ref" (Class metrics)
Number of statements	Depth of inheritance tree**	Total number of operands
Comment frequency	Average complexity of functions**	Number of distinct operands
Average size of statements	Call graph depth	Total number of operators
Cyclomatic number (VG)	Sum of cyclomatic numbers of application functions	Number of distinct operators
Maximum nesting level	Number of lines	Halstead intelligent content
Vocabulary frequency	Number of application functions	Halstead mental effort
Number of paths	Coupling factor**	Halstead program level
Number of callers	Polymorphism factor**	Halstead difficulty
Structuring	Attribute inheritance factor**	Halstead volume
Number of direct calls	Method inheritance factor**	Halstead estimated length
Number of parameters	Attribute hiding factor**	Halstead length
Number of distinct operands	Method hiding factor**	Halstead vocabulary
** MOOD = Metrics for Object Oriented Design		

**Table 6 Available metrics with Logiscope and Halstead Quality Models in Logiscope**

The metrics within the Quality model "Logiscope.ref" can be divided into **Class metrics** and **Application metrics**. Each SW unit consists out of some **application functions** which again consist out of a set of classes. Figure 22 visualizes the building blocks of an application function within a SW unit.



**Figure 22 Example: Application function**

Furthermore, the \*\* marked metrics (Table 6) are rather design than code metrics and therefore specifically useful for Object Oriented (OO) designed Software (McQuaid, 1996). OO software focuses more on modeling the real world and the software system in terms of its objects, in contrast to the traditional approach which was function-oriented and separated the data and procedures. Because of this different design approach, design related metrics should be applied (Chidamber and Kemere, 1994). The source code for the Allura Xper systems is generally written in C++ language with an OO design in mind (communications with Oborzynski, Xtravision software developer at PMS). These Metrics for Object Oriented Design (MOOD) metrics are therefore very important metrics to be measured.

### 5.3.2 Data source: Audit by CV software development team

The second main source for gaining software code metrics is the processed data from the "Clear Case" database which is represented on the PMS intranet site "Just Another Homepage" (JAH). The Clear Case database contains all source code for all software versions and accompanying documentation. The CV software development team processes this raw data into general statistics (SW code metrics) by running scripts on the source code files and places them on (JAH) for management and developers. The following SW metrics on subsystem and unit level for each Software Version and accompanying SPs are available at JAH:

- Number of lines of code (LOC)
  - Total lines of code
  - Added lines of code
  - Modified Old: Modified lines in old base line
  - Modified New: Modified lines in new base line
  - Deleted lines of code
  - Same: Unchanged lines of code
- Number of empty lines of code
- Number of comment lines of code

- Number of files
- Number of directories
- Mondriaan: number of interactions between interfaces of subsystems
- Number of improvement changes / problem solutions
- New added or extended functionalities

These are all relative simplistic size metrics except for the “Mondriaan” metric.

The software code metric “number of lines of code” is divided into some sub-metrics, the division tries to identify how **radical** the new software version or SP has been **changed** compared with its predecessor. This innovation ratio is important in order to compare metrics scores between *software versions* and *SPs*, because **incremental small changes won’t change the general structure** of the software code, the metrics will only measure marginal structure changes or stay constant. Figure 23 compares two consecutive SPs and shows all sub-metrics of LOC. It also shows that the majority of the software code remains the same, which means that the general structure or characteristics of the software probably won’t change for the new SP.

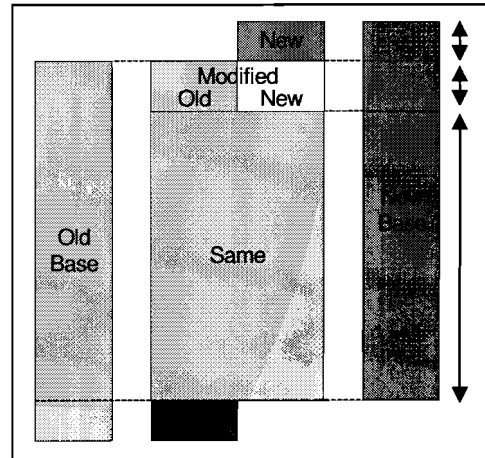


Figure 23 Comparing old base with new base (derived from; JAH intranet site PMS)

Table 7 shows all PBLs (PBL20, PBL40, and PBL60), Software versions and SPs for the Allura Xper FD20 and FD20/10 product line. There are two major ways in which the software is improved, these are: “**Improvement by adding new / extended functionality**” and “**Improvement by solving problems (Software bugs)**”. The horizontal arrow shows the development in SPs which represents the improvement by *solving problems*. The vertical arrow shows the development in *new PBLs* (complete new software version), which means *added functionality* to the previous PBL.

PBL/ Project	Commercial Name	Software: New Version	Service Packs				
			SP-1	SP-2	SP-3	SP-4	SP-5
PBL_23/ Rocket-A	Allura Xper FD20	PBL_2.0.0/ PDC_23.25.0	PBL_2.0.1/ PDC_23.25.1	PBL_2.0.2/ PDC_23.25.2	PBL_2.0.3/ PDC_23.25.3	PBL_2.0.4/ PDC_23.25.47	PBL_2.0.5/ PDC_23.25.53
PBL_41/ Rocket-B2	Allura Xper FD20 Allura Xper FD20/10	PBL_4.0.0/ PDC_41.12.0	PBL_4.0.1/ PDC_41.13.4	-	-	-	-
PBL_43/ Rocket-B2+	Allura Xper FD20 Allura Xper FD20/10	PBL_4.3.0/ PDC_43.10.0	PBL_4.3.1/ PDC_43.10.14	PBL_4.3.2/ PDC_43.10.22	PBL_4.3.3/ PDC_43.10.39	PBL_4.3.4/ PDC_43.10.42	-
PBL_61/ Rocket-C2	Allura Xper FD20 Allura Xper FD20/10	PBL_6.0.0/ PDC_61.18.0	PBL_6.0.1/ PDC_61.18.12	-	-	-	-

Table 7 Overview of Software versions and SPs PBL40’s (JAH intranet site PMS)

In the case of “*bug solving*” only minor changes in the software occur. After measuring the Software Code metrics by Logiscope for all the SPs of the PBL43 subsystem Acquisition (a major and critical subsystem), hardly any changes in the metrics scores could be found. (Appendix 10) It can be concluded that the general code structure metrics from Logiscope cannot be used in order to explain the changes between the SPs.

The JAH metrics are now more useful, because they register how many lines of code are **added** or **modified** in order to create this new functionality. Besides, the number of **improvement changes / problem solutions** is registered for each new SP. However, the question remains if these minor changes in the product characteristics should be considered as an increased system complexity. From the other side, it is not illogical to think that these changes in reliability between the SPs might be explained by these minor changes in the product characteristics. The next paragraph will elaborate on this issue in more detail.

## 5.4 Findings and conclusions on available complexity data

The findings and conclusions about the data provided by the two data sources are described in paragraph 5.4.1 and paragraph 5.4.2.

### 5.4.1 Available code metrics for the code complexity model

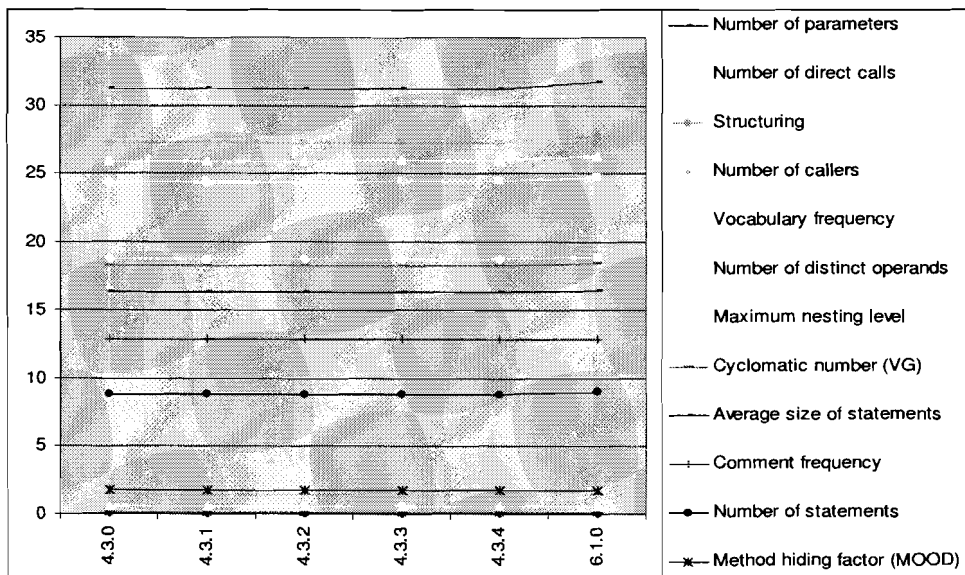
Not all software code metrics are available for all hierarchical system levels (paragraphs 5.3.1 and 5.3.2). Now the available and applicable metrics for the “code complexity” model will be defined.

As the Logiscope audit takes much time, because loading the data into the Logiscope program is time consuming; only one PBL is selected for the Logiscope audit. This is the Allura Xper FD20 / FD20/10 (PBL41 Rocket B2 and PBL43 Rocket B2+), this PBL40 is selected because this system has also been selected for the Reliability performance as described in paragraph 4.5.

The Logiscope and JAH metrics are available on the hierarchic levels “subsystem” and “unit” for all Software versions and SPs of the Allura Xper FD20 and FD20/10 series. On system level only the Mondriaan metric is available, therefore we restrict the research to subsystem and unit level only (Table 8). Also the Service Packs (4.3.0 - 4.3.4 and 6.1.0) are not taken into account, because the changes between the Software Versions and the SPs were negligible (see Figure 24).

Level:	Available Software Metrics:
System	Mondriaan metric
Subsystem	Logiscope metrics and JAH metrics excl. the Mondriaan metric
Unit	Logiscope metrics and JAH metrics excl. the Mondriaan metric

**Table 8 Available Software metrics at PMS**



**Figure 24 Changes in some code metrics among Software versions and Service Packs**

An overview of the metric scores gained by the audit with Logiscope and the data from the JAH intranet site are represented for each Software subsystems and unit in appendix 11.

Now the available code metrics are known, the complexity factor diagram from paragraph 5.1.3 can be finished by coupling the metrics to each specific criteria of software complexity based on the descriptions of the metrics in appendix 8 and appendix 9. Table 9 shows all the metrics mapped with the complexity criteria from the “code complexity” factor diagram. The cursive marked metrics are from the JAH intranet site and the other metrics are from the Logiscope audit. The correlations between the metrics and the criteria are investigated by a factor analysis in paragraph 5.4.2.



Factor	Criteria	Metrics			
<b>Complexity</b>	<b>Size</b>	1. Number of statements	10. Halstead volume		
		2. Comment frequency	11. Halstead estimated length		
		3. Number of parameters	12. Halstead length		
		4. Sum of Cyclomatic Numbers	13. <i>LOC sub-metrics</i>		
		5. Number of lines	14. <i>Empty lines of code</i>		
		6. Number of application functions	15. <i>Comment lines of code</i>		
		7. Number of classes	16. <i>Number of files</i>		
		8. Total number of operands	17. <i>Number of directories</i>		
		9. Total number of operators	18. <i>Additions and changes</i>		
		<b>Structure</b>	<b>Structure</b>	1. Average size of statements	12. Attribute inheritance factor**
				2. Cyclomatic number (VG)	13. Attribute hiding factor**
3. Maximum nesting level	14. Method hiding factor**				
4. Number of paths	15. Number of distinct operands				
5. Structuring	16. Number of distinct operators				
6. Number of direct calls	17. Halstead intelligent content				
7. Number of parameters	18. Halstead mental effort				
8. Depth of inheritance tree**	19. Halstead program level				
9. Average complexity of functions**	20. Halstead difficulty				
10. Coupling factor**	21. Halstead vocabulary				
11. Polymorphism factor**	22. Call graph depth				
<b>Criticality</b>	<b>Criticality</b>	1. Number of callers	2. Number of direct calls		

\*\* Metrics for Object-Oriented Design  
*cursive marked metrics* are from JAH others are from Logiscope

**Table 9 Complexity criteria and related metrics**

### 5.4.2 Reducing the number of software complexity metrics

There are too many metrics in order to create a regression model, especially because there are only 17 observations for which all metrics are available. Purpose of the factor analysis is to reduce the number of metrics according to the three criteria mentioned in Table 9. Therefore the principle components analysis has been carried out for the 40 software units for which the metrics were known. (Hair, 2006) The factor analysis searches for linear relationships exhibited in the data. When variables are high correlated with each other they share the same variance, the more the variance can be explained by both variables the higher the factor loadings. The first factor found is the factor that can be seen as the single best summary of linear relationships in the data. The second factor is than the second best summary of all linear relationships in the data and so on. (Hair, 2006) In order to apply this principle components analysis, some conditions should be met. The sample size should be at least 50 observations and for each variable that is taken into account at least 5 observations should be available. There are around 40 complexity metrics and only 41 observations. In order to get a more reliable analysis and higher factor loadings, some complexity metrics should be excluded from the research.

The number of complexity metrics is reduced by selecting only the software complexity metrics that show some correlation with the reliability metrics: “field programming errors” or “problem reports”. Therefore, a correlation test among the complexity metrics was performed. (Appendix 12) The result of the extraction of metrics by the correlation with the reliability metrics resulted in the following selection:

- Logiscope: Call graph depth, Depth of inheritance tree, Number of application functions, Number of lines, Sum of cyclomatic numbers of the application functions, Average size of the statements, Cyclomatic complexity (this metric did not have high correlation with any of the reliability

metrics, but it is a well accepted software complexity metric and therefore not excluded for the factor analysis), Maximum nesting level, Number of callers, Number of paths, Number of statements, Halstead intelligent content, Halstead mental effort

- JAH: Total no of lines, No of empty lines, No of comment lines, No of files, No of directories

Now only 18 metrics are left of the 40 initial metrics, and all three criteria for the code complexity factor are represented in the list by one or more metric(s). Now the principle component analysis is carried out with SPS15.0. The first factor analysis resulted into 3 factors and also some metrics that had multiple loadings on 2 or 3 factors or even didn't had any factor loadings at all. The worst metrics were deleted one by one and every time the principle components analysis was repeated until the component matrix in Table 10 was found.

Metric:	Component	
	1	2
CSI2_NumberofApplicationFunctions	0.956	
CSI_JAH3_CommentLines	0.955	
CSI4_SumofCyclomaticNoofApplicationFunctions	0.937	
CSI3_NumberofLines	0.928	
CSI_JAH4_Files	0.881	
CSI_JAH5_Directories	0.778	
CST10_MaximumNestingLevel		0.879
CST9_CyclomaticNumberVG		0.835
CST8_AverageSizeofStatements		0.832
CCR1_NumberofCallers		0.813
CST17_HalsteadMentalEffort		0.796

*Extraction Method: Principal Component Analysis.*

**Table 10 Component matrix after Principal Component**

As you can see this component matrix includes only 2 factors and 11 metrics, 11 metrics is still quiet a lot for 41 observations, because the minimal sample size should at least be 50 observations. However, the factor loadings are all pretty high, 0.75 or higher, and the factors seem logical too. Besides, the Bartlett's test of sphericity in SPS15.0 was significant and also the Kaiser-Meyer-Olkin measure of sampling adequacy (0.617) was suitable.

Besides, the component score covariance matrix also showed that the two factors were independent from each other (Table 11).

Component	Size	Structure
Size	1,000	,000
Structure	,000	1,000

*Extraction Method: Principal Component Analysis*

**Table 11 Component score covariance matrix in SPS15.0**

In order to validate if the categorization of the metrics into two factors is influenced by underlying distributions in the code metrics scores among software units, the sample of 41 systems was divided into two groups. The principle component analysis was repeated and resulted in the two component matrixes in Table 12 and component plots in Figure 26. Both show that the factors and corresponding loadings for each code metrics remain almost the same.

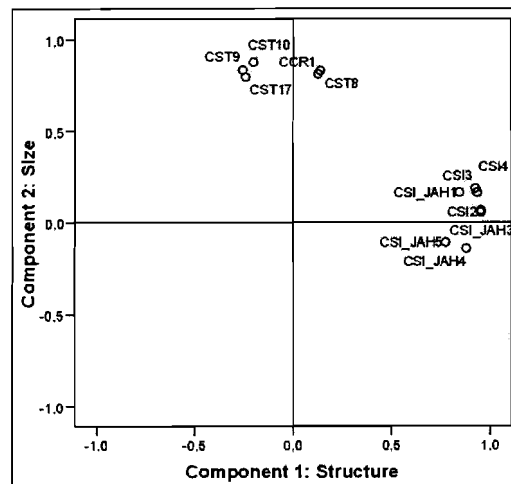
Component Matrix sample 1			Component Matrix sample 2		
	Component			Component	
Metrics:	1	2	Metrics:	1	2
CSI2_NumberofApplicationFunctions	0,973		CSI_JAH3_CommentLines	0,953	
CSI_JAH3_CommentLines	0,943		CSI2_NumberofApplicationFunctions	0,940	
CSI_JAH1_TotalLines	0,916		CSI4_SumofCyclomaticNoofApplicationFunctions	0,934	
CSI4_SumofCyclomaticNoofApplicationFunctions	0,908		CSI3_NumberofLines	0,925	
CSI3_NumberofLines	0,897		CSI_JAH4_Files	0,910	
CSI_JAH4_Files	0,757		CSI_JAH1_TotalLines	0,840	
CSI_JAH5_Directories	0,656		CSI_JAH5_Directories	0,827	
CST10_MaximumNestingLevel		0,938	CST9_CyclomaticNumberVG		0,866
CCR1_NumberofCallers		0,917	CST10_MaximumNestingLevel		0,827
CST8_AverageSizeofStatements		0,898	CST17_HalsteadMentalEffort	-0,407	0,790
CST9_CyclomaticNumberVG		0,880	CST8_AverageSizeofStatements	0,467	0,543
CST17_HalsteadMentalEffort		0,802	CCR1_NumberofCallers		0,501

*Extraction Method: Principal Component Analysis.*

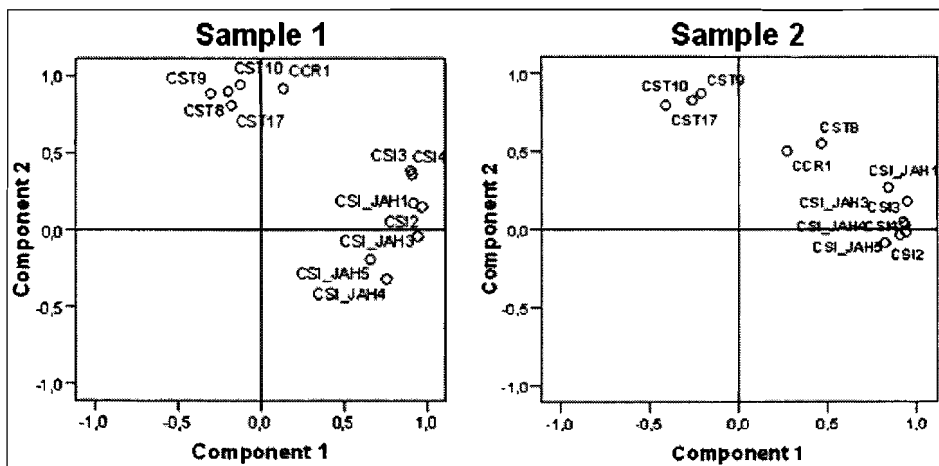
**Table 12 Component matrix for both control groups**

The two components plotted in Figure 25 show that there are two major factors, these factors were also found in Figure 26 where the component plots are shown for the two control groups.

When analyzing the metrics within each component, it seems that components can be sub-divided into “size” and “structure” related metrics. Only the metric “number of callers” (indicates the frequency with which the application function is called by other functions) seems to correlate with the component “structure” as well. Due to the fact that this metric also correlates with reliability, it is not excluded from the analysis and will be an indicator for the component “structure”.



**Figure 25 Component plot with factor loadings for each code complexity metric**



**Figure 26 Component plots for both control groups**

These metrics should now be aggregated into composite factor scores. There are several ways to aggregate the metrics into a summated scale, which are (Hair, 2006):

- Take the average score of all the metrics that are within the component.
- Different weights for each metric, the next methods are available in SPSS15.0 (Field, 2005):
  - Regression method: The scores that are produced have a mean of 0 and a variance equal to the squared multiple correlation between the estimated factor scores and the true factor values. The scores may be correlated even when factors are orthogonal. (SPSS15.0)

- Barlett scores: The scores that are produced have a mean of 0. The sum of squares of the unique factors over the range of variables is minimized. (SPSS15.0)
- Anderson-Rubin Method: a modification of the Bartlett method which ensures orthogonality of the estimated factors. The scores that are produced have a mean of 0, have a standard deviation of 1, and are uncorrelated. (SPSS15.0)

First of all, a factor score is a composite score for each observation on a particular factor. The Barlett method ensures composite scores that correlate only with their own factor. However, the factor scores can still correlate with each other. The Anderson-Rubin method, a modification of the Barlett method, ensures that the factor scores are uncorrelated and standardized with a mean of 0 and a standard deviation of 1. (Field, 2005) The objective in this research is to reduce the number of variables into a few factors which can be used in order to develop a single or multiple regression model. Multicollinearity among the predicting variables can negatively affect the regression model, therefore uncorrelated factor scores are desired in order to prevent the occurrence of multicollinearity among the predicting variables (Field, 2005). The **Anderson-Rubin method is selected** in this research in order to create the factor scores.

## 5.5 Validation of the used audit programs Logiscope and JAH

It is hard to validate if the used complexity code metrics are well designed and give correct indications for the actual code complexity. In order to come up with some sort of validation, some of the metrics are compared between the different data sources (Logiscope and JAH) and the audit by the TIOBE Coding Standard (TICS) analyzer. The TICS analyzer checks if the code is developed according to the general programming guidelines defined by PMS in order to get more standardized code among different software developers. There is only a limited number of complexity metrics available in the TICS analyzer; these are Lines of Code (LOC) and Cyclomatic complexity (VG).

The comparison for the metric LOC between Logiscope, JAH and TICS is shown in Figure 27. Among the three sources you can see for the metric LOC that these differ between the sources. This difference arises from the different scripts that are used by Logiscope and JAH in order to audit the source code. However, the differences in the measures do appear to be consistent among the software units. It seems that in general the LOC scores from JAH are larger than from Logiscope, this can be explained by the dialect of both audits, while the JAH dialect is much more complete than the Logiscope dialect. Therefore the JAH audit can read more files with PMS specific programming language which are excluded by the Logiscope dialect because it doesn't understand this PMS specific programming code.

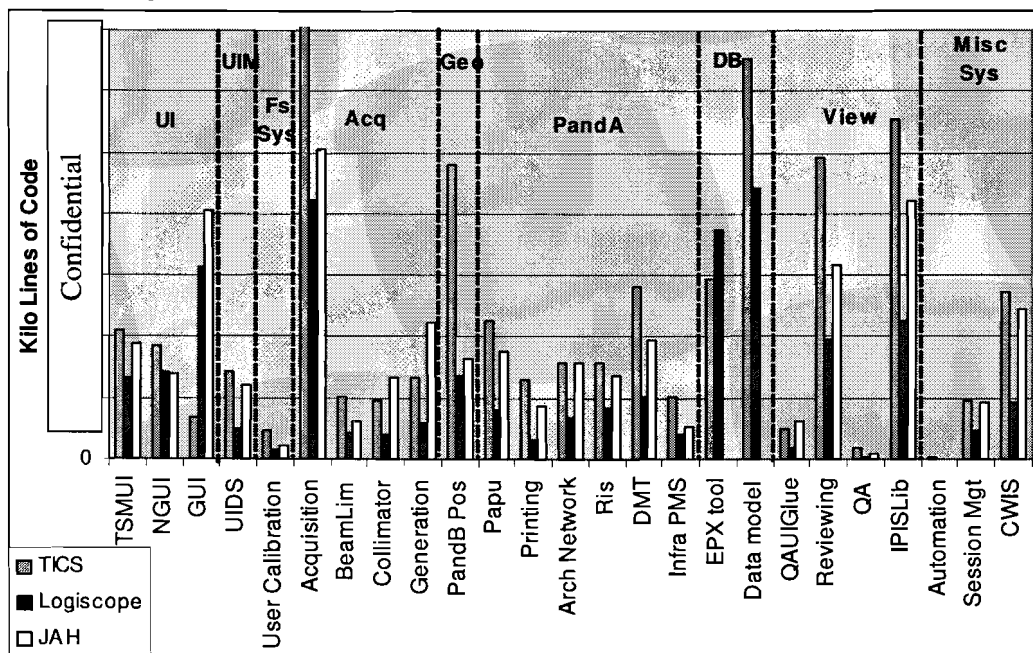


Figure 27 Comparing LOC between Logiscope and JAH

The exclusion of some parts of the source code by Logiscope doesn't seem to affect the correctness of interpreting the size of the software units in LOC. The Kendall's tau correlation test (Table 13) shows that the three measures of LOC have high and significant correlation among each other. This fact supports the conclusion that the metrics retrieved from Logiscope and JAH give representative metric scores.

**Kendall's tau Correlations**

<i>Lines of Code (LOC)</i>		TICS	Logiscope	JAH
TICS	Corr. Coefficient	1,000	,740(**)	,684(**)
	Sig. (2-tailed)	.	0,000	0,000
	N	25	25	23
Logiscope	Corr. Coefficient	,740(**)	1,000	,771(**)
	Sig. (2-tailed)	0,000	.	0,000
	N	25	25	23
JAH	Corr. Coefficient	,684(**)	,771(**)	1,000
	Sig. (2-tailed)	0,000	0,000	.
	N	23	23	23

\*\* . Correlation is significant at the 0.01 level (2-tailed).

**Table 13 High correlations among LOC scores for: TICS, Logiscope and JAH**

Besides LOC, the metric cyclomatic complexity (VG) was available in TICS and Logiscope. The same correlation test between TICS and Logiscope's VG is performed and shown in Table 14.

**Kendall's tau Correlation test**

<i>Cyclomatic Complexity (VG)</i>		TICS	Logiscope
TICS	Corr. Coefficient	1,000	,412(**)
	Sig. (2-tailed)	.	0,004
	N	25	25
Logiscope	Corr. Coefficient	,412(**)	1,000
	Sig. (2-tailed)	0,004	.
	N	25	25

\*\* . Correlation is significant at the 0.01 level (2-tailed).

**Table 14 High correlations among VG scores for: TICS, Logiscope and JAH**

Based on these validations by multiple data sources for the complexity metrics LOC (size related) and VG (structure related), this research concludes that the metrics measured by Logiscope and JAH do represent the actual code complexity well.

## 6. Software complexity versus software reliability

This chapter starts to describe the expected relationships between SW complexity SW and reliability, based on findings from other researchers (paragraph 6.1).

Subsequently, these general relationships are tested by correlation tests on the metrics for reliability and complexity from chapters 4 and 5 (paragraph 6.2). These correlation tests give answer to the question if SW reliability (in terms of PRs, FPEs and SW failures) is affected by SW complexity. And if so; which “complexity code” metrics do correlate with which reliability metrics? This paragraph provides **insight** in the relations between SW complexity and reliability.

Paragraph 6.3 constructs several regression models in order to **explain** the variance in the reliability metrics by the complexity factors “size” and “structure” (paragraph 6.3.1) and some others to **predict** the FPEs or SW failures for a SW unit based on its complexity scores (paragraph 6.3.2).

Paragraph 6.4 investigates if a categorization of the SW units into groups with similar “complexity” scores will reduce the variance in the reliability metric “SW failures” within each complexity class. In this case, the SW complexity can **explain** the differences in the reliability performance among SW units. Finally, paragraph 6.5 summarizes the results and findings in this chapter.

### 6.1 Expected relation between SW complexity and reliability

Based on literature (Lew, *et al.*, 1988; Khoshgoftaar and Munson, 1990(B), 1991; Yin, *et al.*, 2004) it is generally assumed that an increase in complexity will have a negative impact on the reliability performance. However, this increase in complexity doesn’t unconditionally correspond with an increase in the software design and code metrics. Because in some cases there will exist negative correlation. The code complexity model in paragraph 4.2.3 shows the complexity criteria and the related metrics which are described in Table 9.

It is assumed that the software complexity criteria (and its metrics) have different impact on reliability. The *Size and Structure* metrics are assumed to have a **positive correlation** with the reliability metrics (Field programming errors or Test problem reports), because the larger the program and the more complex it structure the lower the reliability performance. In contrast, the *Criticality* metrics are assumed to have a **negative correlation** with the reliability metrics, the more critical a component (e.g. the more frequent the component is used/called) the more testing is applied and the lower the reliability metrics scores (programming errors and problem reports) will be.

However, there can also be **exceptions** to these general assumed relationships. For example, some researchers like Mc Cabe who developed the metric “cyclomatic complexity” (McQuaid, 1996), argue that a large sized software program doesn’t necessary influence the SW reliability performance in a negative way. This opinion is shared by software developers at PMS (communications with J. Boot; metrics officer at PMS). They argue that large sized software may include more standard code or repeated code which positively influences the software reliability.

This argument should be captured by the factor “**structure**” within the complexity factor diagram; a large sized software code should correlate with low scores on its structure metrics or the compiled structure factor scores. However, this argument will be rejected in paragraph 6.3

### 6.2 Correlations with reliability metrics

The following reliability metrics and complexity metrics/factors are available for investigating the relation between SW reliability and SW complexity:

- **Reliability metrics:** test problem reports, field programming errors, error types, entropy (based on programming errors), and software failures (derived by mapping programming errors with FMEA failure causes).
- **Complexity metrics:** subdivided into the factors:
  - *Size:* no. of application functions, no of lines of code, sum of cyclomatic complexities of all application functions (Logiscope), and no. of code and comment lines, no. of files, no. of directories (JAH).
  - *Structure:* cyclomatic complexity, average size of statements, maximum nesting level, Halstead mental effort, number of callers (Logiscope)

The complexity metrics are available for all SW units of the PBL40 Allura Xper systems. The reliability metrics “PR and FPE” are only available for 17 SW units and the reliability metric “software failure” (SWF) is available for 19 SW units. These data sets are available in appendix 12 and appendix 13.

In order to get a better understanding of the underlying relations between the reliability metrics and the complexity metrics and factors, independent “bivariate” correlations between all metrics are performed in SPSS15.0 (Table 15).

Kendall's tau Correlation test															
		CS2 Number of Application Functions	CS3 Number of Lines of code (Logiscope)	CS4 Sum of Cyclomatic No of Application Functions	CST8 Average Size of Statements	CST9 Cyclomatic Complexity (WG)	CST10 Maximum Nesting Level	CST17 Halstead Mental Effort	CCR1 Number of Callers	CSL_JAH1 Total no of Lines of code	CSL_JAH3 No of Comment Lines	CSL_JAH4 No of Files	CSL_JAH5 No of Directories	Factor 1 Size	Factor 2 Structure
Test Problem Reports	Corr. Coefficient	.424(*)	.424(*)	.446(*)	0.042	-0.075	-0.107	.403(*)	-0.106	0.212	0.191	0.266	0.312	<b>0.382</b>	<b>0.106</b>
	Sig. (2-tailed)	0.037	0.037	0.028	0.835	0.715	0.602	0.047	0.602	0.297	0.348	0.192	0.129	0.060	0.602
	N	16	16	16	16	16	16	16	16	16	16	16	16	16	16
Field Programming Errors	Corr. Coefficient	.456(*)	.423(*)	.439(*)	0.133	-0.241	-0.209	0.025	-.390(*)	.423(*)	.456(*)	0.366	0.277	<b>.456(*)</b>	<b>-0.041</b>
	Sig. (2-tailed)	0.016	0.026	0.020	0.484	0.204	0.274	0.896	0.040	0.026	0.016	0.054	0.148	0.016	0.827
	N	17	17	17	17	17	17	17	17	17	17	17	17	17	17
No of FPE types	Corr. Coefficient	0.340	0.217	0.217	.650(*)	-0.093	-0.464	0.217	-0.093	0.217	0.340	0.217	0.356	<b>0.217</b>	<b>0.217</b>
	Sig. (2-tailed)	0.229	0.444	0.444	0.022	0.743	0.101	0.444	0.743	0.444	0.229	0.444	0.220	0.444	0.444
	N	9	9	9	9	9	9	9	9	9	9	9	9	9	9
Entropy	Corr. Coefficient	0.319	0.261	0.261	.667(*)	-0.087	-0.203	0.261	-0.203	0.145	0.261	0.145	0.333	<b>0.261</b>	<b>0.435</b>
	Sig. (2-tailed)	0.242	0.338	0.338	0.014	0.750	0.456	0.338	0.456	0.595	0.338	0.595	0.233	0.338	0.110
	N	9	9	9	9	9	9	9	9	9	9	9	9	9	9

\*\*. Correlation is significant at the 0.01 level (2-tailed).

\*. Correlation is significant at the 0.05 level (2-tailed).

**Table 15 Bivariate Kendall's tau correlation test between complexity factors & reliability metrics**

The correlations between the reliability metrics and the complexity factors (created in chapter 5) are not particular high. Besides, it seems that the **reliability metrics: test PRs and the FPEs the correlations, do correlate more with the factor “size” than the factor “structure”**. This is also expressed by low and insignificant correlations between these two reliability metrics and the “structure” related complexity metrics like cyclomatic complexity and maximum nesting level.

The reliability metrics: number of FPE types and Entropy do even correlate less with the complexity factors, which also reflects in the correlations with the independent complexity metrics. However, the high correlation between the “entropy” and the complexity factor “structure” seems logical. A higher entropy score for the SW unit (i.e.: more SWF states and more homogenous distribution of the errors over the SWF states) correlates with a more complex structure of the SW unit.

Most of the correlations between the reliability and complexity metrics are not significant. Therefore only the FPEs will be used for the regression analysis in the next paragraph.

Table 16 shows Kendall's Tau correlation test on SWFs and the complexity metrics doesn't show much correlation. When applying the Pearson correlation test, which requires normal distributed variables, there is a high and significant correlation found between SWFs and the complexity factor size (Field, 2005). Although the variable “SWF” is not normally distributed, it seems that there is a strong relationship between the reliability metrics (PR, FPE and SWF) and the complexity factor size

based on the Pearson correlation test (Table 16). Therefore this single relationship between the factor size and reliability metrics FPE and SWF will be further investigated in the next paragraph. Because of the low correlation scores between the complexity metrics and the reliability metrics “PR, FPE types and entropy”, these metrics are not further investigated and are left out the regression model that is developed in the next paragraph.

**Kendall's tau and Pearson Correlation test (N = 19 for all correlations)**

		CS12 Number of Application Functions	CS13 Number of Lines of code (Logiscope)	CS14 Sum of Cyclomatic No of Application Functions	CS18 Average Size of Statements	CS19 Cyclomatic Complexity (VG)	CS110 Maximum Nesting Level	CS117 Halstead Mental Effort	CCR1 Number of Callers	CSL_JAH1 Total no of Lines of code	CSL_JAH3 No of Comment Lines	CSL_JAH4 No of Files	CSL_JAH5 No of Directories	Factor 1 Size	Factor 2 Structure
<b>Kendall's tau correlation with SW failures</b>	Corr. Coefficient	0.188	0.293	0.258	0.218	0.006	0.106	-0.012	0.082	0.164	0.211	0.112	0.095	0.188	0.235
	Sig. (2-tailed)	0.263	0.080	0.123	0.195	0.972	0.528	0.944	0.624	0.327	0.208	0.506	0.574	0.263	0.161
<b>Pearson correlation with SW failures</b>	Pearson Correlation	0.568*	0.672**	0.565*	0.679**	-0.154	-0.024	-0.181	0.007	0.906**	0.833**	0.666**	0.432	0.701**	0.295
	Sig. (2-tailed)	0.011	0.002	0.012	0.001	0.529	0.924	0.459	0.978	0.000	0.000	0.002	0.064	0.001	0.220

\*\* . Correlation is significant at the 0.01 level (2-tailed).  
 \* . Correlation is significant at the 0.05 level (2-tailed).

**Table 16 Kendall's tau and Pearson correlation tests between complexity metrics and SW failures**

### 6.3 Regression Models

The developed regression model could be used in two ways:

1. It can be used in order to explain the variance in the reliability metrics among SW units the complexity metrics or factors.
2. It can also be used in order to predict the reliability metrics of a certain SW unit based on its complexity metric scores and the established relations between reliability and complexity metrics scores from other SW units.

Paragraph 6.3.1 describes the first approach and paragraph 6.3.2 the second approach.

#### 6.3.1 Explaining the variance in reliability metrics by regression analysis

There are several regression models (Field, 2005):

- Single and multiple regression models; with 1 (single) or more (multiple) predicting variables
- Independent and dependent designs: independent designs use different groups for different conditions of the predicting variables, while dependent designs have repeated measures on the same sample group; however, it requires manipulative predicting variable(s).
- Models that assume that the variables are normally distributed (or parametric) and non-parametric (distribution unknown) variables

Several models were built in order to come to the best regression model. First of all, single regression models were made for both complexity factors and the reliability metrics FPE and SWF. Appendix 14-A shows the results of this analysis, it can be concluded that the **2<sup>nd</sup> polynomial single regression model** with predicting variable “size” and dependent variables “FPE and SWF” gave the best fit with the observed data.

This regression model for the outcome variable FPE is expressed as:

$$FPE = 20.2 * F1^2 + 17.9 * F1 + 6.3, \text{ with } F1 \text{ being the "factor size" complexity score.}$$

The regression model for the outcome variable SWF is expressed as:

$$SWF = 763.6 * F1^2 - 428.9 * F1 - 225.8, \text{ with } F1 \text{ being the "factor size" complexity score}$$

The fit of the regression model with the data is expressed in  $R^2$ , it represents the amount of variation in the outcome variable (FPE or SWFs) that can be explained by the model based on the predicting variable(s) (Field, 2005). For the formulas that are used to calculate the  $R^2$  for the single and multiple regression analysis I refer to Field (2005) and the SPSS15.0 help function.



The single regression models for the factor structure with the reliability metrics FPE and SW (both the linear as well as the polynomial model) didn't fit the data very well. In other words, no applicable regression model for the predicting factor structure and the dependent reliability metrics has been found. However, due to the opinion of other researchers and SW developers at PMS who claim that the factor "size" cannot be used as the only single SW complexity aspect (paragraph 6.1), also a multiple regression analysis is carried out including both complexity factors "size and structure".

Based on all the regression models that are presented in appendix 14-B, it can be concluded that the **2<sup>nd</sup> polynomial single regression models** are showing the highest R<sup>2</sup> scores, even higher than the **multiple linear regression model** which are based on both complexity factors.

However, when the reliability metrics FPE and SWFs are normalized and transformed into  $\sqrt{FPE}$  and  $\sqrt{SWFs}$ , linear regression between the predicting variables and the outcome variables ( $\sqrt{FPE}$  and  $\sqrt{SWFs}$ ) could be expected. In this case the **multiple linear regression model** (including both complexity factors) should be applicable and provide a better fit than the **single linear regression model** based on the complexity factor "size" solely.

The multiple linear regression model including both complexity factors has a better fit (expressed in R<sup>2</sup>) with the data than the single linear regression model based on size solely. The R<sup>2</sup> for the multiple regression model is 0.609 and the R<sup>2</sup> for the single regression model is only 0.607, it can be concluded that this is a negligible difference. Therefore, it can be concluded that the **2<sup>nd</sup> polynomial single regression model** with the predicting variable "size" can **explain** the variance in the reliability metrics FPE and SWFs pretty well for respectively 91% and 85% (when the outliers are excluded).

### 6.3.2 Predicting reliability metrics by the regression model

This paragraph investigates if the regression models, developed in the previous paragraph, can be used in order to predict the reliability metrics FPE and SWFs for a next generation SW units based on their complexity scores. In order to do so, the observations should be subdivided into a "training" group and a "control" group. The parameters for the regression model are derived based on the training group, subsequently the predicted reliability outcomes based on the complexity scores of the control group will be compared with the actual amount of the reliability metrics. Appendix 14-C contains the selection of the SW units into training and control SW units.

In order to predict the FPE for the SW units within the control group based on their complexity scores, the following regression models are used:

- A. 2<sup>nd</sup> polynomial single regression model (F1) to predict FPE
  - $FPE = 22.46 * F1^2 + 14.23 * F1 + 4.79$
- B. Single linear regression model (F1) to predict  $\sqrt{FPE}$ 
  - $\sqrt{FPE} = 3.91 * F1 + 2.37$
- C. Multiple linear regression model (F1 + F2) to predict FPE
  - $FPE = 45.5 * F1 + 3.0 * F2 + 18.4$
- D. Multiple linear regression model (F1 + F2) to predict  $\sqrt{FPE}$ 
  - $\sqrt{FPE} = 3.9 * F1 - 0.13 * F2 + 2.3$

In order to predict the SWFs for the SW units within the control group based on their complexity scores, the following regression models are used:

- E. 2<sup>nd</sup> polynomial single regression model (F1) to predict SWF
  - $SWF = 96.15 * F1^2 - 865.6 * F1 - 382.06$
- F. Single linear regression model (F1) to predict  $\sqrt{SWF}$ 
  - $\sqrt{SWF} = 14.328 * F1 + 11.098$
- G. Multiple linear regression model (F1 + F2) to predict SWF
  - $SWF = 1048.9 * F1 + 1149.8 * F2 + 306.604$
- H. Multiple linear regression model (F1 + F2) to predict  $\sqrt{SWF}$ 
  - $\sqrt{SWF} = 14.1 * F1 + 14.1 * F2 + 11.1$

The outcomes of the predicted FPE and SWF scores for the SW units within the control groups are shown in Table 17 and Table 18. It can be concluded that these predictions are not consistent; there are SW units for which the predicted FPE or SWF score is higher than for other SW units that have higher actual FPE or SWF scores than this specific unit. Besides, most of the predictions are not even coming close to the actual scores. Therefore it can be concluded that these regression models (although their relatively good fit with the data) do not produce reliable predictions for the FPE and SWFs.

This bad performance of these regression models can be explained by overrated  $R^2$ 's due to the small samples (only 12 observations for FPE and only 13 observations for SWF) on which the regression models are based.

Control SW unit*	Actual FPE	Predicted FPE			
		Model A	Model B	Model C	Model D
Acq_Collimator	0.00	2.99	0.33	-4.39	0.41
MiscSys_Automation	3.00	12.41	2.12	-27.53	1.97
PandA_ArchNetwork	0.00	2.82	0.48	-0.72	0.43
UI_NGUI	14.00	2.61	1.84	9.07	1.52
Viewing_Reviewing	22.00	16.80	18.02	39.62	18.00

**Table 17 Actual and predicted FPE for the SW units within the control group**

Control SW unit	Actual SWF	Predicted SWF			
		Model E	Model F	Model G	Model H
Acq_Collimator	0.00	272.86	16.62	-943.22	23.44
Geo_Geometry CV	394.00	-525.97	203.07	502.86	191.03
IP_Image processing	6.00	-326.66	104.82	-135.77	31.60
MiscSys_Session Mgt	35.00	901.62	0.05	639.33	202.91
UI Modules_UIDS	7.00	219.50	20.31	-1141.72	52.49
Viewing_IPISLib	16.00	-534.68	422.49	1194.32	523.05

**Table 18 Actual and predicted SWF for the SW units within the control group**

## 6.4 Explaining variance within reliability by complexity classes

Another approach to establish if the variance within the reliability performance among the SW units can be explained by complexity, is not using the complexity factors “size” and “structure” as independent factors but as a composite complexity class.

Just taking the average scores of the two complexity factors has the following drawbacks:

- The factor scores are based on different metrics and may not have the same underlying distributions.
- Besides, one factor may have much more weight in the average score than the other. Based on the correlation and regression analyses in paragraphs 6.2 and 6.3, it can be concluded that the complexity factors “size” seems to correlate much better with the reliability metrics (FPE and SWFs) than the complexity factor “structure”. Therefore, the “size” complexity scores should weight more than the “structure” complexity scores.

Because of these two reasons it is decided to categorize both “factor” scores into “high”, “middle” and “low” before they are aggregated into one composite complexity class in which the “size” scores have more weight than the “structure” scores. The categorization prescriptions that are used to categorize both factor scores into the three complexity classes: Low (1), Middle (2) and High (3) are mentioned in Table 20. The categorization prescription for the composite complexity categories, again: Low (1), Middle (2) and High (3), is shown in Table 19.

Factor classes	Low	Middle	High
factor size	<0.2	>0.2 & <1	>1
factor structure	<0	>0, & <0.5	>0.5

**Table 20 Categorization prescriptions for factor scores**

Composite complexity class	Factor score size			
	1	2	3	
Factor score	1	1	2	3
structure	2	2	2	3
	3	2	3	3

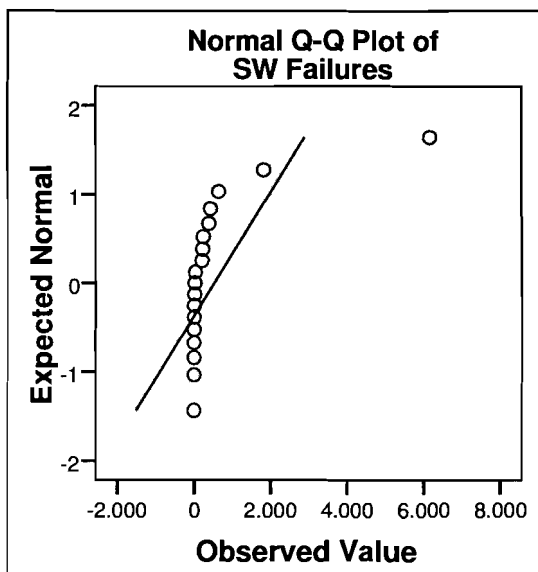
**Table 19 Categorization prescription for composite complexity classes**

The categorization of the SW units into these complexity classes is presented in appendix 15. The categorization of the SW units by their complexity class resulted in the following means, standard deviations and variances in SWFs for each complexity class (Table 21). It can be concluded that the SWF means are ranked consistently with the complexity classes from low to high. However, the variance in SWF among the SW units is not reduced for all complexity classes. The complexity class “high” contains systems that have a much higher variance than the initial variance in SWF among the SW units when no categorization was applied. In other words, categorizing the SW units according to their complexity level does not lead to less variance in each of the complexity classes.

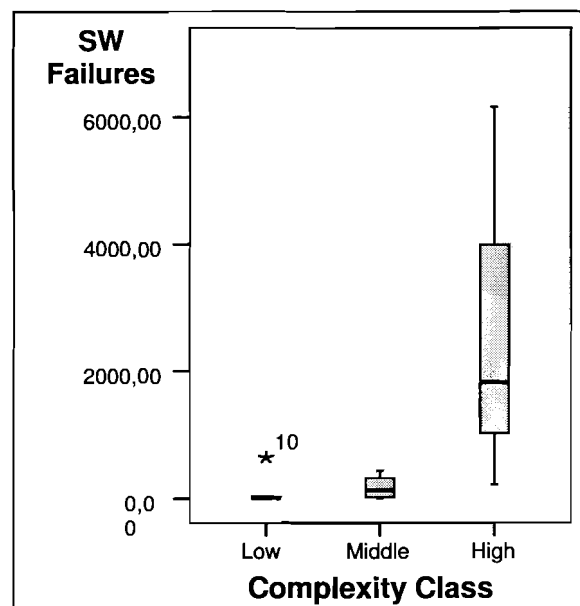
SW	No categorization	Categorization by complexity		
		High	Middle	Low
Mean	542,32	2728,33	172,63	92,25
Standard deviation	1423,63	3070,78	176,77	224,58
Variance	2026723,67	9429682,33	31249,13	50436,50

**Table 21 Means, standard deviations and variance for categorized and uncategorized SW units**

In order to establish if these means are significantly differing from each other, a comparing means test is performed for non-parametric data. Based on this test it can be concluded that the average number of SWF within complexity class “high” is significantly larger than the average SWF within complexity class “middle or low”. The non-parametric tests are selected because the complexity classes are categorical data and the SWFs also do not follow a normal distribution as the Kolmogorov-Smirnov test in SPSS15.0 is significant and the Q-Q plot in Figure 28 do not seem to follow a straight line. The Kruskal-Wallis test is selected because it measures differences between several independent groups for non-parametric data (Field, 2005). It is the counter method of the one-way independent ANOVA test (assuming normal distributed data) used in paragraph 6.3. Performing the Kruskal-Wallis test in SPSS resulted in the box-plots as shown in Figure 29.



**Figure 28 Normal Q-Q plot for SWF**



**Figure 29 Box-plots of SWF per complexity class**

Figure 29 visualizes the high variance among the SWF between the SW units in complexity class “high” (excluding the outlier MiscSys\_Automation; SW unit with low complexity level and rather high no. of SWF). However, the Kruskal-Wallis test statistic shows that the mean ranks (presented in Table 22) do significantly differ among the complexity classes; as the calculated chi-square was 6.294 with 2 degrees of freedom which corresponds with an asymptotic significance level of 0.043. The Kruskal-Wallis test uses the ranking of both variables (SWF and complexity) because of the non-normal distributed data (Field, 2005). The mean rank represents the average rank of the SWFs in each complexity class.

	Complexity Class	N	Mean Rank
Software Failures	Low	8	6.94
	Middle	8	10.69
	High	3	16.33
	Total	19	

**Table 22 Mean rankings of the complexity classes: low, middle and high.**

## 6.5 Conclusions on software complexity versus reliability

### **Software reliability and complexity definitions & availability and quality of required data**

This master thesis has defined the desired and available reliability indicators and complexity metrics for the software within the Allura Xper CV X-ray scanner at PMS.

The available reliability performance indicators were:

- Problem reports (PR) during testing
- Field programming errors (FPE) derived from the system log-files
- Software failures (SWF), derived from the failure classification by the FMEA developed by the PhD researcher K. Kevrekidis.

These reliability indicators did correlate among each other, so it can be assumed that they are all representative for the actual reliability performance.

The complexity metrics for the SW subsystems and SW units were derived from the carried out Logiscope audit and the JAH intranet site within the CV development department of PMS.

Over 40 separate complexity metrics were derived and transformed into two common complexity factors “size” and “structure”, for which composite factor scores were defined.

For 17 observations (SW units) all complexity metrics and reliability metrics PR and FPE were available and for 19 observations all complexity metrics and reliability metric SWF are available.

### **Established relations between complexity factors and reliability metrics**

The established relations between the developed complexity factors and the reliability metrics FPE and SWF are expressed in the following three ways:

- *Correlation between reliability and complexity metrics:*

Based on the Kendall’s tau correlations test between the two complexity factor scores and the reliability metrics FPE, it can be stated that there is a significant correlation between the complexity factor “size” and the reliability metric FPE. The same conclusion can be derived for the complexity factor size and the reliability metric SWF based on the Pearson correlation test, the reason for the low scores with Kendall’s tau correlation test cannot be explained.

- *Explaining the variance within FPE and SWF based on the defined complexity factors:*

The developed regression models in chapter 6 can be used in order to explain the variance within the reliability metrics FPE and SWF rather well. The variance in the FPE and SWF scores among the observed SW units could be explained best by the complexity factor “size”.

- *Prediction of the FPE and SWF based on the SW unit’s complexity factor scores:*

The regression model based on two factors F1 and F2 is over rated, due to the small number of observations within the training group, and can therefore not be used in order to predict reliability metrics FPE or SWF for the SW units in the control group accurately.

### **Classification of SW units based on the complexity factors “size” and “structure”**

The classification of the SW units into groups with similar complexity levels resulted in the following conclusions about the complexity categories:

- The average SWF score of the SW units within a higher complexity classes is higher than the average SWF score for SW units within the lower complexity classes.
- The ranked SWF means (by the Kruskal-Wallis test) within the higher complexity classes are significantly higher than within the lower complexity classes.
- The variance in the SWF scores among the SW units is, in comparison with the unclassified group of SW units, reduced for the lower complexity classes (low and middle). In contrary, the variance in the SWF scores among SW units classified as “high” complex is increased compared to the unclassified group of SW units. In other words; an increase in complexity also causes an increase in the variance within the reliability performance (expressed in SWF).

## 7. Hardware complexity versus reliability

This chapter includes only a quick and short investigation on the relationship between hardware complexity and hardware (HW) reliability. Due to the extended research on software (SW) complexity and its relation with software reliability in chapter 4, only limited time was available for HW complexity. This chapter consists out of a brief description on the available HW related reliability metrics (paragraph 7.1) and a small survey that is used in order to establish the HW complexity for the Allura Xper CV scanner at PMS (paragraph 7.2). Paragraph 7.3 describes the found relationships between the complexity metrics and the reliability metrics. Finally, the results and findings are summarized in paragraph 7.4

### 7.1 Hardware reliability performance indicators

In the Master thesis preparation (Albers, 2008) a number of indicators for HW reliability performance were selected like, MTBF or Failure probability. However, none of the indicators are available for HW components at PMS. The only available data that could indicate the performance of the HW is the “refined” call data into component replacements (corrective maintenance) for a panel of 50 systems for each Product Base Line (PBL). This translation is carried out quarterly by Customer Support within the CV department of PMS. The component replacements can be caused by HW or SWFs, and therefore this indicator is not very accurate but it is the best indication for the HW reliability performance available and will be used in this master thesis.

The hardware of the Allura Xper CV scanner is subdivided into the following HW building blocks: Table (AD5/AD7), M-Cabinet, Monitors, Operator controls, X-ray tube (PolyG2/Clea), R-Cabinet, X-ray detector (Velara), and Xtravision. A more detailed description of these building blocks can be found in appendix 1.

The reliability scores, expressed in “component replacements per HW unit”, are shown in Table 23 for all the Product Base Lines of the Allura Xper CV scanner. The review period varies for each PBL: 4 years for PBL10, 2 years for PBL30, 3 years for PBL20, and 1.5 years for PBL40.

HW Units:	PBL20	PBL40	PBL10	PBL30
AD5	Confidential			
M-Cabinet				
Monitors				
Operator Controls				
X-ray tube				
R-Cabinet				
X-ray detector				
Xtravision				

Table 23 Overview of component failures over

### 7.2 Hardware complexity and selected complexity metrics

Based on the literature review in the Master thesis preparation, the hardware system complexity can be divided into two groups, which are:

- Qualitative complexity theories like: *system symmetry* (Flood and Carson, 1990), *Non-Holonomic constraint* (Zhao and BeMent, 1992), *shape complexity* (Rodriguez-Toro et al., 2003), *system integration and differentiation* (Heylighen, 1996), and *simplicity* (Bunge, 1962)
- Quantitative complexity attributes: these are group of system attributes that represent the complexity of the product (Calvano and Joh, 2003; Hobday, 1998; Barlcay and Dann, 2000) like for example the number of components.

The last approach is selected in order to come up with a “hardware” complexity measure for the Allura Xper system. The following complexity attributes were found in the literature mentioned in the paragraph above:

- Number of hierarchic layers in the architecture
- Number of unique components
- Relations and interactions between components
- Number of design updates or changes
- Criticality of the component
- Degree of technological novelty or required specialism
- Number of functional specialists involved
- Degree of embedded (control) software

All of these attributes seem to be plausible to express the complexity for the Allura Xper and are selected in agreement of M. Loos, System architect in the CV development department at PMS. For some of these attributes, it is possible to retrieve quantitative measures. These attributes are, the number of hierarchic layers, number of components, number of design updates or changes and the number of functional specialists involved. Quantitative measures for the other attributes are almost impossible to retrieve. Due to the master thesis’s time limitation and the laboriously process, none of the attributes are quantified based on available data at PMS.

However, these eight complexity attributes were transformed into a small survey in which the subsystems mentioned in paragraph 7.1 were ranked in the categories: low, middle and high, by two system architects in the CV development department at PMS. The subsystems were ranked by comparing them with each other, this way it is not possible to score all subsystems “high” for a single complexity attribute. The reason to rank the subsystems in this way was to prevent that all subsystems score the same on all complexity attributes and no difference within and among the complexity attributes is defined. This would make the complexity attributes inapplicable for explaining the differences in the reliability performance among the subsystems.

*Questionnaire:*

1. How many hierarchic levels does each subsystem contain?
2. How many unique components does each subsystem contain?
3. How integrated are the components in each subsystem? Weakly integrated subsystems have components that are hardly related or interfering with each other while highly integrated subsystems contain many relations and interference between its components.
4. How many design updates/changes do occur in the subsystem? Is the subsystem a kind of constant subsystem that doesn’t change much or is it changed radically in each new PBL or even during one PBL?
5. How critical is the component? Criticality in terms of; “does a failure of this subsystem cause serious hazardous situations during operation” and/or “does a failure of this subsystem affect the other subsystems as well which results in expensive or time consuming repairs”.
6. How complex or unique is the applied technology within the subsystem?
7. How many functional specialists are involved in the design process for each subsystem? (System architects, designers, software programmers, mechanical engineers, etc.)
8. How much control software is involved for running the subsystem?

*Results of the questionnaire:*

The results from the ratings by the two system architects are presented in appendix 16. The subsystems are ranked according to their “overall” complexity, which is calculated by the average score of all the attributes scores for each subsystem. In order to come to these complexity score, the rankings: low, middle and high are translated into 1, 2 and 3. Therefore a higher score on the complexity attributes represents a more complex subsystem.

The rankings of the raters were not totally consistent but when their rankings are categorized into subsystems with low complexity and high complexity, the ranking is consistent for both raters and also corresponds with the ranking of the average rater scores as can be seen in Table 24. The ranges for low complexity are 1 till 3 for Rater 2 & Average rating, and. 1 till 4 for Rater 1. This difference in classification is caused by the first rater, for which some subsystems had equal ranks.

Because of the consistent ranking in Table 24, it is assumed that the average scores of the two raters represent the complexity levels of the subsystems in a consistent way. The overall complexity scores based on the averaged rating of the two system architects is shown in Table 25.

Subsystem:	Ranking rater 1	Ranking rater 2	Average ranking	Complexity level
Monitors	1	1	1	Low
R-cabinets	2	3	2	Low
Operator Controls	3	1	3	Low
Xtravision	4	2	4	Low
X-ray detector	5	4	5	High
Table	6	5	6	High
X-ray tube	7	6	7	High
M-cabinet	8	4	7	High

Table 24 Overview of complexity rankings

Complexity attributes:		Table	M-Cabinet	Monitors	Operator Controls	X-ray Tube	R-Cabinet	X-ray Detect or	Xtravision
1	Number of Hierarchic levels	3	2	1	1,5	3	1,5	1,5	1,5
2	Number of components	3	2,5	1	1	3	2	2	1
3	Relations between components	3	2	1,5	1,5	3	1	2,5	1,5
4	Number of design updates / changes	1,5	3	1,5	2	2	3	1,5	1
5	Criticality of the component	2	3	1	1	2	1,5	3	1
6	Technology level	2,5	2,5	1	1,5	2,5	1,5	3	1,5
7	Number of specialists involved	2	2,5	1	1	3	1,5	2	1,5
8	Degree of control software	2,5	3	1	1,5	2,5	1,5	2	3
<b>Overall complexity score</b>		2,44	2,56	1,13	1,38	2,63	1,69	2,19	1,50
<b>Ranking</b>		6	7	1	2	7	4	5	3

Table 25 Average results from raters 1 and 2

*Validation and interpretation of the HW complexity metrics:*

There is no standard metric or reference metric that can be used in order to validate if the complexity attributes give a correct representation of the actual system complexity.

Validation of the HW complexity metrics isn't that straightforward as for Software in which standardized programming rules are used in order to verify if a code is correct or not. Still there are some rules applied within PMS like: one aims for a maximum of 10 subsystems in the architecture. (Communications with M. Loos, system architect at PMS) Also in literature, they generally apply questionnaires (like in Barclay and Dann, 2000) instead of actual quantitative data. Therefore validating the complexity scores for HW seems undoable and the questionnaire is used in order to draw conclusions on the relationship between HW complexity and the HW reliability performance.

Furthermore, the number of observed HW subsystems is rather small (only 8 subsystems) and therefore statistically grounded conclusions on correlations between the subsystem's complexity scores and reliability scores can not be extracted from this small sample.

In order to see if there are some similarities between the rankings in reliability and complexity, all subsystems for all four PBLs will be joined together into a **larger sample** of 30 observations. (7 subsystems multiplied by 4 PBLs plus two times the optional subsystem "Xtravision" from the PBL20 and PBL40). Based on this sample conclusions will be drawn on the relation between the complexity

classes (high/low) and the reliability performance of the subsystem, this analysis is described in paragraph 7.3.2.

The classification of the average complexity scores over all complexity attributes into **complexity classes (low/high)** are validated by a correlation test (Kendall's tau correlation test within SPSS15.0) between the complexity attributes scores and their assigned complexity class. Based on the results from this correlation test (Table 26), it can be concluded that there is a significant correlation between the complexity class and the complexity attribute scores. This supports the conclusion that the complexity classes can be used as an **aggregated complexity score** for the attributes: no. of hierarchic levels, no. of component, integration level, criticality, technology level, no. of specialists and degree of control software.

**Kendall's tau Correlations (N= 30)**

		Hierarchical Levels	No of Components	Integration Level	Design Changes	Criticality	Technology Level	Specialists	Control Software
Complexity Classes	Corr. Coefficient	,808(**)	,746(**)	,404(*)	0,283	,746(**)	,808(**)	,746(**)	,559(**)
	Sig. (2-tailed)	0,000	0,000	0,021	0,107	0,000	0,000	0,000	0,001

\*\* . Correlation is significant at the 0.01 level (2-tailed).

\* . Correlation is significant at the 0.05 level (2-tailed).

**Table 26 Kendall's Tau correlations between complexity classes and complexity attributes**

## 7.3 Relationship between hardware complexity and reliability

This paragraph investigates if there are some similarities between the reliability metrics found in paragraph 7.1 and the complexity attributes in paragraph 7.2 This paragraph is subdivided into two subparagraphs, the first one describes the expected relationship between the HW complexity attributes and the HW reliability performance based on logical reasoning and the findings in the literature study during the master thesis preparation. Paragraph 7.3.2 visualizes the complexity and reliability metrics and their similarities for all the subsystems of the Allura Xper system. Finally, paragraph 7.3.3 describes whether the categorization by complexity actually reduces the variance of the reliability performance.

### 7.3.1 Expected relations between HW complexity attributes and reliability

The following relationships between the HW complexity attributes and the reliability performance are assumed:

- *Number of hierarchic layers in the architecture:* the more hierarchic layers the harder it is to understand the architecture and less orderly the subsystem will be. This is supposed to have a negative impact on the reliability performance.
- *Number of unique components:* the more unique components are used, the more difficult the system will be to understand and how more unique problems will arise.
- *Relations and interactions between components:* the more components depend on each other the more vulnerable the entire system will be to a failure of a single component in the system.
- *Degree of technological novelty or required specialism:* the more complex technology is applied, the more difficult it will be to cope with all factors that affect the system performance. The more innovative the applied technology, the less is known about how this technology will behave in practice and the more problems may arise.
- *Number of functional specialists involved:* a large number of functional specialists may result in more contradictions in the functional interests, which makes weighing the interests more difficult. Furthermore, a large number of functional fields may also lead to a wide scale of functional related problems.
- *Number of design updates or changes:* When one assumes that design updates or changes are carried out because of incompetence of the system to perform its function(s), the number of changes or updates may be an indicator of many failures of this (sub)system in the past (field reliability).
- *Degree of embedded (control) software:* By assuming that the level of control software resembles the degree of interaction between system software and system hardware, an increase of this interactions may lead to more failures because the HW performance now strongly depends on the SW performance and vice versa.



These complexity attributes are supposed to have a negative impact on the reliability performance, in other words; the more higher the attribute scores the lower the reliability performance of the system and the more failures are found.

- *Criticality of the component*: Component criticality comprehends catastrophic consequences for the system performance or its operators in case the component fails. The more critical the component, the more effort will be spent in designing the component properly. Much effort spent during the design and test phase may lead to less field failures.

Therefore this complexity attribute is supposed to have a positive affect on the system’s field reliability performance.

### 7.3.2 Verify assumed relations between HW complexity and reliability

First of all, Kendall’s Tau correlation test (in SPSS15.0) is applied on the complexity attributes scores and number of failure for all 30 observations.

**Kendall's tau Correlations (N = 30)**

	Hierarchical Levels	No of Components	Integration Level	Design Changes	Criticality	Technology Level	Specialists	Control Software	Complexity Class
Failures Corr. Coefficient	0,117	0,067	0,165	0,037	-0,078	-0,045	0,067	-0,207	0,019
Sig. (2-tailed)	0,422	0,646	0,259	0,802	0,593	0,760	0,646	0,157	0,901

\*. Correlation is significant at the 0.05 level (2-tailed).

**Table 27 Correlations between number of failures and complexity attribute scores**

As Table 27 shows, there was not even a single significant correlation found between the failures and complexity attributes. Therefore it was decided to also classify the number of failures into failure classes and compare the categorical variables with each other into a contingency table.

A **contingency table** with a Chi-square test is suitable to establish if there is a relation between two categorical variables (Hair, 2006), therefore this method is selected in order to analyze the relation between the variables HW complexity and HW reliability.

However, this failure classification for the variable “HW reliability”, expressed in the number of failures found in the panel analysis for PBL10 till PBL40, needs to be carried out first. The failures are categorized in two ways: one with 2 failure classes (high and low) and another one with 3 failure classes (high, middle, and low). **Failure class “high”** stands for a subsystem with many failures and failure class “low” represents subsystems with a low number of failure found, the failure classifications are shown in Table 28.

Subsystem	PBL10	PBL30	PBL20	PBL40	PBL10	PBL30	PBL20	PBL40
Table	2	2	6	5	2	2	6	5
M-cabinet	4	5	5	6	4	5	5	6
Monitors	3	3	3	4	3	3	3	4
Operator controls	7	7	8	8	7	7	8	8
X-Ray Tube	6	6	7	7	6	6	7	7
R-Cabinet	5	4	4	2	5	4	4	2
X-Ray Detector	1	1	2	3	1	1	2	3
Xtravision	X	X	1	1	x	x	1	1
Sample size: 30	<b>Two failure classes:</b>				<b>Three failure classes:</b>			
	High	Low			High	Middle	Low	
The failures of the units within PBL10 and PBL30 are ranked from 1 till 7								
The failures of the units within PBL20 and PBL40 are ranked from 1 till 8								

**Table 28 Reliability classifications**

Thanks to these failure and complexity classifications the contingency tables could be created and the Chi-square tests were performed (in SPSS15.0). There are two contingency tables, one including the failure classification low/high and another including the classes: low, middle, and high. Table 29

shows that, as expected, the largest proportion of subsystems with a low complexity score are placed under failure class “low” ( $\pm 65\%$ ). The subsystems with a high complexity score do not show such an obvious difference in the proportions over the failure classes, only around 55% is placed in failure class “high”. One could conclude that subsystems with a low complexity score do experience only few failures in the field and that complex subsystems do naturally experience many field failures.

		Failure classes		Total
		Low	High	
Complexity	Low	9	5	14
Classes	High	7	9	16
Total		16	14	30

**Table 29 Contingency table complexity with failure classes “low” and “high”**

However, in order to confirm this conclusion the null hypothesis from this contingency table should be rejected in order to prove that these two variables are truly not independent from each other. In order to do so, the chi-square should be 2.71 for a confidence coefficient of 90% in a contingency table with only one degree of freedom. The calculated chi-square is only 1.265 and therefore the null hypothesis cannot be rejected at a confidence level of 90%, it could only be rejected at a significance level of 0.261 (a confidence coefficient of 74%). Therefore it is concluded that there is not a strong significant relation between a subsystem’s complexity class and its failure class for the 30 observed subsystems within PBL10, PBL30, PBL20 and PBL40.

The contingency table for the failure classification into low, middle and high resulted even in a lower chi-square (0.268) and it also didn’t satisfy the conditions that all cells are filled with at least 5 observations. All contingency tables, calculated chi-squares and corresponding significance levels can be found in appendix 16.

### 7.3.3 Variance within reliability after complexity classification

The categorization of the subsystems into high and low (from paragraph 7.2) and the related failure scores for each subsystem can be found in appendix 17.

Table 30 shows the mean and variance for subsystems with and without the categorization according to complexity. The variance between the failure scores without categorization is 4002 failures. After categorizing the subsystems into “high” and “low” complexity scores, the variance in failures for the complexity class “high” was reduced till 2671 failures. However, for the subsystems in complexity class “low”, the variance increased till 5775 failures.

Therefore, it can also be concluded that categorizing the subsystems according to their complexity classes will not reduce the “within group” variance for both complexity classes and create more homogeneous groups. This is not such a surprising conclusion, it is a confirmation that the conclusion in paragraph 7.3.2 (complexity and failures are not related) is correct.

	No categorization	Categorization by complexity	
		High	Low
Mean	68,73	63,56	74,64
Standard deviation	63,26	51,68	76,00
Variance	4002,27	2671,20	5775,48

**Table 30 Variance for subsystems with and without categorization according to complexity**

## 7.4 Conclusions on hardware complexity versus reliability

### **Hardware reliability and complexity definitions & availability and quality of required data**

There were no real hardware reliability metrics available for the Allura Xper systems within PMS. Only the number of component replacements (during “corrective” maintenance) is available as an indicator for the hardware reliability performance. Although a component replacement can also be caused by a software failure, it is assumed that the replacements are all caused by hardware failures. This assumption reduces the trustworthiness of the conclusions on the relationship between hardware reliability and complexity. Nevertheless, the results in chapter 7 still present an indication of how the hardware complexity and reliability are interrelated for the PMS Allura Xper systems.

As the investigation of software complexity and its impact on software reliability was very time consuming, the HW complexity metrics for the HW building blocks were based on the results of a short questionnaire among 2 system architects within the CV development department of PMS.

For 30 observations (HW building blocks) all defined complexity attributes and the reliability metric “hardware failures” (no. of component replacements in panel analyses) were available.

### **Established relations between hardware complexity and reliability**

No significant correlations have been found among the HW complexity attributes or classes and HW failures. Therefore the complexity metrics cannot be used in order to explain the variance within the HW failures, let alone predict the number of HW failures.

When categorizing the HW building blocks by complexity and HW failures (low/high), a contingency table was used in order to establish if there is a relationship. However, the test was not significant and increasing the number of classes would only deteriorate the significance of the chi-square test.

### **Categorizing the HW building blocks based on its complexity scores**

The HW building blocks were categorized into two groups: building blocks with low complexity scores and building blocks with high complexity scores. The average number of HW failures (and its variance) was larger for the group which included the “low complex” building blocks. This result contradicts with the expectation that more complex HW building blocks will on average have a higher number of HW failures. The only logical explanation for this remarkable result is that the reliability metrics or the complexity scores are not relevant or representative for the actual situation.

## 8. Conclusions & recommendations

This final chapter contains the conclusions (paragraph 8.1) and recommendations (paragraph 8.2) based on the research finding in chapters 4 till 7.

### 8.1 Conclusions

#### Reflection on the research question

The general research question was to gain more insight into the relationship between system complexity and the system's reliability performance. This master thesis came up with some conclusions on this relationship for the software units as well as the hardware building blocks based on the Allura Xper systems at PMS. Although, the relationship between hardware complexity and reliability has been investigated less thoroughly, this thesis came up with an overview of the required and available data at PMS for the reliability factor "system complexity".

Furthermore, all research questions are answered except for the last sub question: "*How can extended knowledge about system complexity and its relation with reliability performance assist the decision process in product development*". Reason for this, was the lack of time to investigate the decision process in product development. The other research questions and corresponding findings are as follows:

1. *How should the reliability performance of the professional system be defined or measured?*

This research defined several software related reliability indicators that could be used in order to express the reliability performance of the Allura Xper system. These indicators were: test problem reports (PR), field programming errors (FPE), mean time between programming errors (MTBF), entropy and classified software failures (SWF). For hardware only one reliability indicator was available, which was the number of component replacements from the panel analyses.

2. *How should "system complexity" be defined in such a way that there will arise no misconceptions in interpreting these factors?*

First the required system complexity metrics were defined for both hardware and software. Subsequently the available complexity metrics were defined / measured. For software complexity, around the 40 code metrics were collected, these metrics were transformed into two complexity factors "size" and "structure". Furthermore, the metrics from different sources were compared in order to validate if the complexity metrics scores were representing the software complexity consistently. There were 8 hardware complexity attributes defined and quantified by a small survey among system architects. These eight complexity attributes were transformed into an aggregated complexity measure.

3. *How should the relationship between "system complexity" and the reliability performance of the system be established?*

Software complexity versus reliability: The relationship between system complexity and reliability has been researched in several ways. First of all, several metrics are used to express system complexity and the reliability performance and they are independently investigated. Secondly, there are several types of tests performed:

- Correlation tests among complexity factors and the reliability indicators: SWF, FPE, PR, MTBF and entropy.
- Regression analysis among complexity factors and the reliability indicators FPE and SWF.
- Chi-square tests (contingency tables) for the software units and hardware building blocks categorized according to its complexity and reliability.

These analyses were performed independently for the system's hardware and software.

It shows that the complexity factor size does correlate with the number of FPE and SWF, and factor "structure" does not. The correlation can be explained by the assumption that larger software code

leads to more FPEs or SWFs due to higher chance of mistakes by the programmer. However, the absence of correlation between the factor structure and FPE and SWF can not be explained with certainty, but probably the “structure” related metrics show less discriminative scores among the software units and therefore cannot explain the variance in FPE or SWF.

Furthermore, the developed regression models are based on only a small number of observations. Therefore there is a serious risk that the regression models are over-fitting the data and cannot be used in order to predict the reliability performance in FPE or SWF accurately.

#### Hardware complexity versus reliability:

There were no correlations found between the hardware complexity attributes and the reliability indicator “component replacements”. Therefore no conclusions on this relationship are found.

#### *4. How to create a product classification process based on system complexity?*

The software units and the hardware building blocks were classified according to its complexity scores as well as its reliability performance level. Based on these classifications it can be concluded that the means and variance of the reliability performance (expressed in SWF) increases when the group of software units are classified as “high” complex. For the hardware building blocks, the variance within reliability performance could not be explained by the classification of the building blocks according to complexity.

#### **Contribution to scientific research**

The main contribution to the scientific research is the overview of the complexity and reliability metrics that are required in order to investigate the relation between these two variables. Moreover, a data set and its limitations are provided, so other researchers can use and interpret this data set in a correct way as well. This is important as this master thesis is a contribution to the Ph.D. research of K. Kevrekidis at the Quality & Reliability Engineering department at Eindhoven University of Technology. Furthermore, this research combines theoretical definitions and approaches of system complexity and tries to apply them in practice. It shows that many theoretically important complexity concepts (or software code metrics) are not that relevant in practice or cannot be used at all in order to investigate the relation with system reliability.

#### **Strengths and weaknesses of the research**

##### *Weaknesses:*

Weak point in the research is the selection of code complexity metrics based on their correlation with the programming errors instead of the (by FMEA) classified software failures. This decision is made due to the available data at that time, but for repeated factor analyses it is recommended to limit the number of software code metrics by selecting the with software failures correlated code metrics. A second weak point is the limited number of observations in the training group on which the regression models are based. This led to overrated regression models, not applicable for prediction of the software failures based on the complexity factor scores. Another weak point is that the impact of other extraneous moderating variables is not quantified. However, creating an experimental setting is rather difficult when investigating an expensive and complex professional system like the Allura Xper system at PMS.

##### *Strengths:*

Strength of this research is that not just a general data set (like the dataset provided by Kitchenham, 1987) is used in order to investigate the relation between complexity and reliability, but an actual and updated data set has been created and used for investigation. Therefore the conclusions can be interpreted in a more corrective way, which is very important when vague concepts like system complexity and system reliability are investigated. By working closely together with the system architects and software developers within the CV department of PMS, the selection and measurement of the complexity metrics is not only based on theoretical knowledge but also on practical relevance. Together with the fact that several information sources are used (in order to validate the metric scores among the sources), it can be concluded that the used complexity metrics do represent the actual system complexity well.

## 8.2 Recommendations

### Software complexity versus software reliability

The following topics and directions are recommended for further research within investigation of the relationship between SW complexity and SW reliability:

- Find more discriminative complexity code metrics for “structure” related complexity.
- Repeat the regression analysis for a larger training group and investigate if the accuracy of the reliability predictions based on the complexity scores improves.
- Add the (in literature) recommended complexity metric “fan in / fan out” to the code complexity model and investigate if more complexity factors arise or if the ability to explain the variance in the reliability scores among SW units by the complexity factors improves.
- Investigate which underlying reasons cause the increased variance in software errors and failures for systems with high scores on the complexity factor “size”.
- As the complexity factor “size” seems to be related to reliability, the number of SWF could be normalized by for example the Lines of Code (LOC) in order to provide more representative and fair measures of the actual reliability performance of the SW units.
- Perform more case studies on other complex professional repairable systems that include (embedded) software, and discover if there are similar findings and conclusions. Subsequently, one can investigate if these findings can be combined into a general applicable regression model that fits the SW of complex professional repairable systems in general.
- Find out how the established relations between complexity and reliability can provide more focus/insight for the development team and support them in making product improvement decisions.

### Hardware complexity versus hardware reliability

The following topics and directions are recommended for further research within investigation of the relationship between HW complexity and HW reliability:

- This master thesis investigated this relationship quickly and not very thoroughly, therefore more extended investigation of this relationship is recommended. Especially because not much quantitative research on this relationship has been carried out in literature yet.
- Within PMS, effort should be spent on getting insight in the actual failure causes that have led to the component replacements (during “corrective” maintenance). The required reliability measures should be defined upfront the process of data collection and filtering, in order to make sure that all required reliability data is available.

### General recommendations for further research

Here follow some general recommendations for further research on the research topics system complexity and system reliability:

- A challenging research would be to quantify the interactions between hardware and software reliability, and see how system complexity influences this interaction.
- Carry out more quantitative case studies on several types of complex professional repairable systems in order to see if the conclusions and regression models from this research can be generalized.
- Investigate other factors (for example the in this master thesis called extraneous variables: “environment” and “user profile”) that could explain the variance within the reliability performance.

## References

- Ascher, H.E., Feingold, H., 1984, *Repairable Systems Reliability: Modeling, Inference, Misconceptions and their Causes*, New York: Dekker, 1984
- Baker, M.D., 1991, Implementing an Initial Software Metrics Program, *Proceedings - IEEE National Aerospace and Electronics Conference*, 3, pp 1289-1294
- Barclay, L., Dann, Z., 2000, New-product-development performance evaluation: a product-complexity-based methodology, *IEE Proceedings-Science, Measurement and Technology*, 147(2)
- Brombacher, A.C., Sander, P.C., Sonnemans, P.J.M., Rouvroye, J.L., Managing product reliability in business processes 'under pressure', *Reliability Engineering and system Safety*, 88, pp 137-146
- Bunge, M., 1962, The complexity of Simplicity, *The Journal of Philosophy*, 59(5), pp 113-135
- Calvano, C.N., John, P., 2004, Systems Engineering in an Age of Complexity, *Systems Engineering*, 7(1), pp 25-34
- Chidamber, S.R., Kemerer, C.F., 1994, A Metrics Suite for Object Oriented Design, *IEEE Transactions on Software Engineering*, 20(6), pp 476-493
- O'Connor, P.D.T., *Practical Reliability Engineering*, Chichester: Wiley, 1991
- Cornacchio, J.V., 1977, Maximum-entropy complexity measures, *International Journal of General Systems*, 3(4), pp 215-225
- Coskun, C., Grabowski, M., 2005, Software complexity and its impacts in embedded intelligent real-time systems, *The Journal of Systems and Software*, 78, pp 128-145
- Da-Wei, E., 2007, The Software Complexity Model and Metrics For Object-Oriented, *IEEE International Workshop on Anti-counterfeiting, Security, Identification, ASID*, pp 464-469
- Doty, L.A., 1989, *Reliability for the technologies*, 2<sup>nd</sup> edition, New York: Industrial Press, 1989
- Ebert, C., 1996, Classification techniques for metric-based software development, *Software Quality Journal*, 5, pp 255-272
- Edmonds, B., 1999, What is Complexity? - the Philosophy of Complexity per se with application to some examples in evolution, In F. Heylighen and D. Aerts, *The Evolution of Complexity*, Dordrecht, Kluwer
- Fenton, N.E., Pfleeger, S.L., *Software metrics: a rigorous and practical approach*, London: International Thomson Computer Press, 1996
- Field, A., *Discovering statistics using SPSS*, London: SAGE publications, 2005
- Flood, R.L., Carson, E.R., 1990, *Dealing with complexity: an introduction to the theory and application of systems science*, 3<sup>rd</sup> edition, London, Plenum, 1990
- Glaser, B.G., Strauss, A.L., *The discovery of grounded theory: strategies for qualitative research*, Chicago: Aldine, 1967

- Hair, J.F., Black, W.C., Babin, B.J., *Multivariate data analysis*, Upper Saddle River: Prentice Hall, 2006
- Harrison, W., 1992, An Entropy-Based Measure of Software Complexity, *IEEE Transactions on Software Engineering*, 18(11), pp 1025-1029
- Henry, S., Kafura, D., 1981, Software structure metrics based on information flow, *IEEE Transactions on Software Engineering*, 7(5), pp 510-518
- Heylighen, F., 1996, The Growth of Structural and Functional Complexity during Evolution, published in: F. Heylighen & D. Aerts, "*The Evolution of Complexity*", Kluwer Academic Publishers, 1996
- Heylighen, F., 2002, Complexiteit en Evolutie, *Cursus nota's 2003-2004 Vrije Universiteit Brussel*
- Hobday, M., 1998, Product Complexity, Innovation and Industrial Organization, *working paper by CoPS Innovation Centrim at CENTRIM/SPRU funded by the ESRC*
- Kantz, H., Koza, C., 1995, The Elektra Railway Signalling-system: Field experience with an actively replicated system with diversity, *Proceedings – IEEE Annual International Conference on Fault-Tolerant Computing*, pp. 453-458
- Khoshgoftaar, T.M., Munson, J.C., 1990(A), The lines of code metric as a predictor of program faults: A critical analysis, *Proceedings - IEEE Computer Society's International Computer Software & Applications Conference*, pp 408-413
- Khoshgoftaar, T.M., Munson, J.C., 1990(B), Predicting software development errors using software complexity metrics, *IEEE Journal on Selected Areas in Communications*, 8(2), pp 253-261
- Kitchenham (1987), Towards a Constructive Quality Model: Part II : Statistical Techniques for Modelling Software Quality in the Esprit Request Project, *IEEE Software engineering journal*, 2(4), pp 114-126
- Kitchenham, B.A., Linkman, S.J., 1990, Design Metrics in Practice, *Information and Software Technology*, 32(4), pp 304-310
- Kumar ,R., *Research methodology: a step-by-step guide for beginners*, London: SAGE publications, 1999
- Lancon, E., Saclay, C.E., 1995, Software Metrics to Improve Software Quality in HEP, *CHEP '95 Computing for the Next Millennium*, Rio de Janeiro, September
- Lange, C.F.J., 2005, Course in Software Architecting (2II10), Faculty: Mathematics & Computer Science at the Technical University Eindhoven.
- Lew, K.S., Dillon, T.S., Forward, K.E., 1988, Software Complexity and Its Impact on Software Reliability, *IEEE Transactions on Software Engineering*, 14(11), pp1645-1655
- Lewis, E.E., 1996, *Introduction to Reliability Engineering*, New York: Wiley, 1996
- Löfgren, L., 1977, Complexity of Descriptions of Systems: A foundational Study. *International Journal of General Systems*, 3, 197-214.
- Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, CN., Diot., C., 2004, Characterization of Failures, in an IP Backbone, *Proceedings - IEEE INFOCOM* , 4, pp 2307-2317



- McQuaid, P.A., 1996, Profiling Software Complexity, *A PhD dissertation submitted to the graduate faculty of Auburn University, Alabama*, August 30.
- Miller, J.E., Kulp, R.W., Orr, G.E., 1984, Adaptive probability distribution estimation based upon maximum entropy, *IEEE Transaction on Reliability*, 33, pp 353-357
- Munson, J.C., Khoshgoftaar, T.M., 1991, The Use of Software Complexity Metrics in Software Reliability Modeling, *Proceedings - IEEE International Symposium on Software Reliability Engineering*, pp. 2-11
- Munson, J.C., Khoshgoftaar, 1993, Measurement of Data Structure Complexity, *IEEE Software*, 9(11), pp 217-225
- Nakagawa, T., Yasui, K., 2003, Note on reliability of a system complexity considering entropy, *Journal of Quality in Maintenance Engineering*, 9(1), pp 83-91
- Neufelder, A.M., *Ensuring software reliability*, New York: Dekker, 1993
- Prasad, B., 2001, Product, Process and Methodology, Systematization to Handle Structural and Computational Complexity in Product Realization, *Systems research and behavioral science*, 18, pp 523-543
- Ren, Y.T., Yeo, K.T., 2006, Research challenges on complex product systems (CoPS) innovation, *Journal of the Chinese Institute of Industrial Engineers*, 23(6), pp 519-529
- Rodriguez-Toro, C.A., Tate, S.J., Jared, G.E.M., Swift, K.G., 2003, Complexity metrics for design (simplicity + simplicity = complexity), *Proceedings of the institution of mechanical engineers*, 217(5), pp 721-725
- Roelfsema, S., Ion, R.A., 2004, Early reliability prediction on field data, *Master thesis at Technical University Eindhoven*, Eindhoven: Technische Universiteit Eindhoven
- Shepperd, M., 1988, A Critique of Cyclomatic Complexity as Software Metrics, *Software Engineering Journal*, 3(3), pp30-36
- Verschuren, P., Doorewaard, H., 1995, *Het ontwerpen van een onderzoek*, 1<sup>st</sup> edition, Utrecht: Lemma, 1995
- Wong, K., 1990, What is wrong with existing reliability prediction methods?, *Quality and reliability engineering international*, 6, pp 251-257
- Yin, M-L., Peterson, J., Arellano, R.R., 2004, Software Complexity Factor in Software Reliability Assessment, *IEEE Proceedings of the Annual Reliability and Maintainability Symposium*, pp. 190-194
- Zhao, Y., BeMent, S.L., 1992, Kinematics, Dynamics and Control of Wheeled Mobile Robots, *in Proceedings on 1992 IEEE International Conference on Robotics and Automation, Nice, France*, pp 91-96
- Zuse, H., *Software Complexity: Measures and Methods*, New York; Walter de Gruyter & Co., 1990
- Zuse, H., 1992, Properties of Software Measures, *Software Quality Journal*, 1, pp 225-260

## Websites:

Garlikov, R., 2000, <http://www.garlikov.com/science/sciteach.htm>

Lucas, C., 2000, Quantifying Complexity Theory, CALResCo Group  
<http://www.calresco.org/lucas/quantify.htm>

Philips Healthcare intranet site – Just Another Homepage (JAH)

ReliaSoft, 2007, ReliaSoft Corporation,  
[http://www.weibull.com/LifeDataWeb/lifedataweb.htm#data\\_classification.htm](http://www.weibull.com/LifeDataWeb/lifedataweb.htm#data_classification.htm)

Sethna, J.P., Statistical Mechanics: Entropy, Order Parameters, and Complexity, *Oxford University Press 2006*,  
<http://pages.physics.cornell.edu/sethna/StatMech/EntropyOrderParametersComplexity.pdf>

Snoeyink, C., 2007, <http://craigsnoeyink.blogspot.com/2007/06/worst-death-scene-ever-goes-to-universe.html>

Telelogic Logiscope, <http://www.telelogic.com/products/logiscope/index.cfm>

TIOBE Coding Standard (TICS) analyzer,  
<http://www.tiobe.com/index.php/content/company/Home.html>

## Alternative sources:

Albers, 2008, Master Thesis Preparation, *Internal paper within the Quality & Reliability Engineering department at the Technical University Eindhoven*

Asten, van R., Product data analyst at the Customer Service department at Philips Medical Systems

Bouts, M., Project manager at the Cardio/Vascular development department at Philips Medical Systems

IEEE standard 1061-1992, Standard for a Software Quality Metrics Methodology, *The Institute of Electrical and Electronics Engineers*, December 1992

ISO/IEC 9126-1, 2001(E), Software engineering – Product quality

Kevrekidis, K., PhD student at the Quality & Reliability Engineering department at the Technical University Eindhoven.

Kevrekidis, 2007, Monitoring and Predicting the Reliability Behavior of Early Failures in the Field based on Decision Rules, *Internal PhD Research paper within the Quality & Reliability Engineering department at the Technical University Eindhoven*

Loos, M., System architect at the Cardio/Vascular development department at Philips Medical Systems

Oborzynski, K., Software developer at the Cardio/Vascular development department at Philips Medical Systems

Sonnemans, P.J.M., Researcher at the Industrial Design department at the Technical University Eindhoven.

Stollman, G., Reliability manager at the Cardio/Vascular development department at Philips Medical Systems

Telelogic AB, *Telelogic Logiscope: Audit – Basic Concepts*, Malmö: Telelogic Logiscope manual, <http://www.telelogic.com/support>, 2006

Wolvekamp, C., Product data analyst at the Customer Service department at Philips Medical Systems

# List of appendices

List of appendices.....	60
Appendix 1 System architecture .....	61
Appendix 2 Problem analysis.....	62
Appendix 3 Research outline .....	65
Appendix 4 System entropy .....	67
Appendix 5 Overview of software releases .....	69
Appendix 6 Software reliability performance .....	70
Appendix 7 Software reliability metrics .....	75
Appendix 8 Overview of physical software metrics.....	79
Appendix 9 Logiscope code metrics descriptions.....	81
Appendix 10 Code metrics for FD20 SW releases .....	84
Appendix 11 All available SW Code Metrics.....	85
Appendix 12 Reduction of code metrics.....	88
Appendix 13 Correlating code metrics with SWF .....	90
Appendix 14 regression models.....	92
Appendix 15 SW units categorized by complexity.....	99
Appendix 16 Results questionnaire .....	101
Appendix 17 Categorization of HW subsystems.....	103

## Appendix 1 System architecture

The Allura Xper CV scanner consists out of the following Software subsystems:

- User Interface (UI): software that is related to the user interface monitors that are shown to the operators
- User Interface Modules (UI Modules): all software units that are related to the user interface modules for controlling the system. Examples are: footswitch, keyboard, viewpad, etc.
- Field Service system (FS): software for configuration of system-level parameters and system-wide performance tests.
- Misc. system: software to automatically switch between acquisition and reviewing
- Acquisition: Software that is related to the tube or acquisition of the X-ray beam.
- Viewing: Software for reviewing the pictures after they are processed and stored.
- Printing and Archiving (Panda): Software that is required for printing and archiving the pictures.
- Geometry and Positioning (POS): Software that is required in order to control the moving parts of the CV scanner like the stand and the table.
- Database (DB): Database for storing settings and pictures.
- Image detection (ID): Software that is related to the flat detector which detects the X-ray beam.
- Image Processing (IP): Software that processes the detected data into pictures with optimal quality.
- Host: the host server which includes much embedded software and hosts all other software units.
- Infrastructure (Infra):
- Vx Works (VxW): contains all VxWorks support code, like for example a transport layer and a download manager.
- Operating System (OS): this subsystem provides the host Operating System, which is a standardized Windows software application.
- Field Service FrameWork (FSFW): The subsystem "FSFW" contains the Field Service FrameWork. It provides the plug-in framework, along with base classes that can be used by the subsystems for the implementation of the Field Service Component (FSC) plug-ins.

A list of all the Hardware subsystems is presented:

- X-ray Tube
- X-ray Detector
- Table (AD5/AD7)
- M-cabinet
- R-Cabinet
- Monitors
- Operator controls
- Xtravision (optional)

## Appendix 2 Problem analysis

The following phenomenon supports the statement that there are an increased number of relatively “unknown” reliability affecting factors or at least these factors cause a large variance in the system’s reliability performance, specifically in the early life phase.

### High variance in the reliability performance among systems

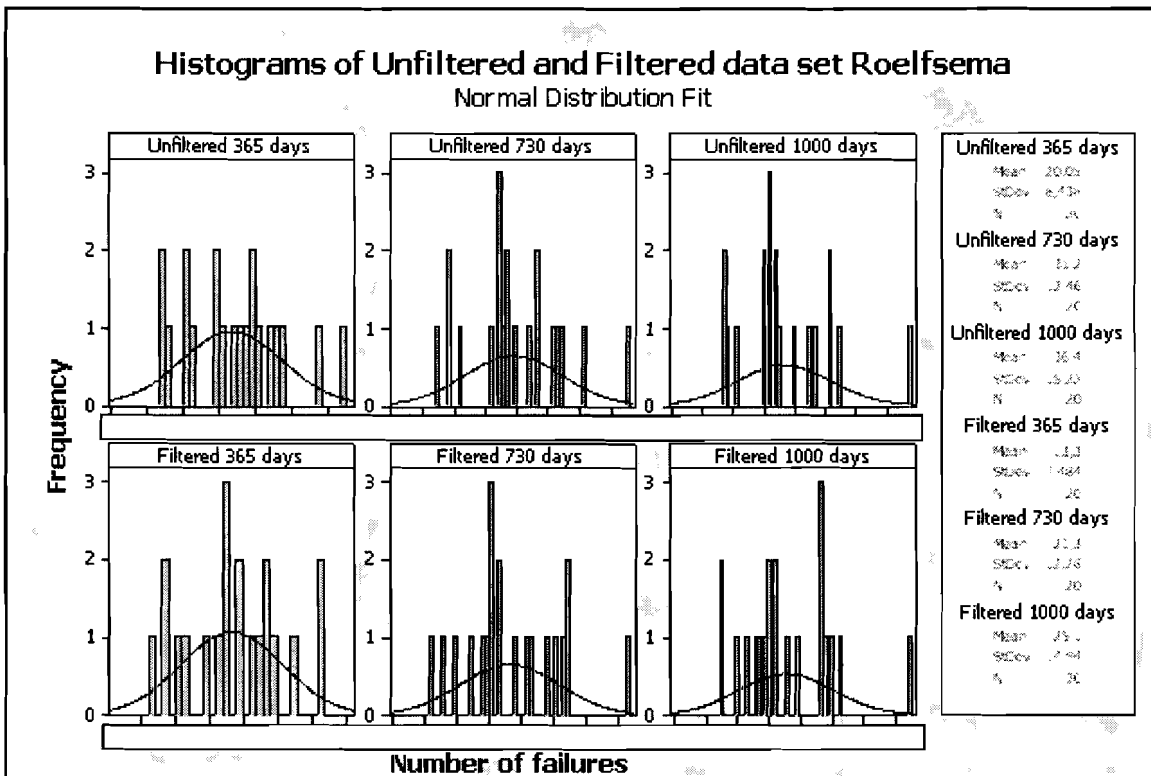
This phenomenon is based on communications with Ir. Kevrekidis, PhD student at TU/e, and on the findings of the Master thesis by Roelfsema (2004) who investigated a sample of 20 Allura Xper FD10 CV systems.

Roelfsema used “call data” from the Service Database in order to express the reliability performance of these 20 systems by the number of registered calls. Roelfsema mentions that there is a high spread in the failure patterns among the 20 systems, which indicates a high variance in the failure calls per system. When analyzing the data set (table 2.1) this indeed shows a high spread in the variance of failure calls and the data even seem uniformly distributed which even worsens the predictability of the number of failures calls for an individual system. This corresponds with Roelfsema’s conclusion that the predictive model he developed doesn’t fit individual systems.

System	365 days			730 days			1000 days		
	Failures		Diff	Failures		Diff	Failures		Diff
	Unfiltered	Filtered		Unfiltered	Filtered		Unfiltered	Filtered	
1	Confidential								
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
Av.	20	13	34%	31	21	32%	36	25	31%

Table 2.1 Failures of filtered and unfiltered data for the 20 systems (Roelfsema, 2004)

This large variance in failure calls for the sample of 20 systems is also expressed by the “uniform” distribution shown in figure 2.1. The data and conclusions from Roelfsema will be verified with all call data for the systems placed in the USA.



**Figure 2.1 Histograms from data set Roelfsema**

Roelfsema applied the following filtering principles for the raw call data:

- Filter corrective maintenance calls
- Filtering out the calls before the installation date of the system in the field
- Filtering out calls without jobs, and jobs without calls. (by job is meant a visit at the customer by a service engineer and a call means contact between the customer and the service desk)
- Filter out the Field Change Orders (specific customization of the product to the customer wishes is not seen as a failure of the system)
- Manually filter out all planned maintenance calls, installation activities and site visits incorrectly booked as corrective maintenance.

#### **Selected data set and filtering for validating the data set from Roelfsema (2004)**

At the beginning of this research there was only a single data source available, which is the service database. This service data contained the number of calls received from the customer and the number of jobs performed at the customer by the service engineer. In order to get the most representative estimate of the reliability performance of the systems, some filtering on the data is required.

First of all only systems that are placed in the USA are selected for the analysis, because:

- The calls are written in English language
- In general customers complain more frequently in the USA than in other countries.
- The USA represents roughly 50% of the total customers for CV systems

Beside this first selection of solely USA systems, the next filtering is applied:

- Only systems that contain a service contract or are within their warranty period are selected, because these customers do contact the service desk more easily because it doesn't result in extra service costs for these customers.
- Only "corrective maintenance" calls are selected because planned maintenance and installation activities calls should not be interpreted as a system failure.
- Filtering out the Field Change Orders (specific customization of the product to the customer wishes is not seen as a failure of the system)

- Filtering out calls before the installation date of the system in the field, these negative Failure times do not represent system failures because the system was not used in practice yet.

Furthermore the failure times are calculated by distracting the installation date from the call date. Therefore these failure times are expressed in days. The actual mean time between failures can't be calculated because the exact operating times were not known in the start of this Master thesis.

Next to these filtering procedures a last filtering has been applied by the Information analysts Riny van Asten (Product Data Analyst) and Cees Wolvekamp (Product Data Analyst). For each system type they selected a panel of 50 systems for which they filtered out incorrect calls. Incorrect calls are calls that resulted in a non-successful service job, which required a follow-up call and service job in order to tackle the actual root cause of the failure. Next to this filtering they established which part of the system was actually responsible for the system failure (service call).

About the sample:

- 4 sets of 50 systems: PBL10, PBL20, PBL30, PBL40
- Sample = 200 systems
- Population = 2000 systems
- Location: all systems are placed in US

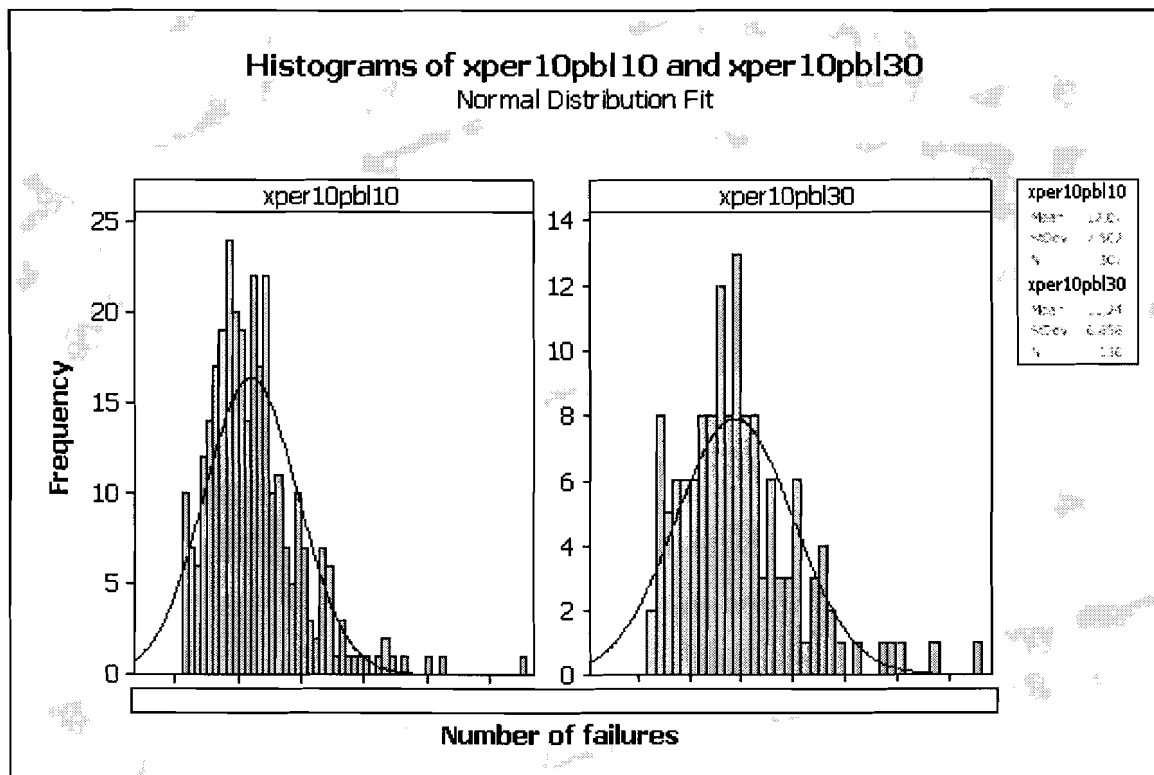


Figure 2.2 Histograms of call based failures for PBL10 and PBL30

### Conclusions:

As figure 2.2 shows, the failure calls are not uniformly distributed but rather normally distributed. Although the histograms indicate that the calls are normally distributed, the variance in failure calls differs strongly among the systems. The majority of the systems has between 0 and 15 failures during their first year after the installation date, which corresponds with the warranty period. Extended knowledge on the relatively unknown reliability factor system complexity should enable a classification of the systems according to this factor, which can explain this variance in the reliability performance. It is expected that a classification according to system complexity enables better understanding of the problems/issues that cause the high variance in reliability performance.



## Appendix 3 Research outline

The following research phases have been carried out subsequently:

### *Formulation of the research problem*

First of all reviewing the literature in order to get a better understanding of the research topic (complexity) and its implications has been carried out in the Master Thesis preparation (Albers, 2008). This literature review revealed some “blind spots” in the research field on “system complexity”. These “blind spots” were translated in a problem situation, followed by a research question and research assignment. (Paragraphs 2.1 till 2.3)

The assumptions in the problem situation (the assumed phenomena “increasing early life failures trend” and “increased variance in the reliability performance among systems”) are verified by the problem analysis which can be read in chapter 2 and appendix 2.

### *Designing the research concept:*

In order to make a concept for the research design, the research structure has to be defined and the applied study design that fulfills this research structure has to be selected. The definition of the research structure and the selected study design are described in the first two paragraphs of chapter 3. In these paragraphs the related / investigated variables are defined and decomposed into subgroups of variables first.

This research concept (in paragraph 3.1 and 3.2) is designed in order to come up with good hypotheses. These are defined in paragraph 3.3 and will provide answers to the research questions in chapter 2. These answers might contribute to more insight in the relationship between system complexity and system reliability as described in the problem situation in paragraph 2.2.

The research concept and the corresponding hypotheses are based on literature and communication in the beginning of the research. The extent to which this theoretical research concept can be followed depends heavily on the findings in the next research steps described in the chapters 4, 5, 6 and 7. The findings from these research phases and their constraints limited the hypotheses that could be tested and even slightly changed the research concept.

### *Construction of the data collection instrument*

After the problem analysis has been finished and the research design concept is defined, the method of data collection should be established. In the Master Thesis preparation Part 2, all methods of data collection were mentioned and regarding the situation at PMS the best data collection methods were selected. Not all methods were applicable, the problems and solutions regarding the data collection are discussed in the research phase 6 “Data Collection”.

### *Selection of sample*

The case study takes the Allura Xper product line as the sample for investigation. However, this product line is subdivided in several similar but not equal systems. The selection of the actual system types depends on the available data, which differs for each system type and each variable of interest. The selection of the actual system type will therefore be discussed in the variable specific chapters separately.

Next to the selection of the sample type also the sample size will depend on the available data for each system type. Again, the availability differs per system type and variable of interest and therefore will be described in the concerning chapters.

### *Create the research proposal*

The research proposal is described in chapters 2 & 3 and includes an overview of all the previous research phases and the proposal of the theoretical research concept.

### *Data collection*

This thesis report will elaborate on which data was required for each research variable. The required data selection has been separated in two types for Hardware as well as for Software. The variables Software complexity and Software reliability will be described in chapter 4 and 5. The variables Hardware complexity and Hardware reliability will be described in chapter 7. For each of the variables is described which indicators are desired in theory and which indicators can actually be collected in practice for the CV X-ray systems at PMS. Besides, these chapters also describe how these indicators and their values should be interpreted.

### *Process data*

In order to actually establish the relationship between system complexity and system reliability, the raw data has to be processed. The data should be valid and reliable in order to come to correct conclusions. This requires some processing and validation of some data (sources). This processing and validation of data (sources) is again carried out separately for: system reliability, hardware complexity and software complexity and is discussed in chapter 4 and 5.

Besides validating and processing the gained data, the relationship between the variables needs to be investigated. Therefore the correct statistical procedures should be selected and carried out properly. These activities are dealt with separately for Software in chapter 6 and Hardware in chapter 7.

### *Conclusions and Recommendations*

Finally, the conclusions and recommendations that follow from the analysis results and finding in chapter 6 and 7 are described in the last chapter 8.

## Appendix 4 System entropy

Entropy is a pure mathematical approach to measure the complexity. Entropy stands for an indicator that measures the level of disorder in a closed system. Entropy is extensively used in the field of thermodynamics. However, in this preparation master thesis the scope is limited to the relation between (maximum) entropy and reliability.

There is much confusion about what entropy really means, there are several fields that use entropy all in a different way. Generally there are 3 interpretations of entropy (Sethna, 2007):

- Entropy measures the disorder in a system
- Entropy measures the interpreters ignorance about a system
- Entropy measures the irreversible changes in a system

However, within these fields there is still criticism among researchers, in specific Physicists, (Snoeyink, 2007; Rick Garlikov, 2000) who blame each other of incorrect definitions of entropy. This research will not focus on the debate about the soundness of entropy definitions or formulas. It will only focus on applying the concept of entropy (as measuring the disorder in a system) to solve reliability issues regarding system complexity.

This focus is chosen based on Nakagawa and Yasui (2003) who state that the concept of entropy can be very useful when a measure of complexity is required to analyze the influence of complexity on reliability. The concept of entropy is already applied several times in the past on reliability problems. For example Cornacchio (1977) uses maximum entropy to classify complexity. Or Miller et al. (1984) uses maximum entropy to estimate probability distributions. Summarized, entropy is a mathematical concept with which the complexity can be quantified. But, entropy is a difficult concept and it will require the necessary training and expertise of the researcher before this concept can applied on a higher level, for example in reliability.

The concept of entropy is about “*describing the level of uncertainty that remains after influencing system characteristics are known*”. So, according to the concept of entropy a system can be in several states, these states can be separated in micro and macro states. **Micro-states** are variables *in* the system for example disorder/order, defect/running, or number of components. **Macro-states** are variables *outside* the system that can influence the system itself, for example the air quality in an operation room is a Macro-state for a medical system/device. When the entropy for complex devices with different Micro states are measured, they should be measured in the same Macro-state, in other words; *under the same circumstances*.

Finally, the entropy measures the degree to which the probability of the system being in a certain system state is distributed over the different system states. In other words; the better the actual distributions for all the Micro-states can be predicted, the higher the entropy value will be. Entropy could be used to measure complexity regarding reliability performance as well. In the following example (table 4.1), a system consist of 2 main components and can be in 3 system states. From this example one could also retrieve the reliability performance which is 80%.

<b>Components:</b>	<b>System state 1: Operating</b>	<b>System state 2: Break down C1</b>	<b>System state 3: Break down C2</b>
C1	Operating	Failure	Operating
C2	Operating	Operating	Failure
<i>Distribution:</i>	80%	15%	5%

**Table 4.1: Example of probability distributions over several systems states**

The accuracy with which one can predict the system’s state in which the system in practice will be, determines the level of entropy. The more chaos, the higher the level of entropy will be. Generally an increased number of system states also increases the chaos in the system.

The standard formula for entropy (S) in the field of statistical mechanics is  $S = k_B \ln \Omega = k_B \ln \Omega$ , with  $k_B$  as the Boltzmann constant (depending on macro-state temperature) and  $\Omega$  representing the number of the Micro-states of the system. This formula changes depending on the type of study that needs to be carried out. (Sethna, 2007) A more relevant formula for measuring the disorder of a system could be:

$H(p) = -\sum_{i=1}^m p_i \log p_i$ , with  $H(p)$  being the level of entropy,  $m$  being the maximum number of system states and  $p_i$  being the probability that the system is in state  $i$ . The entropy value is calculated by the sum of all the probabilities for each state in which the system could be. (Heylighen, 2002)

When the system state can be predicted with 100% accuracy the entropy in the system would be zero (minimum entropy). As Figure 4.1 shows, when we know for certain that the system is in state  $s_0$  we can state:  $p(s_0) = 1$ , and  $p(s \neq s_0) = 0$  which results into the following calculation;

$H(p) = -1 \cdot \log 1 + m \cdot 0 \cdot \log 0 = \log 1 = 0$ , which means minimum entropy.

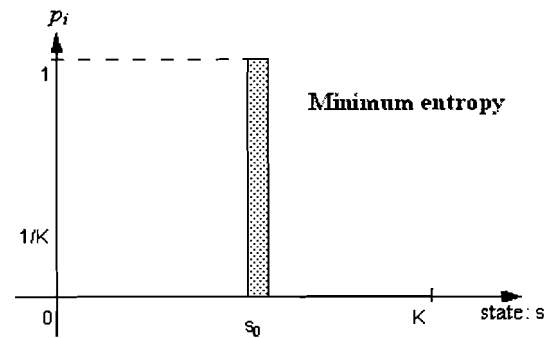


Figure 4.1: Minimum entropy [Source: Heylighen, 2002]

In contrary, when we have no clue about which state the system will be in and the distributions for each state are equal the entropy in the system is at its maximum.

$p_i = 1/K$ , with  $K$  being the total number of system states. This results in the entropy formula:  $H(p) = -K \cdot (1/K) \log(1/K) = \log K = \infty$ , which is the Maximum entropy as shown in figure 4.2. (Heylighen, 2002)

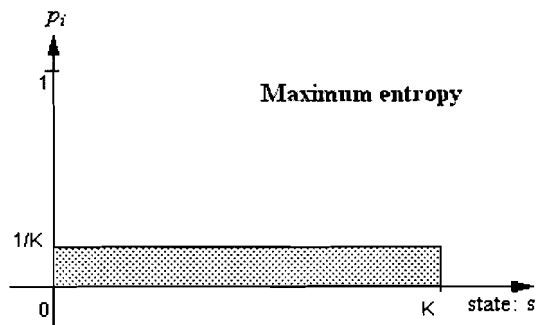


Figure 4.2: Maximum entropy [Source: Heylighen, 2002]

Generally we know in some extent which system state is more likely to be than (the) other(s) system state(s). For this case  $H(p)$  will be between zero and infinity ( $0 < H(p) < \infty$ ). The more difference there is between the probability distributions for each system state, the lower the level of entropy  $H(p)$  will be. (Heylighen, 2002) Figure 4.3 shows the intermediate entropy.

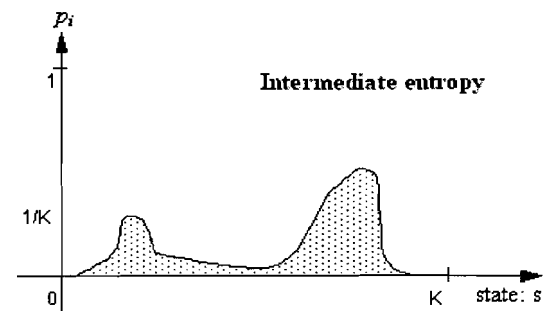


Figure 4.3: Intermediate entropy [Source: Heylighen, 2002]

### Summary and conclusion

Advantage of the concept entropy is the mathematical approach to measure complexity which will come up with quantitative results which are mathematically grounded. A disadvantage is the difficulty to apply this concept in practice. It will require many assumptions and an abstract way of thinking with accompanying limited results. Therefore entropy will be used as a metric for reliability performance, the higher the entropy level the more failure states exist and the more homogenous the numbers of failure occurrences are distributed over these failures states. In other words; the higher the entropy level, the less reliable the system will be.

## Appendix 5 Overview of software releases

PBL/ Project	Commercial Name	Software:	Service packs						
		Full Version	SP-1	SP-2	SP-3	SP-4	SP-5	SP-6	SP-7
PBL_10/ IBIS-3	Allura Xper FD10; <i>rel. 1</i>	PBL_1.2.0	PBL_1.2.1	PBL_1.2.2	PBL_1.2.3	PBL_1.2.4	PBL_1.2.5	PBL_1.2.6 <sup>1</sup>	-
PBL_23/ Rocket-A	Allura Xper FD20; <i>rel. 1</i>	PBL_2.0.0/ PDC_23.25.0	PBL_2.0.1/ PDC_23.25.1	PBL_2.0.2/ PDC_23.25.2	PBL_2.0.3/ PDC_23.25.3	PBL_2.0.4/ PDC_23.25.47	PBL_2.0.5/ PDC_23.25.53	PBL_2.0.6 <sup>1</sup> / PDC_23.25.xy	-
PBL_34/ Rocket-B	Allura Xper FD10; <i>rel. 2</i>	PBL_3.1.0/ PDC_34.6.0	PBL_3.1.1/ PDC_34.6.1	PBL_3.1.2/ PDC_34.6.2	PBL_3.1.3/ PDC_34.6.42	PBL_3.1.4/ PDC_34.6.51	PBL_3.1.5/ PDC_34.6.58	PBL_3.1.6/ PDC_34.6.66	PBL_3.1.7 <sup>1</sup> / PDC_34.6.xy
PBL_41/ Rocket-B2	Allura Xper FD20; <i>rel. 2</i>	PBL_4.0.0/ PDC_41.12.0	PBL_4.0.1/ PDC_41.13.4	-	-	-	-	-	-
PBL_43/ Rocket-B2+	Allura Xper FD20; <i>rel. 2.2</i>	PBL_4.3.0/ PDC_43.10.0	PBL_4.3.1/ PDC_43.10.14	PBL_4.3.2/ PDC_43.10.22	PBL_4.3.3/ PDC_43.10.39	PBL_4.3.4/ PDC_43.10.42	PBL_4.3.5 <sup>1</sup> / PDC_43.10.xy	-	-
PBL_51/ Rocket-C1	Allura Xper FD10; <i>rel. 3</i>	PBL_5.0.0/ PDC_51.22.0	PBL_5.0.1/ PDC_51.22.1	PBL_5.0.2/ PDC_51.22.2	PBL_5.0.3/ PDC_51.22.3	PBL_5.0.4/ PDC_51.22.4	PBL_5.0.5/ PDC_51.22.511	PBL_5.0.6/ PDC_51.22.61	-
PBL_61/ Rocket-C2	Allura Xper FD20; <i>rel. 3</i>	PBL_6.0.0/ PDC_61.18.0	PBL_6.0.1/ PDC_61.18.12	PBL_6.0.2/ PDC_61.18.29	PBL_6.0.3/ PDC_61.18.31	-	-	-	-

# Appendix 6 Software reliability performance

## 6-A Software reliability performance

First of all software reliability should be expressed in a quantitative measure. However, software reliability is rather a qualitative than a quantitative variable. There are many ways in which software reliability can be expressed and there are more methodologies to use. One way to express software reliability is the methodology provided by the ISO quality handbook, as shown in figure 6.1.

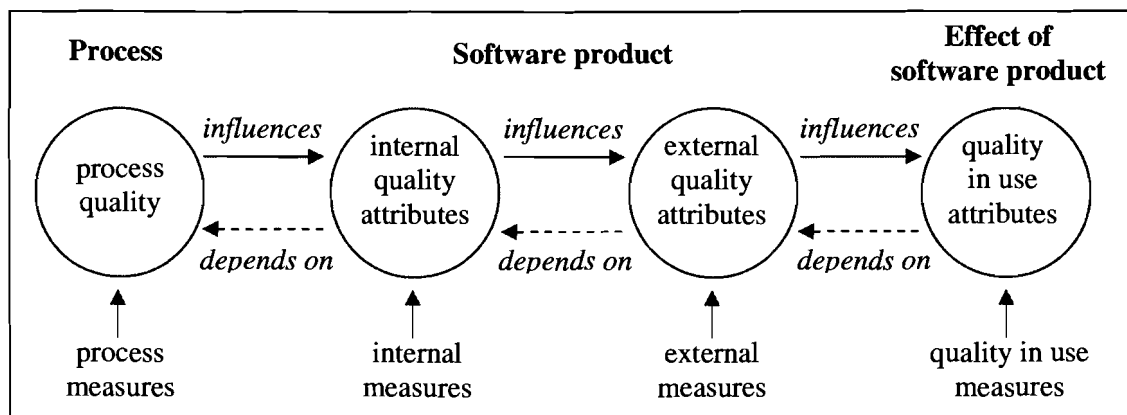


Figure 6.1 Quality model framework (ISO/IEC 9126-1:2001(E))

The model is built up out of process quality (the quality of the development process), internal quality attributes (static measures of internal software characteristics of the actual source code), external quality attributes (that measure the behavior of the code when it is executed) and quality in use attributes (quality in the user's point of view).

ISO focuses on improving the (development) process, because here the quality of the product starts. ISO assumes that an improved development/building process will result in better internal quality, this will result in better external quality, which in the end increases the quality in use. This concept is a continuous (improvement) circle which is used in order to improve the product and set up the requirements, as figure 6.1 visualizes.

This master thesis will restrict itself solely to the internal and external quality of the software product because the "quality in use" will require insight in the user profiles and their user needs. This insight is not available and is part of the PhD research by K. Kevrekidis.

According to the ISO handbook the external and internal software quality can be divided into 6 sub characteristics, which are:

- Functionality; suitability, accuracy, interoperability, security, and functionality compliance.
- Reliability; maturity, fault tolerance, recoverability and reliability compliance.
- Usability; understandability, learn ability, operability, attractiveness, usability compliance
- Efficiency; time behavior, resource utilization and efficiency compliance.
- Maintainability; analyzability, changeability, stability, testability, maintainability compliance.
- Portability; adaptability, install ability, co-existence replace ability, portability compliance.

This research solely tries to establish a relationship between the software characteristics expressed in quantitative metrics and the reliability performance of the software specifically. However, also software reliability performance is an ambiguous term and should be expressed in quantitative metrics as well in order to establish its relationship with the software complexity.

The ISO handbook describes software reliability as "the capability of the software product to maintain a specified level of performance when used under specified conditions". It also mentions that there is no wear-out or ageing in software and failures are due to faults in requirements, design and

implementation. So failures occur due to the way the software product is used in instead of the elapsed time, which is the case for hardware failures. The ISO handbook uses the following factors to express the software performance: *Maturity, Fault tolerance, Recoverability*. A detailed description of each performance criteria and the related metrics/indicators according to the ISO handbook can be found in appendix 6B. This thesis restricts itself to the fault tolerance, which means only this aspect of reliability will be taken into account.

Fenton (1996) provides the following description of Software Failure and its occurrence. A software failure occurs when there is an error made during programming and this error remains in the software even after it is tested and launched for field usage. When the system is used in the field and the functions that require the code in which the error is made are executed, the system will fail. This reasoning is best explained by the cartoons in figure 6.2. A human error during programming can lead to a fault (Software bug) which again can result in a system failure. It should be mentioned that not all errors result in Software bugs (faults) and not all software bugs cause a system failure. (Fenton, 1996)

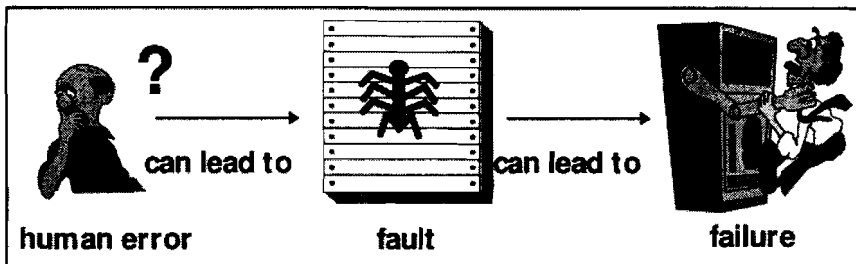


Figure 6.2 From error to failures [Fenton, 1996]

Based on this reasoning the following definitions can be made for software reliability measures (Fenton, 1996):

- *SW errors*: A processing error is a system state when it has triggered a fault but the failure has not occurred yet. These are not SW faults yet.
- *SW faults*: Occurs when a human error truly results in a mistake in the software product. The fault is the encoding of the human error and is also called a “software bug”.
- *SW failure*: is the departure of a system from its required behavior (Fenton, 1996) which corresponds with the external quality attributes and the quality in use attributes from the ISO handbook. A SWF can be discovered before system delivery (during testing) and after system delivery (during operation in field).
- *Crashes*: these are a special kind of failures where the whole system is not able to perform its function anymore.

As already mentioned before a fault does not always result into an actual failure (when the code is never executed the fault does not result in a system failure). According to Fenton (1996) SW reliability should be defined in terms of failures observed during operation rather than in terms of faults. The selection of the SW reliability measures in this thesis depends on the available and retraceable data at PMS, which will be described in the next paragraph.

Of course, not all software errors and software failures have the same severity or are the same type of errors/failures. The software errors and failures can be categorized according to fault types and fault severity.

One way to categorize software errors into error/fault types is the methodology mentioned by Neufelder (1993) who categorizes software errors in the following 5 areas:

- *Regenerated errors*: These errors are initially not in the software but are created when software was adapted in order to correct another error.
- *Undetected error*: this is a programming error that hasn't been discovered yet.

- *Performance error*: these errors actually impede the software program from executing its task; this can be separated in lost functionality or in performance like response time / memory usage / etc.
- *Data error*: these errors occur due to incorrect entered data or values
- *Initialization error*: occurs due to incorrect/incomplete initialization of variables.

Another approach to categorize the software faults/errors into classes is classification by IEEE standard 1061 (1992). This standard provides the following categorizations: Logic problem, Computational problem, Interface / timing problem, Data handling problem, Data problem, Document quality problem, and Enhancement. These categories are again subdivided into root failure causes which are shown in appendix 6C.

Next to this categorization of errors into types of software errors also the errors can be categorized according to their criticality. Neufelder (1993) categorizes errors in 5 groups:

- *Catastrophic*: these errors may cause an irreversible mission failure, or a safety hazard that may result in an injury or life threatening situation.
- *Critical*: these kinds of errors cause an unacceptable system failure that will result in long system downtime or loss of important data.
- *Moderate*: these errors result into system failures that result into short or temporary system downtime and / or some unavailable system functions.
- *Negligible*: these errors have no significant effect on the system performance, the system has to be restarted or the errors don't cause unavailable system functions.
- *All others*: all errors that cannot be categorized in the groups above or errors that are not caused by the code itself but by incorrect operator's actions or hardware failures.

## 6-B Reliability aspects defined by IEEE (standard 1061)

This part of the appendix describes the three reliability aspects for software as defined in the ISO handbook: Maturity, Fault tolerance, and Recoverability.

### B1. Maturity metrics

Internal /External	Metric	Description
external	Estimated latent fault density	No. of faults detected during a defined trial period and predict potential number of future faults using a reliability growth estimation model
external	Failure density against test cases	Count the no of detected failures and performed test cases and compare them
external	Failure resolution	Measure how many failure conditions are resolved.
external	Fault density	The number of detected faults and the fault density
external	Fault removal	The amount of faults removed during testing in comparison to the total no. of faults detected and total no. of faults predicted
external	Meant time between failures	The no. of failures occurred during a defined period of operation, the average time interval between software failures
external	Test coverage	the amount of required test cases executed during testing
internal	Fault detection	Number of faults detected in the reviewed period compared with the estimated faults to be detected
internal	Fault removal	Count the number of faults removed and detected during the design/coding and compare them
internal	Test adequacy	measures if there are enough tests runs



## B2. Fault tolerance metrics

Internal /External	Metric	Description
external	Breakdown avoidance	The number of breakdowns occurrence with respect to the no of failures, measured by the log files
external	Failure avoidance	The number of avoided fault patterns (that could cause serious and critical failures) found during testing in comparison to the number of considered fault patterns
external	Incorrect operation avoidance	the number of test cases of incorrect operations which were avoided to cause critical and serious failures and compare it to the no of executed test cases of incorrect operation patterns to be considered.
internal	Failure avoidance	The number of failure patterns that are found during design/coding
internal	Incorrect operation avoidance	The number of functions that are implemented with incorrect operations avoidance capability

## B3. Recoverability

Internal /External	Metric	description
external	Availability	Test the system in a production like environment for a specified period of time performing all user operations and measure the repair time period, each time the system is unavailable.
external	Mean down time	The mean time period in which the system is unavailable
external	Mean recovery time	The recovery time that is required after the system breaks down
external	Restart ability	The number of system restarts within a specified time period and compare this number of restarts when the system was brought down during the specified trial period
external	Restorability / Restore effectiveness	the number of successful restorations compared with its required restoration in the specifications
internal	Restorability / Restore effectiveness	The capability of the system to restore itself after an abnormal event or request and the effectiveness of the restoration.

## 6-C Software failure classification by IEEE standard 1061

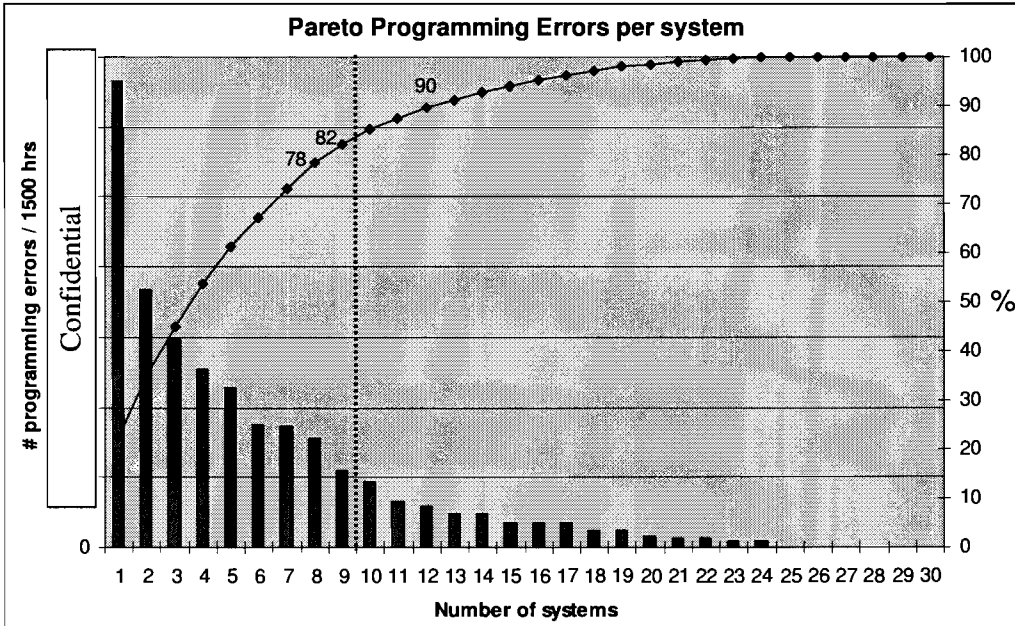
<b>Logic Problem</b>	Forgotten cases or steps	<b>Data problem</b>	Sensor data incorrect or missing
	Duplicate logic		Operator data incorrect or missing
	Extreme conditions neglected		Embedded data in tables incorrect or missing
	Unnecessary function		External data incorrect or missing
	Misinterpretation		Output data incorrect or missing
	Missing condition test		Input data incorrect or missing
	Checking wrong variable		Ambiguous statement
<b>Computational problem</b>	Iterating loop incorrectly	Incomplete item	Incorrect item
	Equation insufficient or incorrect Missing computation Operand in equation incorrect	Missing item	Conflicting items
	Operator in equation incorrect Parentheses used incorrectly	Conflicting items	Redundant items
	Precision loss Rounding or truncation fault Mixed modes	Confusing item	Illogical item
	Sign convention fault	Non-verifiable item	Unachievable item
<b>Interface / timing problem</b>	Interrupts handled incorrectly	<b>Document quality problem</b>	Applicable standards not met
	I/O timing incorrect Timing fault causes data loss		Not traceable
	Subroutine/module mismatch Wrong subroutine Incorrectly-located subroutine call Non-existent subroutine called Inconsistent subroutine arguments		Not current
			Inconsistencies
<b>Data handling problem</b>	Initialized data incorrectly	Incomplete	No identification
	Accessed or stored data incorrectly Flag or index set incorrectly Packed/unpacked data incorrectly	Enhancement	Change in program requirements Add new capability Remove unnecessary capability
	Referenced wrong data variable	Update current capability	Improve comments
	Scaling or units of data incorrect	Improve code efficiency	Implement editorial changes
	Dimensioned data incorrectly Variable type incorrect Subscripted variable incorrectly	Improve usability	Software fix of a hardware problem
	Scope of data incorrect		Other enhancement.

# Appendix 7 Software reliability metrics

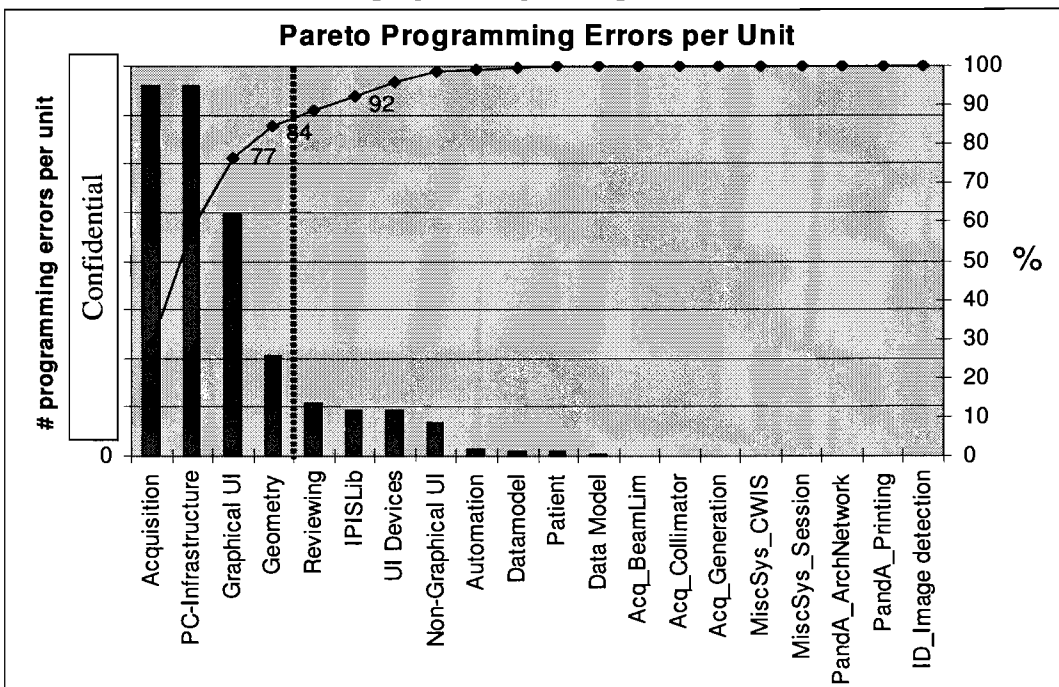
## 7-A Pareto analysis

### A.1 Field programming errors

The Pareto of the “normalized” programming errors per system:

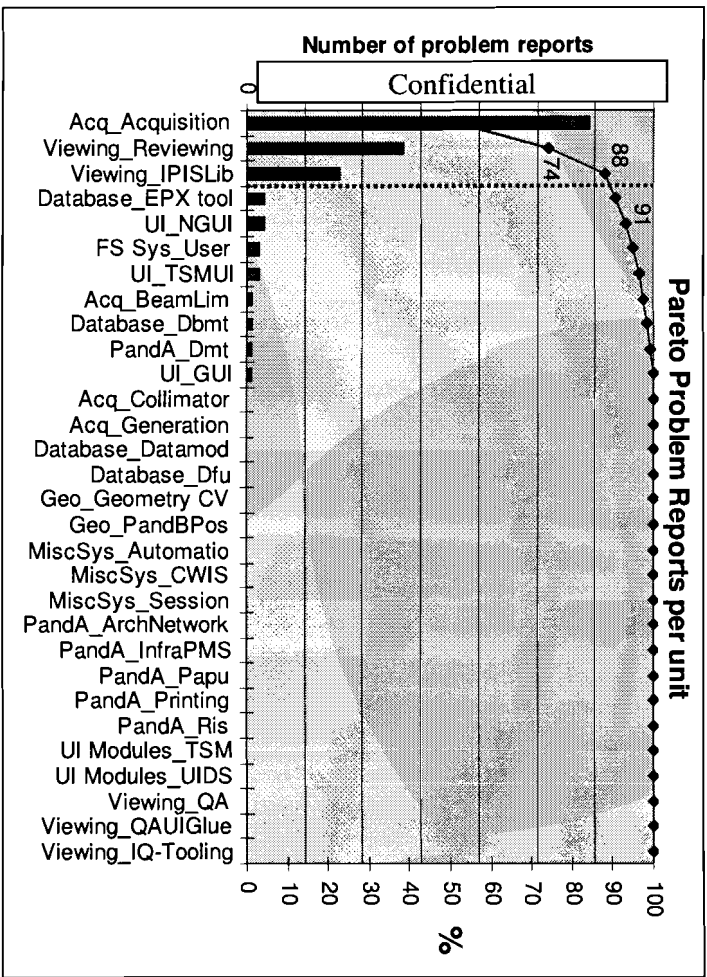


The Pareto of the “normalized” programming errors per software unit:



## A.2 Test problem reports

Pareto on Problem reports during testing per software unit



Unit	Failure states	Failure description	Frequency	p	H(p)/state	H(p)
Acquisition			152			0,203
	1	"Programming Image Detector failure"	27	0,178	0,133	
	2	"Programming XRay-Generator failure"	125	0,822	0,070	
Automation			3			0
	3	"Programming error: COM server not available."	3	1,000	0,000	
Data Model Translator			1			0
	4	"Programming error: unspecified exception."	1	1,000	0,000	
Datamodel			2			0,301
	5	"Programming error: DmCacheLib	1	0,500	0,151	
	6	"Programming error: unspecified exception."	1	0,500	0,151	
Geometry			39			0,296
	7	"Programming error: CLE1 4157 dev_cls.c#1880 p_task_can_527_1:can_int.c"	1	0,026	0,041	
	8	"Programming error: GECO 4304 b\gsc\src\gsc_tfgeco.c#1845 Emb\bb\src\dev_cls.c:p_call_tsk"	10	0,256	0,152	
	9	"Programming error: GECO 430A b\gsc\src\gsc_tfgeco.c#1845 Emb\bb\src\dev_cls.c:p_call_tsk"	28	0,718	0,103	
Graphical UI			101			0,320
	10	"Programming error: GSC:"	2	0,020	0,034	
	11	"Programming error: COM server not available."	80	0,792	0,080	
	12	"Programming error: process terminated due to a fatal exception."	1	0,010	0,020	
	13	"Programming error: starting Application failed"	10	0,099	0,099	
	14	"Programming error: unspecified exception."	8	0,079	0,087	
IPISLib			19			0
	15	"Programming error: COM method failed."	19	1,000	0,000	
Non-Graphical UI			14			0,330
	16	"Programming error: a function call failed"	10	0,714	0,104	
	17	"Programming error: COM server not available."	3	0,214	0,143	
	18	"Programming error: unspecified exception."	1	0,071	0,082	
Patient Administration			2			0
	19	"Programming error: unspecified exception."	2	1,000	0,000	
PC-Infrastructure			152			0,383
	20	"Programming error: COM server not available."	53	0,349	0,160	
	21	"Programming error: process terminated due to a fatal exception."	1	0,007	0,014	
	22	"Programming error: structured exception."	89	0,586	0,136	
	23	"Programming error: unspecified exception."	9	0,059	0,073	
Reviewing			22			0,132
	24	"Programming error: process terminated due to a fatal exception."	2	0,091	0,095	
	25	"Programming error: unspecified exception."	20	0,909	0,038	
UI Devices			19			0
	26	"Programming error: unspecified exception."	19	1,000	0,000	

7-B Entropy values:

### 7-C Problem reports during testing:

Subsystems	Rocket B2+ (PBL 43)						
	Critical	Major	Average	Minor	Enhancement	Total	Severe
<b>Acquisition</b>							
Acquisition							
BeamLim							
Collimator							
Generation							
<b>DB</b>							
DBMT							
Datamodel							
Dfu							
EPXTool							
<b>FSSys</b>							
UserCalibration							
<b>Geo/PBPos</b>							
GeometryCV							
PandBPos							
<b>Misc Sys</b>							
Automation							
CWIS							
SessionMgt							
<b>PandA</b>							
ArchNetwork							
Dmt							
InfraPMS							
PandA							
Papu							
Printing							
Ris							
<b>UI</b>							
GUI							
Ngui							
TsmUI							
<b>UIM</b>							
TSM							
Uids							
<b>Viewing</b>							
IPISLib							
QA							
QAUiGlue							
Rev							
IQ-Tooling							

Confidential

## Appendix 8 Overview of physical software metrics

The main categories, in which the quantitative (physical) software metrics are grouped, will now be explained in more detail.

### *Software science parameters:*

These metrics are derived from the concept of *Halstead's software science metrics* (1972) in which Halstead attempts to derive software attributes like implementation efforts, clarity, structure, error rates and language levels from basic metrics which are the number of unique operators/operands and the total number of operators/operands. These quantitative metrics are related to some qualitative attributes such as program size and effort. The corresponding formulas and reasoning can be found in Halstead (1972). Simpler "*program size*" metrics are: no. of lines of code (LOC), no. of comments, no. of bytes or no. of source statements (McQuaid, 1996) Furthermore, *Kolmogorov complexity* is used to express the software complexity in measures of size/length as well. (Löfgren, 1977)

### *Control-flow metrics:*

Control flow metrics are related to the order in which the statements in a program are executed. A well known and widely applied metric of this kind is McCabe's *cyclomatic complexity* metric. This metric uses the number of control paths as an indicator of software complexity and can be calculated by the following formula:  $m = e - n + 2p$ , with  $e$  is the number of edges,  $n$  is the number of nodes and  $p$  is the number of connected components (Shepperd, 1988). Other control-flow metrics are *NPATH*, *Knot count* and *Nesting level complexity* (McQuaid, 1996).

### *Data-flow metrics:*

Data-flow metrics measure the complexity that is caused by data usage. In other words, the data structure determines the software complexity. Examples of data-flow metrics are Attribute complexity of a data structure (Munson and Khoshgoftaar, 1993) and Chapin's Q measure. (McQuaid, 1996)

### *Information-flow metrics:*

This metric measures software complexity as the number of information flows between the module and its environment and is called the Fan-in Fan-out metric (Henry and Kafura, 1981).

### *Hybrid metrics:*

Hybrid metrics are metrics that combine two or more other metrics into one metric.

### *Entropy based software complexity metric:*

Harrison (1992) applied the general concept of entropy in order to measure software complexity. He

developed the Average Information Content Classification (AICC) metric:  $AICC = -\sum_{i=1}^{n_1} \frac{f_i}{N_1} \log_2 \frac{f_i}{N_1}$ ,

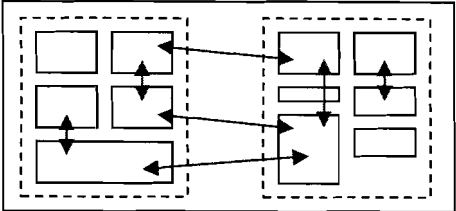
with  $f_i$  the number of occurrences of the  $i^{\text{th}}$  unique operator and  $N_1$  the total number of non-unique operators. AICC is an ordinal measure for software complexity and only can order software programs from more to less complex but no conclusions about the actual distance between the programs can be made. (Harrison, 1992)

### *Cohesion and coupling:*

In the case of large software systems / programs, object oriented software designs are generally applied to structure the program so the systems remains transparent. This is also the case for the software in the CV X-ray systems at PMS. The software is grouped into functional blocks that are connected to the core tasks of the CV systems like for example acquisition of viewing. The complexity within these subgroups can be measured by the metrics as described above and is defined as the cohesion. However, the relationships and interactions between these subgroups create complexity if

the software as well. The metric that expresses this kind of software complexity is called coupling. (Lange, 2005; Da-Wei, 2007)

Figure 8.1 shows the coupling and cohesion of the software. The blue arrows represent the *coupling* between the functional software blocks and within each module the red arrows represent the complexity of the functional block itself which is called the *cohesion*. This complexity can be measured by the interactions between the components, but also the number of components and the size of these components is a metric to express the complexity of this functional block.



**Figure 8.1 Coupling and Cohesion**  
[Derived from; Lange, 2005]



## Appendix 9 Logiscope code metrics descriptions

The descriptions of the software code metrics are derived from the Telelogic Logiscope manual (Telelogic AB, 2006) and the the research paper by Lancon and Saclay (1995).

### Quality model "Logiscope.ref" (Class metrics):

Metrics	Description / Interpretation
Number of statements	Statements that are counted; 1--> control statements (break, continue, do, for, go to, if, labels, return, switch, while, case, default) 2--> statements followed by ; , 3--> Empty statements. (Telelogic AB, 2006)
Comment frequency	number of comments / number of statements (Lancon and Saclay, 1995)
Average size of statements	Program length / number of statements = $(N1+N2)/lc\_stat$ . This metric detects components that have, on average, long statements. This generally makes the code more complex to understand and harder to analyze (Lancon and Saclay, 1995)
Cyclomatic number (VG)	This metric is measured based on the number of nodes of decisions. Formula $v = 1 + \text{Sum}(n_i - 1)$ --> $n_i$ is number of edges departing from the node $i$ . This metric quantifies the complexity of the control structure. It indicates the effort the reader must make to understand the function's algorithm and how much tests should be ran (Telelogic AB, 2006)
Maximum nesting level	maximum number of nested loops or conditions within a module (Lancon and Saclay, 1995)
Vocabulary frequency	The average number of times the vocabulary is used in a component. = $(N1+N2) / (n1+n2)$ --> when the value is high it means that it contains very similar or even duplicated statements. Good indicator of the required maintenance effort. Corrections have to be made at every place where the code has been duplicated (Telelogic AB, 2006)
Number of paths	A high value is often due to a the fact that the function has too many executable statements. Solution is to reduce the no of statements by subdividing the function or factorizing any code repetitions it contains. (code with GOTO statements cannot be analyzed) (Telelogic AB, 2006)
Number of callers	No of functions calling the designated function. This number of calls to a function is an indicator of criticalness. The more a function is called the more it is critical and the more it should be reliable (Telelogic AB, 2006)
Structuring	Unknown
Number of direct calls	The number of edges starting from a component. (Lancon and Saclay, 1995)
	Number of direct calls in a function. Different calls to the same function count for one call (Telelogic AB, 2006)
Number of parameters	The number of parameters is a good indicator for the test effort required for the function/software program; a small increase in the number of parameters can result in a significant increase in the possible combinations of execution scenarios (Telelogic AB, 2006)
	Tries to capture coupling between modules. Understanding modules with large number of parameters will require more time and effort (assumption). Modifying modules with large number of parameters likely to have side effects on other modules.

**Quality model "Logiscope.ref" (Application metrics):**

<b>Metrics</b>	<b>Description</b>
Depth of inheritance tree	Depth of the Inheritance Tree (DIT) --> no of classes in the longest inheritance link. The longer the DIT the greater the no of inherited functions and the more complex the application. (Telelogic AB, 2006)
Average complexity of functions	Weighted metrics per class (WMC), it is the sum of static complexities (ct_vg) of class methods. Metric is almost the same as ct_vg (Telelogic AB, 2006)
Call graph depth	The number of call graph levels --> the more hierarchy the more complex the system
Sum of cyclomatic numbers of the application functions	Sum of the vg's of all application functions--> sum (ct_vg) (Telelogic AB, 2006)
Number of lines	total number of lines in the function / software program (Telelogic AB, 2006)
Number of application functions	Sum of all VG for all functions in the application (Telelogic AB, 2006)
<b><i>MOOD = Metrics for Object Oriented Design:</i></b>	
Coupling factor (MOOD)	Coupling between classes within the application (software program/unit) itself. It uses the client-supplier relations, and excludes the couplings within the inheritance tree) (Telelogic AB, 2006)
Polymorphism factor POF (MOOD)	The "pof" numerators are the sum of overriding methods in all classes. This is the actual number of possible different polymorphic situations. The "pof" denominator represents the maximum nr of possible distinct polymorphic situation for that class as the sum for each class of the nr of new methods multiplied by the nr of descendants. The value will be high when all the new methods defined in each class would be overridden in all of their derived classes. (Telelogic AB, 2006)
Attribute inheritance AIF factor (MOOD)	The "aif" numerator is the sum of inherited attributes in all classes of the project. The AIF denominator is the total number of available attributes (locally defined plus inherited) for all classes. (Telelogic AB, 2006)
Method inheritance factor (MOOD)	The sum of inherited methods in all classes of the project. (Telelogic AB, 2006)
Attribute hiding factor AHF (MOOD)	The "ahf" numerator is the sum of invisibilities of all attributes defined in all classes. The invisibility of an attribute is the percentage of total classes from which this attribute is not visible. The "ahf" denominator is the total nr. of attributes defined in the project. (Telelogic AB, 2006)
Method hiding factor MHF (MOOD)	The "mhf" numerator is the sum of the invisibilities of all methods defined in all classes. The invisibility is the percentage of total classes from which this method is not visible. The "mhf" denominator is the total nr of methods defined in the project. (Telelogic AB, 2006)

**Quality model "Halstead.ref" (class metrics):**

<b>Metrics</b>	<b>Description</b>
Total number of operands	Total number of operands (Telelogic AB, 2006)
Number of distinct operands	Number of unique/distinct operands used in the code. Operands are: Literals (Decimal, Octal, Hexadecimal, Floating, Character, String, Boolean), identifiers (variable names, type names, function names, etc.), file names in #include clauses (ex: #include<stdlib.h>), Operator names (Ex: new, delete, +, -, *= and, >, +=, etc.). (Telelogic AB, 2006)
Total number of operators	Total number of operators (Telelogic AB, 2006)
Number of distinct operators	Number of unique/distinct operators used in the code. Unary operators, Binary operators, Ternary conditional operator, assignment operators, statements, declarations, declarers, classes derived classes, special member functions overloading templates, preprocessing directives, declarations and types (Telelogic AB, 2006)
Halstead intelligent content	$I=L*V$ (Telelogic AB, 2006)
Halstead mental effort	$E = V*D = (n1*N2 *(N1+N2)*\log2*(n1+n2))/(2*n2)$ (Telelogic AB, 2006)
Halstead program level	$L=(2*n2) / (n1*N2)$ (Telelogic AB, 2006)
Halstead difficulty	$D=(n1 *N2)/(2*n2)$ (Telelogic AB, 2006)
Halstead volume	$V=N*\log2(n)$ (Telelogic AB, 2006)
Halstead estimated length	$N=n1*\log2(n1)+n2*\log2(n2)$ (Telelogic AB, 2006)
Halstead length	$N= N1+N2$ (Telelogic AB, 2006)
Halstead vocabulary	$n=n1 +n2$ (Telelogic AB, 2006)

# Appendix 10 Code metrics for FD20 SW releases

The following table includes the software code complexity metrics for several service packs. The PDCs represent new software releases. PDC 41.12.0, PDC 43.10.0 and PDC 61.18.0 are new software versions (with added functionality) and the other PDCs represent service packs (updates of the software versions). The standard deviation (StDev) and the ratio “StDev/Average score” represent the degree of change in the code metrics over the PDCs.

Metrics:	PDC43.10.0	PDC43.10.13	PDC43.10.22	PDC43.10.34	PDC43.22.42	PDC61.18.1	St Dev	StDev/Avg
Depth of inheritance tree	5	5.00	5.00	5.00	5	7	0.82	0.153
Average complexity of functions	1.93	1.93	1.93	1.93	1.93	1.98	0.02	0.011
Call graph depth	26.00	26.00	26	26	26	29	1.22	0.046
Sum of cyclomatic no. of application functions	26,587.00	26,587.00	26,591.00	26,620.00	26,622.00	27,719.00	456.54	0.017
Number of lines	281,798.00	281,797.00	281,826.00	282,326.00	282,352.00	296,902.00	6081.23	0.021
Number of application functions	13,802.00	13,802.00	13,802.00	13,818.00	13,818.00	13,996.00	76.99	0.006
Coupling factor	0	0	0	0	0	0	0.00	x
Polymorphism factor	0.09	0.09	0.09	0.09	0.09	0.09	0.00	0.000
Attribute inheritance factor	0.04	0.04	0.04	0.04	0.04	0.04	0.00	0.000
Method inheritance factor	0.43	0.43	0.43	0.43	0.43	0.43	0.00	0.000
Attribute hiding factor	0.91	0.91	0.91	0.91	0.91	0.91	0.00	0.000
Method hiding factor	0.29	0.29	0.29	0.29	0.29	0.29	0.00	0.000
Number of statements	7.04	7.04	7.04	7.04	7.04	7.22	0.07	0.010
Comment frequency	4.07	4.07	4.07	4.07	4.07	3.92	0.06	0.015
Average size of statements	3.46	3.46	3.46	3.46	3.46	3.54	0.03	0.009
Cyclomatic number (VG)	1.93	1.93	1.93	1.93	1.93	1.98	0.02	0.011
Maximum nesting level	0.53	0.53	0.53	0.53	0.53	0.56	0.01	0.023
Number of distinct operands	5.62	5.62	5.62	5.63	5.63	5.78	0.06	0.011
Vocabulary frequency	1.54	1.54	1.54	1.54	1.54	1.56	0.01	0.005
Number of paths	54.3	54.3	54.3	54.28	54.28	67.31	5.31	0.094
Number of callers	0.96	0.96	0.96	0.96	0.96	1.05	0.04	0.038
Structuring	0.39	0.39	0.39	0.39	0.39	0.38	0.00	0.011
Number of direct calls	2.99	2.99	2.99	2.99	2.99	3.07	0.03	0.011
Number of parameters	0.91	0.91	0.91	0.91	0.91	0.89	0.01	0.009
Total number of operands	11.13	11.13	11.13	11.16	11.16	11.57	0.18	0.016
Number of distinct operands	5.62	5.62	5.62	5.63	5.63	5.78	0.06	0.011
Total number of operators	22.22	22.22	22.22	22.26	22.26	23.33	0.45	0.020
Number of distinct operators	9.73	9.73	9.73	9.73	9.73	9.97	0.10	0.010
Halstead intelligent content	11.91	11.91	11.91	11.92	11.92	12.49	0.24	0.020
Halstead mental effort	5,770.39	5,770.32	5,770.32	5,799.67	5,808.86	6,376.36	242.45	0.041
Halstead level	0.25	0.25	0.25	0.25	0.25	0.24	0.00	0.016
Halstead difficulty	8.88	8.88	8.88	8.89	8.89	9.18	0.12	0.014
Halstead volume	167.47	167.47	167.47	167.88	167.96	176.79	3.74	0.022
Halstead estimated length	57.71	57.71	57.71	57.79	57.8	59.87	0.87	0.015
Halstead length	33.35	33.35	33.35	33.41	33.43	34.9	0.62	0.019
Halstead vocabulary	15.35	15.35	15.35	15.36	15.36	15.76	0.17	0.011
<i>Application metrics</i>								
<i>Class metrics</i>								
<i>Class metrics (Halstead)</i>								

# Appendix 11 All available SW Code Metrics

All size, criticality and structure related code metrics for software subsystems:

Subsystem	C SI_HalsteadEstimateLength	C SI_HalsteadLength	C SI_HalsteadVolume	C SI_TotalNumberofOperands	C SI_TotalNumberofOperators	C SI_NumberofApplicationFunctions	C SI_NumberofLines	C SI_SumofVGofApplicationFunctions	C SI_CommentFrequency	C SI_NumberofParameters	C SI_NumberofStatements	C SI_LOCSubmetrics	C SI_EmptyLinesOfCode	C SI_CommentLinesOfCode	C SI_NumberOfFiles	C SI_NumberOfDirectories	C SI_NumberOfFunctions	C SI_NumberOfClasses	C SI_AddedAndModifiedCode	C SI_ChangesIn%OfUnchangedCode	C CR_NumberofCallers	C CR_NumberofDirectCalls
Acquisition	57.8	33.43	167.96	11.16	22.26	13818	292352	26622	4.07	0.91	7.04	529182	154320	434671	5392	830	18167	2953	164	0.82	0.96	2.96
Database	72.3	41.36	218	14.98	26.33	37355	830544	72412	2.08	1.14	8.68	1428193	470452	983211	5959	1054	39206	5077	117	0.07	0.95	3.53
FS Sys	48.6	28.27	138.47	9.74	16.53	783	17720	1292	6.43	0.75	6.74	140216	48515	111706	1259	297	1208	813	6	0.16	0.43	2.45
GeometryPositioning	63.4	38.6	201.39	14.1	24.44	8483	190166	17679	3.44	0.99	7.75	839765	172559	510925	4980	1197	10729	2718	253	0.30	0.94	2.45
Image Detection	65.8	37.02	192.12	13.15	23.04	7697	155788	16880	3.9	1.1	7.79	x	x	x	x	x	10484	1677	x	x	0.87	2.85
Image processing	66.8	39.3	206.07	14.59	23.76	8195	160005	17401	2.38	1.2	8	x	x	x	x	x	10860	1689	3224	4.62	0.74	2.7
Misc Sys	55.5	33.01	176.3	11.31	21.68	4244	81705	7984	3.19	0.77	6.7	185301	53101	112492	2355	433	8315	1593	29	0.12	0.83	2.59
PandA	111.5	51.2	351.93	18.26	32.86	8637	212396	18828	4.07	1.12	10.47	516490	129543	365672	4300	707	11685	2468	172	0.25	0.8	3.36
UI	47.4	27.4	130.93	9.04	16.24	18092	303136	32123	3.24	0.72	6.26	365760	111657	407121	4615	788	21011	2494	104	1.76	0.74	2.57
UI Modules	72.9	43.54	241.75	17.06	26.47	2924	85313	6245	4.15	1.02	8.81	121488	31605	73518	1158	232	3700	1032	5	2.62	0.67	2.58
Viewing	61	35.98	185.51	13.1	22.69	12100	235018	24507	3.14	0.9	8.04	412406	128604	251078	3535	709	15643	2661	8998	11.73	0.82	2.88

Subsystem	CST_HalsteadDifficulty	CST_HalsteadIntelligentContent	CST_HalsteadLevel	CST_HalsteadMentalEffort	CST_HalsteadVocabulary	CST_NumberofDistinctOperands	CST_NumberofDistinctOperators	CST_AttributeHidingFactor	CST_AttributeInheritanceFactor	CST_AverageComplexityofFunctions	CST_CallGraphDepth	CST_DepthofInheritanceTree	CST_MethodHidingFactor	CST_MethodInheritanceFactor	CST_AverageSizeofStatements	CST_CyclomaticNumberVG	CST_MaximumNestingLevel	CST_NumberofPaths	CST_Structuring	CST_VocabularyFrequency	CST_PolynomialInheritanceMOOD
Acquisition	8.89	11.92	0.25	5809	15.36	5.63	9.73	0.91	0.04	1.93	26	5	0.29	0.43	3.46	1.93	0.53	54.3	0.39	1.54	0.09
Database	10.88	13.62	0.2	10689	18.85	7.48	11.35	0.9	0.16	1.94	31	7	0.29	0.55	3.91	2.11	0.73	195	0.59	1.59	0.07
FS Sys	7.89	9.71	0.2	4480	13.47	4.95	8.52	0.96	0.06	1.65	7	7	0.26	0.57	3	1.85	0.46	1.99	0.38	1.49	0.06
GeometryPositioning	8.74	14.42	0.22	7556	16.55	6.81	9.71	0.87	0.1	2.08	16	7	0.26	0.51	3.9	2.1	0.5	65.3	0.54	1.6	0.08
Image Detection	9.46	12.05	0.23	6984	17.04	6.6	10.05	0.95	0.06	2.19	15	8	0.32	0.47	3.45	2.21	0.64	98.6	0.58	1.52	0.07
Image processing	9.56	14.95	0.25	8878	17.55	7.19	9.93	0.88	0.04	2.12	13	7	0.27	0.45	3.64	2.15	0.65	90.7	0.73	1.58	0.05
Misc Sys	7.92	11	0.27	8446	14.43	5.74	8.68	0.88	0.04	1.88	11	7	0.25	0.48	3.16	1.89	0.56	100	0.41	1.46	0.06
PandA	10.41	70.04	0.21	11579	21.07	10.14	10.9	0.93	0.07	2.18	19	7	0.29	0.51	3.39	2.19	0.66	136	0.64	1.54	0.06
UI	7.16	11.81	0.23	3823	13.59	5.05	8.48	0.85	0.11	1.78	11	7	0.33	0.41	3.47	1.78	0.42	47.6	0.67	1.47	0.06
UI Modules	9.52	13.43	0.27	10919	18	8.04	9.96	0.97	0.1	2.21	11	7	0.24	0.52	3.49	2.21	0.69	111	0.47	1.55	0.08
Viewing	9.57	12.39	0.28	9156	16.27	6.4	9.76	0.89	0.22	2.03	20	7	0.34	0.6	3.33	2.05	0.61	81	0.48	1.54	0.05

Softwareunit	CS11_CouplingFactor	CS12_NumberOfApplicationFunctions	CS13_NumberOfLines	CS14_SumOfVGofApplicationFunctions	CS15_CommentFrequency	CS16_NumberOfParameters	CS17_NumberOfStatements	CS18_HaltEstimateLength	CS19_HaltEstimateLength	CS110_HaltEstimateVolume	CS111_TotalNumberOfOperands	CS112_TotalNumberOfOperators	CS1_JAHT_TotalLines	CS1_JAH2_EmptyLines	CS1_JAHS_CommentLines
Acq_Acquisition	.00	10806	210809	19716	4.21	89	6.54	54.82	31.11	153.72	9.97	21.14	252729.0	86792.00	240704.0
Acq_BeamLim	.02	881	21092	1862	4.30	120	8.61	63.50	42.82	224.84	16.76	25.88	30487.00	6259.00	18502.00
Acq_Collimator	.01	998	20470	2108	3.83	90	7.84	55.23	33.54	169.91	12.14	21.40	66199.00	16561.00	41364.00
Acq_Generation	.02	1073	30129	2836	2.88	116	10.09	87.25	48.94	261.84	17.87	31.28	112100.0	24857.00	84319.00
Database_Datamodel	.00	10513	221505	18912	2.70	147	6.86	62.29	34.32	170.06	11.87	22.22	757833.0	211630.0	414530.0
Database_Dbmt	.01	15744	323221	28438	1.81	99	8.61	69.91	37.38	191.57	14.07	23.31	360860.0	158721.0	318105.0
Database_Dfu	.01	2280	68889	6757	1.42	130	16.01	127.81	99.57	607.41	33.82	65.66	87807.00	18507.00	29190.00
Database_EPXtool	.01	8382	167911	17883	1.63	82	8.86	73.71	40.08	209.40	14.87	25.12	200735.0	72345.00	169164.0
FS_Sys_UserCalibration	.03	347	7934	870	4.88	50	8.75	61.83	39.52	204.02	13.46	26.08	10590.00	3379.00	8689.00
Geo_GeoCvEmb	.14	349	10021	1480	1.41	163	13.84	99.15	63.14	369.37	26.83	36.21	351879.0	68861.00	184833.0
Geo_GeoDemo	.03	987	23604	3890	.44	138	13.30	104.78	75.75	499.73	30.15	45.60	21347.00	5404.00	6352.00
Geo_GeometryCV	.00	3345	78638	6691	3.87	98	7.81	66.35	39.58	208.84	14.51	25.00	159108.0	41394.00	107087.0
Geo_PandBPos	.01	2790	68669	4505	3.86	80	6.48	54.74	30.46	141.58	9.87	20.69	82049.00	25668.00	76601.00
Geo_Tools	.01	588	1846	891	1.87	111	2.25	13.34	7.88	24.55	2.38	5.40	117014.0	18563.00	62388.00
ID_CXA	.04	1101	31300	3310	3.51	100	11.81	99.53	55.89	301.89	19.93	38.08			
ID_ImageDetection	.00	5942	114125	12730	3.80	111	7.53	63.09	35.87	184.59	12.87	22.25			
IP_ImageProcessing	.01	2948	58064	5788	2.70	101	7.61	61.00	36.13	183.58	13.77	22.37			
IP_InfraVGA_Driver	.17	62	2612	225	.55	185	20.94	170.58	116.85	698.99	48.88	70.18			
IP_IS	.04	1025	24485	2613	1.95	151	9.07	92.48	50.13	264.36	17.54	28.14			
IP_SIP	.00	2854	55870	6897	1.44	132	8.62	67.17	42.27	229.37	18.26	25.74			
IP_VPC	.03	692	13030	1313	2.88	100	7.84	68.88	34.78	177.88	11.87	22.91			
MiscSys_Automation	.00	26	815	57	1.95	68	8.67	74.25	36.79	190.51	13.13	23.67	752.00	262.00	871.00
MiscSys_AutomationPlugs	.02	293	3615	403	5.25	62	4.62	31.55	17.75	78.88	5.76	12.00	14799.00	4815.00	16329.00
MiscSys_CWIS	.00	2703	45945	4493	2.58	72	5.54	48.07	28.22	148.48	9.46	18.78	122321.0	36463.00	73082.00
MiscSys_SessionMgt	.02	786	22604	2389	2.13	82	12.10	97.93	61.59	346.91	21.76	39.00	46668.00	11383.00	21036.00
PandA_ArchNetwork	.03	1364	33909	3020	3.82	105	9.59	80.88	46.57	251.53	16.09	29.48	78628.00	20584.00	68856.00
PandA_Dmt	.00	1885	51308	4330	4.53	140	11.12	101.87	59.40	337.07	22.89	36.37	98185.00	25115.00	74656.00
PandA_InfraPMS	.01	929	19463	1914	3.23	155	8.30	70.02	35.28	171.88	12.84	22.84	26371.00	9210.00	24639.00
PandA_Pap	.01	1863	40863	3627	4.30	72	7.16	47.01	29.18	154.59	9.51	19.66	88818.00	25463.00	66013.00
PandA_Printing	.03	808	14854	1344	3.41	88	7.31	60.64	30.83	157.44	10.79	20.05	42804.00	14319.00	40699.00
PandA_Ris	.02	1184	41392	3797	2.86	120	21.32	357.84	118.74	1168.77	41.11	75.83	68177.00	12702.00	49891.00
UI_Modules_TSMNT	.06	738	31385	2681	1.55	137	15.99	148.41	92.93	551.40	37.73	55.20	42973.00	13033.00	21280.00
UI_Modules_UIDS	.01	1650	24219	2942	4.38	88	6.22	48.26	27.77	143.76	10.60	17.27	60104.00	12872.00	36880.00
UI_GUI	.00	11226	188413	17868	2.27	84	5.94	46.00	26.51	124.82	8.77	17.55	202250.0	68968.00	207111.0
UI_NGUI	.01	2366	70716	7065	3.61	78	8.87	69.50	40.08	202.47	12.05	28.04	69940.00	21784.00	54832.00
UI_TSMUI	.01	4078	68644	6582	5.17	33	5.75	39.21	23.33	111.35	8.29	15.04	94288.00	32792.00	144272.0
Viewing_IPISLib	.01	5817	112864	12578	2.74	91	7.78	59.01	33.48	163.83	12.81	20.58	210856.0	67714.00	119278.0
Viewing_IQ_Tooling	.00	208	4459	471	2.33	85	10.89	77.15	54.12	278.39	22.42	31.69	5301.00	1541.00	3347.00
Viewing_QA	.00	135	1787	223	5.48	87	4.52	32.55	19.86	93.78	6.55	13.41	3946.00	981.00	3787.00
Viewing_QAUIGlue	.03	393	8572	783	2.57	99	8.89	61.21	46.44	256.41	14.40	31.04	30399.00	7869.00	13872.00
Viewing_Reviewing	.01	5111	87883	8822	3.23	88	8.49	65.36	39.25	212.62	13.87	25.29	157743.0	49347.00	106010.0

All available size &amp; criticality related code metrics for software units:

Software unit	CST1_AttributeHidingFactor	CST2_AttributeInheritanceFactor	CST3_AverageComplexityOfFunctions	CST4_CallGraphDepth	CST5_DepthOfInheritanceTree	CST6_MethodHidingFactor	CST7_MethodInheritanceFactor	CST8_AverageSizeOfStatements	CST9_CycloomaticNumberV0	CST10_MaximumNestingLevel	CST11_NumberOfPaths	CST12_Structuring	CST13_VocabularyFrequency	CST14_HalsteadDifficulty	CST15_HalsteadIntelligentContent
Acq_Acquisition	.92	.01	1.81	26	5	.29	.45	3.47	1.82	.5	47.38	.37	1.52	8.56	11.33
Acq_BeamLim	.96	.02	2.11	12	2	.20	.02	3.39	2.11	.6	48.71	.43	1.65	9.69	13.60
Acq_Collimator	.97	.46	2.11	9	9	.36	.56	3.12	2.11	.5	141.58	.40	1.54	7.66	13.58
Acq_Generation	.76	.00	2.74	10	3	.24	.14	3.73	2.74	.8	46.59	.55	1.67	12.61	14.92
Database_Datamodel	.98	.04	1.80	13	4	.22	.51	4.75	1.80	.6	45.79	.79	1.55	7.93	17.30
Database_Dbmt	.91	.17	1.81	31	9	.29	.54	3.39	2.08	.7	70.35	.49	1.53	11.09	11.36
Database_Dfu	.59	.46	2.96	28	4	.34	.27	4.73	2.97	1.1	1943.39	.47	2.23	20.80	17.00
Database_EPXtool	.93	.15	2.11	30	6	.30	.53	3.43	2.31	.8	62.84	.52	1.55	11.57	11.44
FS_Sys_User Calibration	.99	.03	1.93	7	4	.38	.23	3.30	1.93	.6	2.59	.48	1.71	9.84	11.63
Geo_GeoCvEmb	.94	.15	4.24	11	2	.27	.08	3.95	4.24	.8	262.75	.51	1.84	9.89	25.91
Geo_GeoDemo	.87	.38	3.82	13	5	.19	.32	4.19	3.82	1.0	357.99	1.44	1.97	15.25	18.14
Geo_Geometry CV	.94	.00	2.00	10	5	.30	.32	4.09	2.02	.5	31.60	.38	1.64	8.31	16.47
Geo_PandBPos	.74	.00	1.81	16	6	.29	.33	3.89	1.81	.4	1.90	.43	1.54	8.44	11.73
Geo_Tools	.61	.35	1.16	4	6	.19	.77	3.09	1.16	.2	1.16	.72	1.14	2.50	6.41
ID_CXA	.91	.00	3.01	15	4	.50	.20	3.76	3.01	1.0	168.37	.61	1.78	13.97	15.34
ID_Image detection	.95	.08	2.14	12	6	.33	.42	3.48	2.15	.6	97.03	.59	1.51	9.07	11.80
IP_Image processing	.94	.02	1.95	11	5	.30	.46	3.68	1.97	.7	72.14	.68	1.69	8.96	14.38
IP_Infra V0A Driver	.60	.00	3.63	6	2	.46	.00	4.58	3.63	1.8	624.77	.98	2.40	24.70	20.29
IP_IS	.97	.01	2.55	11	4	.33	.14	3.88	2.55	.9	23.75	.57	1.65	11.91	17.32
IP_SIP	.73	.03	2.35	13	4	.25	.06	3.73	2.37	.6	183.80	.97	1.62	9.89	13.53
IP_VPC	.99	.01	1.90	11	4	.33	.13	3.71	1.90	.6	2.83	.49	1.52	9.22	23.84
MiscSys_Automation	.00	.00	2.19	4	1	.00	.00	3.15	2.29	.9	3.92	.58	1.51	11.55	9.99
MiscSys_Automation Plugs	1.00	.00	1.39	4	2	.08	.00	2.68	1.45	.3	76.40	.31	1.30	4.44	9.38
MiscSys_CWIS	.82	.04	1.66	11	3	.24	.27	3.11	1.67	.4	41.06	.40	1.40	6.70	10.34
MiscSys_Session Mgt	.83	.00	3.04	11	2	.39	.00	3.80	3.04	1.2	360.66	.55	1.81	14.08	15.45
PandA_ArchNetwork	.92	.10	2.21	19	3	.41	.18	3.42	2.24	.8	53.40	.55	1.56	11.09	12.71
PandA_Dmt	.95	.06	2.30	13	4	.36	.59	3.79	2.30	.8	100.00	.59	1.71	13.49	14.09
PandA_InfraPMS	.86	.03	2.06	7	2	.28	.25	3.54	2.06	.8	48.48	.74	1.52	10.47	11.63
PandA_Papu	.76	.01	1.85	10	5	.27	.25	2.65	1.84	.5	120.69	.70	1.39	8.29	8.55
PandA_Printing	.98	.00	1.66	14	3	.28	.05	3.14	1.65	.5	6.73	.51	1.40	7.80	10.12
PandA_Ris	.98	.16	3.21	12	3	.26	.28	4.22	3.21	.8	517.61	.88	1.69	11.02	432.75
UI Modules_TSM NT	1.00	.01	3.63	11	2	.26	.04	4.88	3.64	1.3	413.34	.58	2.04	17.88	21.45
UI Modules_UIDS	.98	.20	1.78	10	4	.26	.40	3.06	1.78	.5	4.08	.47	1.39	6.62	11.22
UI_QUI	.68	.19	1.99	11	4	.28	.27	3.78	1.99	.4	16.27	.75	1.45	7.41	12.62
UI_NGUI	1.00	.00	3.00	8	4	.44	.17	3.48	3.01	.9	135.01	.29	1.63	9.09	14.05
UI_TSMUI	1.00	.00	1.61	11	5	.57	.57	2.68	1.62	.3	86.63	.71	1.41	5.47	9.43
Viewing_IPISLib	.96	.28	2.16	19	4	.32	.56	3.34	2.16	.8	31.18	.50	1.54	9.08	12.78
Viewing_ID_Tooling	.75	.00	2.26	9	3	.33	.00	4.04	2.26	.9	3.79	.44	1.99	12.84	16.58
Viewing_QA	1.00	.00	1.65	8	2	.46	.00	2.90	1.65	.3	1.69	.35	1.35	4.23	9.78
Viewing_QAUI glue	1.00	.00	2.04	8	3	.25	.09	3.23	2.04	.4	404.59	.49	1.63	7.99	19.76
Viewing_Reviewing	.76	.20	1.92	16	6	.43	.60	3.36	1.96	.5	127.93	.48	1.54	10.54	11.54

All available structure related code metrics for software units:

# Appendix 12 Reduction of code metrics

This appendix shows the tables with the Kendall's tau correlation test between the complexity metrics and the reliability indicators: problem reports (total and severe), field programming errors, MTBF (Based on field programming errors and total operating time).

**Kendall's tau Correlations**

		Problem Reports (Total)	Problem Reports (Severe)	Programming errors (Field)	Unique Programming Error Types	Entropy	MTBF
Structuring	Corr. Coefficient	-0,257	-0,257	-0,192	-0,155	-0,145	0,332
	Sig. (2-tailed)	0,210	0,210	0,314	0,585	0,595	0,087
	N	16	16	17	9	9	17
Halstead intelligent content	Corr. Coefficient	-0,127	-0,127	-0,058	0,402	0,493	0,017
	Sig. (2-tailed)	0,531	0,531	0,760	0,155	0,070	0,928
	N	16	16	17	9	9	17
Halstead level	Corr. Coefficient	0,299	0,299	0,017	-0,159	-0,209	-0,036
	Sig. (2-tailed)	0,154	0,154	0,930	0,581	0,452	0,856
	N	16	16	17	9	9	17
Halstead mental effort	Corr. Coefficient	-0,106	-0,106	-,390(**)	-0,093	-0,203	0,225
	Sig. (2-tailed)	0,602	0,602	0,040	0,743	0,456	0,243
	N	16	16	17	9	9	17
Number of distinct operands	Corr. Coefficient	-0,266	-0,266	-0,333	-0,031	0,087	0,148
	Sig. (2-tailed)	0,192	0,192	0,080	0,913	0,750	0,445
	N	16	16	17	9	9	17
Number of distinct operators	Corr. Coefficient	-0,085	-0,085	-0,240	-0,093	0,029	0,121
	Sig. (2-tailed)	0,676	0,676	0,205	0,743	0,915	0,529
	N	16	16	17	9	9	17
Total number of operands	Corr. Coefficient	-0,127	-0,127	-0,274	-0,093	-0,145	0,104
	Sig. (2-tailed)	0,531	0,531	0,149	0,743	0,595	0,590
	N	16	16	17	9	9	17
Total number of operators	Corr. Coefficient	-0,042	-0,042	-0,240	0,279	0,261	0,104
	Sig. (2-tailed)	0,835	0,835	0,205	0,325	0,338	0,590
	N	16	16	17	9	9	17
No of TotalLines	Corr. Coefficient	0,212	0,212	,423(**)	0,217	0,145	-,572(***)
	Sig. (2-tailed)	0,297	0,297	0,026	0,444	0,595	0,003
	N	16	16	17	9	9	17
No of EmptyLines	Corr. Coefficient	0,212	0,212	,456(**)	0,279	0,203	-,554(***)
	Sig. (2-tailed)	0,297	0,297	0,016	0,325	0,456	0,004
	N	16	16	17	9	9	17
No CommentLines	Corr. Coefficient	0,191	0,191	,456(**)	0,340	0,261	-,554(***)
	Sig. (2-tailed)	0,348	0,348	0,016	0,229	0,338	0,004
	N	16	16	17	9	9	17
No of Files	Corr. Coefficient	0,266	0,266	0,366	0,217	0,145	-,513(***)
	Sig. (2-tailed)	0,192	0,192	0,054	0,444	0,595	0,008
	N	16	16	17	9	9	17
No of Directories	Corr. Coefficient	0,312	0,312	0,277	0,356	0,333	-,385(*)
	Sig. (2-tailed)	0,129	0,129	0,148	0,220	0,233	0,047
	N	16	16	17	9	9	17

\*\* Correlation is significant at the 0.01 level (2-tailed).

\* Correlation is significant at the 0.05 level (2-tailed).



Kendall's tau Correlations

		Problem Reports (Total)	Problem Reports (Severe)	Programming errors (Field)	Unique Programming Error Types	Entropy	M TBF
Average complexity of functions	Corr. Coefficient	-0,054	-0,054	-0,242	-0,093	-0,145	0,070
	Sig. (2-tailed)	0,794	0,794	0,204	0,743	0,595	0,719
	N	16	16	17	9	9	17
Call graph depth	Corr. Coefficient	0,361	0,361	0,094	-0,063	-0,118	-0,107
	Sig. (2-tailed)	0,083	0,083	0,628	0,826	0,669	0,587
	N	16	16	17	9	9	17
Depth of inheritance tree	Corr. Coefficient	0,372	0,372	,397(*)	0,260	0,035	-,405(*)
	Sig. (2-tailed)	0,086	0,086	0,048	0,397	0,906	0,048
	N	16	16	17	9	9	17
Method inheritance factor (MOOD)	Corr. Coefficient	0,279	0,279	0,302	-0,217	-0,261	-0,350
	Sig. (2-tailed)	0,174	0,174	0,115	0,444	0,338	0,072
	N	16	16	17	9	9	17
Number of application functions	Corr. Coefficient	,424(*)	,424(*)	,456(*)	0,340	0,319	-0,295
	Sig. (2-tailed)	0,037	0,037	0,016	0,229	0,242	0,127
	N	16	16	17	9	9	17
Number of lines	Corr. Coefficient	,424(*)	,424(*)	,423(*)	0,217	0,261	-0,295
	Sig. (2-tailed)	0,037	0,037	0,026	0,444	0,338	0,127
	N	16	16	17	9	9	17
Sum of cyclomatic numbers of the application functions	Corr. Coefficient	,446(*)	,446(*)	,439(*)	0,217	0,261	-0,312
	Sig. (2-tailed)	0,028	0,028	0,020	0,444	0,338	0,106
	N	16	16	17	9	9	17
Average size of statements	Corr. Coefficient	0,042	0,042	0,133	,650(*)	,667(*)	-0,043
	Sig. (2-tailed)	0,835	0,835	0,484	0,022	0,014	0,822
	N	16	16	17	9	9	17
Cyclomatic number (VG)	Corr. Coefficient	-0,075	-0,075	-0,241	-0,093	-0,087	0,070
	Sig. (2-tailed)	0,715	0,715	0,204	0,743	0,750	0,719
	N	16	16	17	9	9	17
Maximum nesting level	Corr. Coefficient	-0,107	-0,107	-0,209	-0,464	-0,203	0,035
	Sig. (2-tailed)	0,602	0,602	0,274	0,101	0,456	0,857
	N	16	16	17	9	9	17
Number of callers	Corr. Coefficient	,403(*)	,403(*)	0,025	0,217	0,261	-0,139
	Sig. (2-tailed)	0,047	0,047	0,896	0,444	0,338	0,472
	N	16	16	17	9	9	17
Number of paths	Corr. Coefficient	0,149	0,149	-0,124	0,340	0,377	0,035
	Sig. (2-tailed)	0,465	0,465	0,512	0,229	0,167	0,857
	N	16	16	17	9	9	17
Number of statements	Corr. Coefficient	-0,106	-0,106	-,373(*)	-0,031	-0,029	0,225
	Sig. (2-tailed)	0,602	0,602	0,049	0,913	0,915	0,243
	N	16	16	17	9	9	17

\*\* . Correlation is significant at the 0.01 level (2-tailed).

\* . Correlation is significant at the 0.05 level (2-tailed).

# Appendix 13 Correlating code metrics with SWF

This appendix shows the tables with the Pearson correlation test (assuming normal distribution) between the code complexity metrics and the reliability indicators: software failures (SWF).

## Correlations between size & criticality related code metrics and SWF:

Pearson correlation test (N= 19 observations)		
Metrics:		SWF
CSI2_NumberofApplicationFunctions	Pearson Correlation	,569(*)
	Sig. (2-tailed)	0,011
CSI3_NumberofLines	Pearson Correlation	,673(**)
	Sig. (2-tailed)	0,002
CSI4_SumofVGofApplicationFunctions	Pearson Correlation	,565(*)
	Sig. (2-tailed)	0,012
CSI_JAH1_TotalLines	Pearson Correlation	,906(**)
	Sig. (2-tailed)	0,000
CSI_JAH3_CommentLines	Pearson Correlation	,833(**)
	Sig. (2-tailed)	0,000
CSI_JAH4_Files	Pearson Correlation	,666(**)
	Sig. (2-tailed)	0,002
CSI_JAH5_Directories	Pearson Correlation	0,432
	Sig. (2-tailed)	0,064
CSI1_CouplingFactor	Pearson Correlation	-0,322
	Sig. (2-tailed)	0,178
CSI5_CommentFrequency	Pearson Correlation	-0,117
	Sig. (2-tailed)	0,633
CSI6_NumberofParameters	Pearson Correlation	,563(*)
	Sig. (2-tailed)	0,012
CSI7_NumberofStatements	Pearson Correlation	-0,172
	Sig. (2-tailed)	0,481
CSI8_HalsteadEstimatedLength	Pearson Correlation	-0,045
	Sig. (2-tailed)	0,854
CSI9_HalsteadLength	Pearson Correlation	-0,106
	Sig. (2-tailed)	0,666
CSI10_HalsteadVolume	Pearson Correlation	-0,130
	Sig. (2-tailed)	0,597
CSI11_TotalNumberofOperands	Pearson Correlation	-0,138
	Sig. (2-tailed)	0,574
CSI12_TotalNumberofOperators	Pearson Correlation	-0,084
	Sig. (2-tailed)	0,732
CCR1_NumberofCallers	Pearson Correlation	0,007
	Sig. (2-tailed)	0,978
CCR2_NumberofDirectCalls	Pearson Correlation	-0,078
	Sig. (2-tailed)	0,750
**. Correlation is significant at the 0.01 level (2-tailed).		
*. Correlation is significant at the 0.05 level (2-tailed).		

**Correlations between structure related code metrics and SWF:**

<b>Pearson correlation test (N= 19 observations)</b>		
<b>Metrics:</b>		<b>SWF</b>
CST1_AttributeHidingFactor	Pearson Correlation	0,079
	Sig. (2-tailed)	0,747
CST2_AttributeInheritanceFactor	Pearson Correlation	-0,150
	Sig. (2-tailed)	0,540
CST3_AverageComplexityofFunctions	Pearson Correlation	-0,184
	Sig. (2-tailed)	0,451
CST4_CallGraphDepth	Pearson Correlation	0,170
	Sig. (2-tailed)	0,486
CST5_DepthofInheritanceTree	Pearson Correlation	0,023
	Sig. (2-tailed)	0,925
CST6_MethodHidingFactor	Pearson Correlation	-0,209
	Sig. (2-tailed)	0,391
CST7_MethodInheritanceFactor	Pearson Correlation	0,262
	Sig. (2-tailed)	0,279
CST8_AverageSizeofStatements	Pearson Correlation	,679(**)
	Sig. (2-tailed)	0,001
CST9_CyclomaticNumberVG	Pearson Correlation	-0,154
	Sig. (2-tailed)	0,529
CST10_MaximumNestingLevel	Pearson Correlation	-0,024
	Sig. (2-tailed)	0,924
CST11_NumberofPaths	Pearson Correlation	-0,105
	Sig. (2-tailed)	0,670
CST12_Structuring	Pearson Correlation	0,397
	Sig. (2-tailed)	0,092
CST13_VocabularyFrequency	Pearson Correlation	-0,022
	Sig. (2-tailed)	0,928
CST14_HalsteadDifficulty	Pearson Correlation	-0,142
	Sig. (2-tailed)	0,561
CST15_HalsteadIntelligentContent	Pearson Correlation	0,427
	Sig. (2-tailed)	0,068
CST16_HalsteadLevel	Pearson Correlation	-0,341
	Sig. (2-tailed)	0,154
CST17_HalsteadMentalEffort	Pearson Correlation	-0,181
	Sig. (2-tailed)	0,459
CST18_HalsteadVocabulary	Pearson Correlation	0,027
	Sig. (2-tailed)	0,913
CST19_NumberofDistinctOperands	Pearson Correlation	0,062
	Sig. (2-tailed)	0,800
CST20_NumberofDistinctOperators	Pearson Correlation	-0,007
	Sig. (2-tailed)	0,978
CST21_PolymorphismFactor	Pearson Correlation	-0,146
	Sig. (2-tailed)	0,551
**. Correlation is significant at the 0.01 level (2-tailed).		
*. Correlation is significant at the 0.05 level (2-tailed).		

# Appendix 14 regression models

## 14-A Single regression analyses

Data for single regression models for complexity factor “size” (F1) & FPE

SW Unit	F1 Size	F2 Structure	FPE	Normalized FPE
Acq_Acquisition				Confidential
Acq_BeamLim				
Acq_Collimator				
Acq_Generation				
Database_Datamodel				
Geo_Geometry CV				
MiscSys_Automation				
MiscSys_CWIS				
MiscSys_Session Mgt				
PandA_ArchNetwork				
PandA_Printing				
UI_GUI				
UI_NGUI				
UI Modules_UIDS				
Viewing_IPISLib				
Viewing_Reviewing				
ID_Image detection				

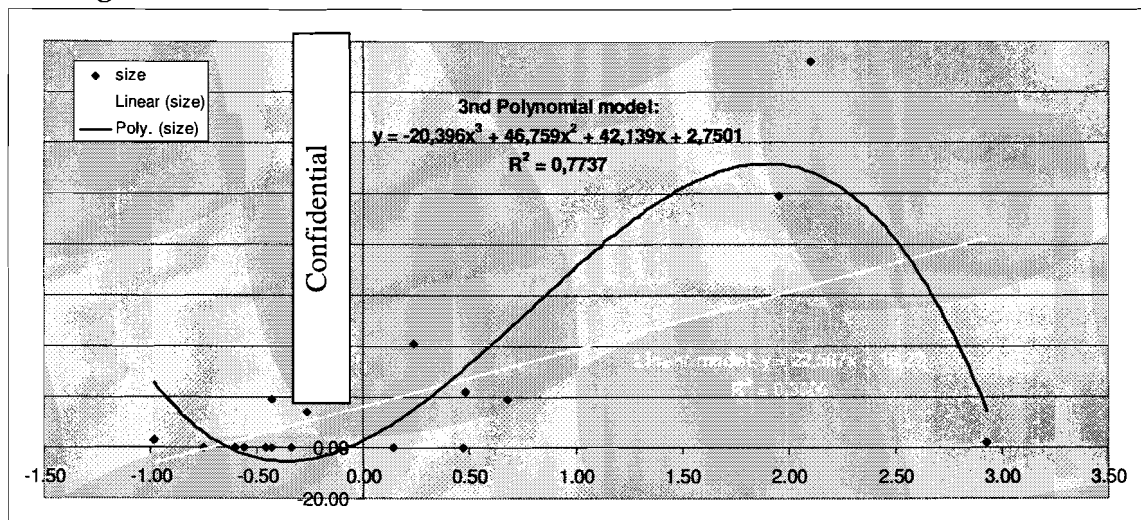
\*\* Normalized FPE = ( $\sqrt{FPE}$ )

### Single regression models for complexity factor “size” (F1) and FPE

There are several “single regression” models created in Microsoft Excel by adding linear and polynomial trend lines to the data in the scatter diagrams. The related regression formulas are shown in the scatter plots, with *y* (axis) representing FPE and *x* (axis) representing the F1 score.

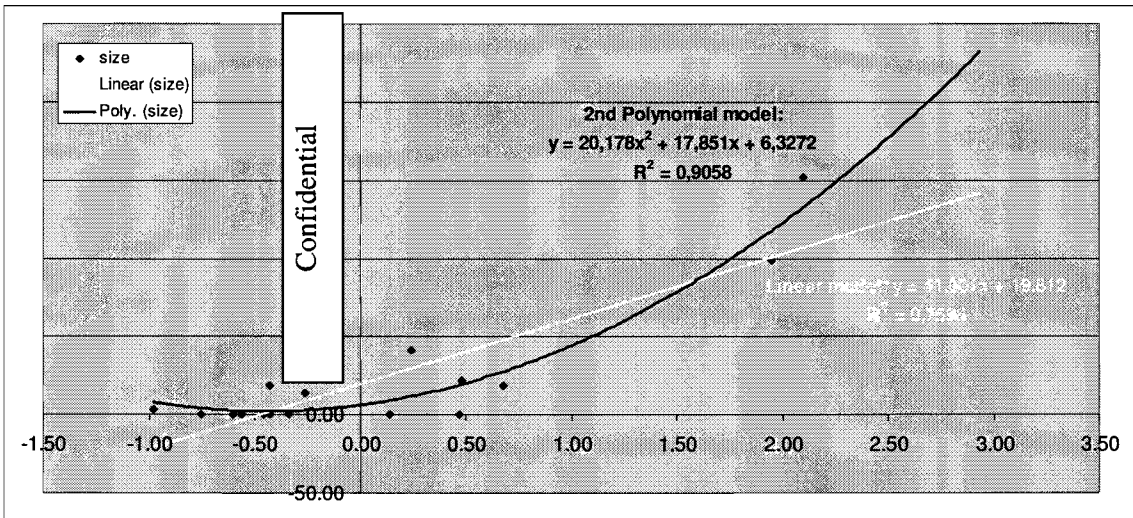
The  $R^2$ , which is calculated for both trend lines, shows how much of the variance within the FPE can be explained by the factor “size”.

### First regression model for factor F1 and FPE:



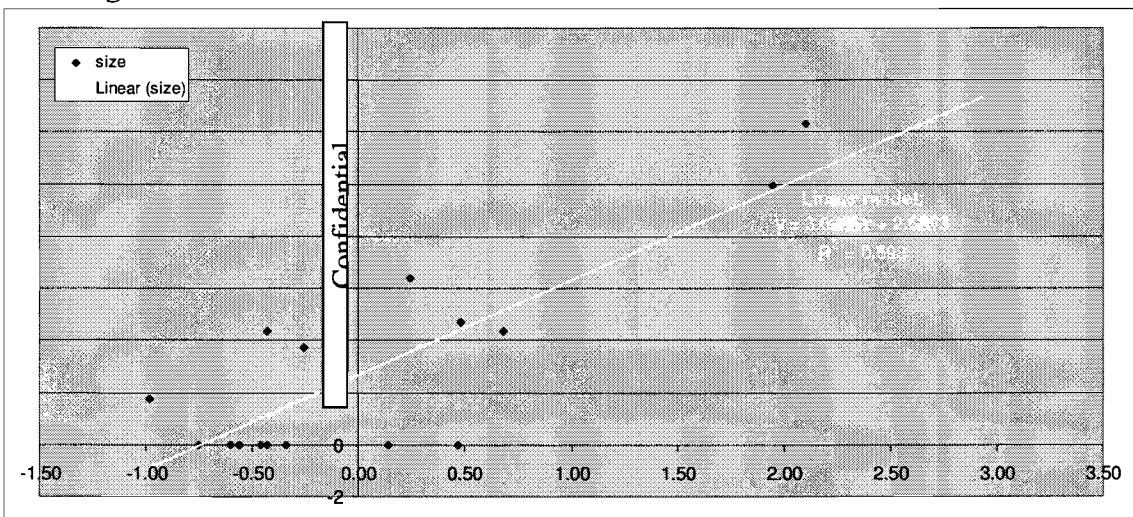
The first single regression model for F1 and FPE includes a linear trend line and a 3rd polynomial trend line. It seems that SW unit Database\_Datamodel is an extreme outlier (extreme high F1 score and rather low FPE score) and will be excluded in the next regression model.

**Second regression model for factor F1 and FPE:**



By excluding the outlier Database\_datamodel, the model fit improves for the 2<sup>nd</sup> polynomial model as well as for the linear regression model. Based on the Rsquare it can be concluded that the 2<sup>nd</sup> polynomial model gives a much better fit with the data than the linear model. In order to include F1 as a variable for the multiple linear regression model, the variable F1 should be normally distributed instead of quadratic which seems the case. In order to improve the fit of the linear regression model with the data, the FPE scores are normalized by taken the square root of the FPE scores as input variable in the following regression model.

**Third regression model for factor F1 and Normalized FPE:**



The fit of the linear regression model with the data is still not very high ( $R^2 = 0.698$ ) but it already improved a lot compared to the first regression model including the outlier Database\_datamodel and excluding the normalized FPE scores.

**Data for single regression models for complexity factor “size” (F1) & FPE**

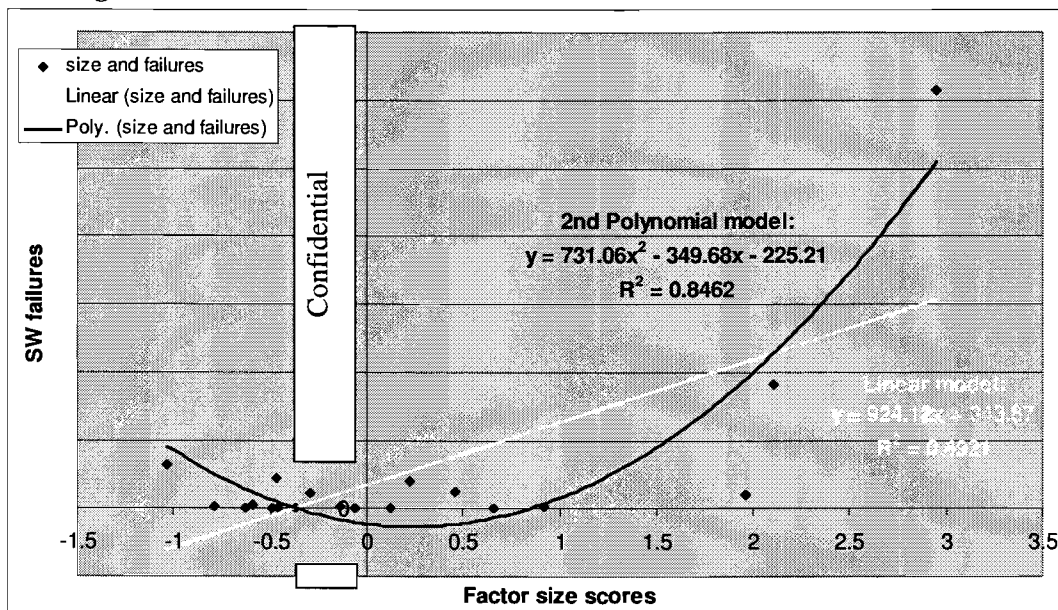
SW unit:	F1 size	F1 Structure	SW Failures	Normalized SW failures
Acq_Acquisition			Confidential	
Acq_BeamLim				
Acq_Collimator				
Acq_Generation				
Database_Datamodel				
Geo_Geometry CV				
Geo_PandBPos				
ID_Image detection				
IP_Image processing				
MiscSys_Automation				
MiscSys_CWIS				
MiscSys_Session Mgt				
PandA_ArchNetwork				
PandA_Printing				
UI Modules_UIDS				
UI_GUI				
UI_NGUI				
Viewing_IPISLib				
Viewing_Reviewing				

\*\*  $Normalized\ SW\ failures = \sqrt{SWfailures}$

**Single regression models for complexity factor “size” (F1) and SWFs**

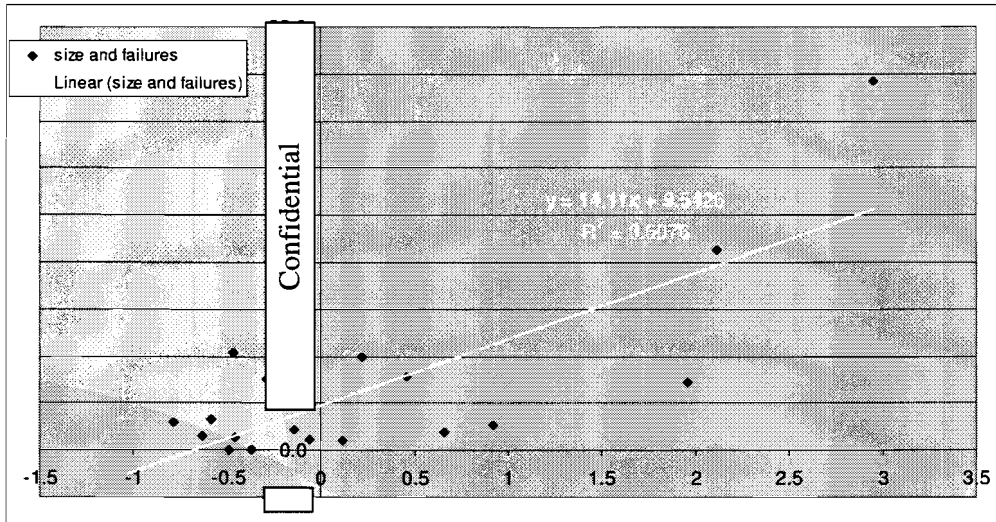
There are two “single regression” models created in Excel by adding linear and polynomial trend lines to the data in the scatter diagrams. The related regression formulas are shown in the scatter plots, with y representing SWFs and x representing the F1 score.

**First regression model for factor F1 and SWFs:**



Based on these two trend lines it shows again that there the 2<sup>nd</sup> polynomial trend line fits the data the best, because this trend line has the highest R<sup>2</sup>. This means that the 2<sup>nd</sup> polynomial model can explain the variance within the SWFs by F1 the best. In order to get more normalized SWFs, so we can use this reliability metric as a dependent variable in the multiple linear regression model, the metric SWFs is normalized by taking the square root of the SWFs. This resulted in the following linear regression model.

**Second regression model for factor F1 and SWFs:**

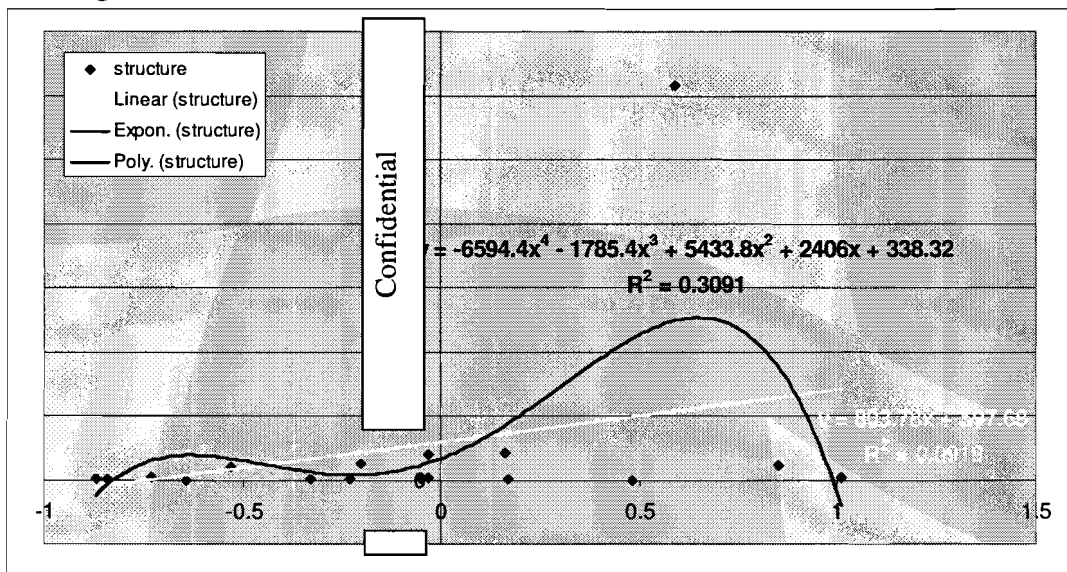


By excluding the outlier MiscSys\_Automation (extreme low score on F2 and high score on SWFs), the linear regression model was improved until a linear regression model with a fitted  $R^2 = 0.6076$ .

**Data for single regression models for the relation between complexity factor “structure” (F2) & SWFs**

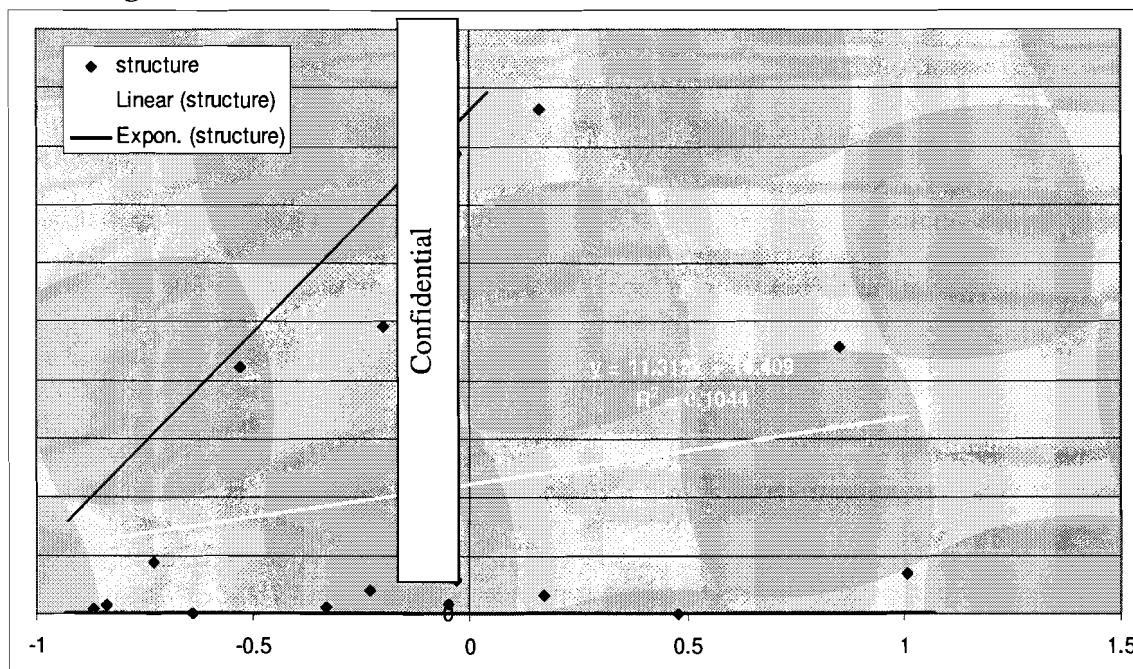
The data for this relationship is already presented in this appendix. Again the y (axis) stands for the SWFs, the x (axis) now stands for the second complexity factor “structure”.

**First regression model for factor F2 and SWFs:**



Both trend lines (polynomial and linear) show very bad fit with the data for complexity factor F2 with SWFs. Normalizing the reliability metric “SWF” is not an option now. In order to improve the linear regression model, the outliers MiscSys\_Automation and Acq\_acquisition (both with extreme high SWF scores) are excluded from the analysis and a second regression model is calculated.

## Second regression model for factor F2 and SWFs:



This simple linear regression model still doesn't fit the data well, based on this scatter plot it seems that there is not logical relation between F2 and SWFs. This corresponds with the finding in chapter 6 paragraph 6.2, that there is no significant correlation between F2 and SWFs, this also counts for the reliability metric FPE which correlates with the metric SWFs.

Although, there doesn't seem to be a linear correlation between the complexity factor structure and the reliability metrics. The multiple regression model will also include the complexity factor "structure" in the hope that this model will result in a better fit with the actual data and will result in more reliable predictions.

### Multiple regression models

The same data sets are applied as mentioned earlier in this appendix. Again the  $R^2$  stands for the degree in which the model can explain the variance within the reliability metrics (FPE, normalized FPE, SWFs and Normalized SWFs) by the complexity factors "size" and "structure". The following multiple linear regression models are made in SPSS15.0 by using an Analysis of Variance (ANOVA):

1. Dependent variable FPE with the predicting variables F1 and F2
2. Dependent variable Normalized FPE with the predicting variables F1 and F2
3. Dependent variable SWFs with the predicting variables F1 and F2
4. Dependent variable Normalized SWF with the predicting variables F1 and F2

These multiple regression models can be used in order to explain the variance in the reliability metrics by the complexity factors F1 and F2. In order to validate if these multiple regression models can also be used for predicting the reliability metrics for a certain SW unit based on its complexity scores, the data is divided into two groups. One training group on which the multiple regression models 2 and 4 are created based on the ANOVA, and one control group that is used afterwards in order to compare the predicted reliability metrics (for normalized FPE and normalized SWFs) with the actual reliability metrics.

The following interpretations and prerequisites for the ANOVA analysis should be taken into account before the results of this analysis are presented:

- The regression method "Forced entry" is selected (called the enter method in SPSS15.0), because the stepwise methods (forward, backward and stepwise) rely on the program's (SPSS15.0) selection of variables based on its mathematical criteria. It is assumed that methodological decisions are taken out of the hands of the researcher by the program itself (Field, 2005). Therefore, the forced entry method is selected in order to keep the ANOVA as basic as possible.



# Appendix 16 Results questionnaire

## 16-A Questionnaires:

### Results questionnaire rater 1:

Subsystems		Product characteristics							
		Table	M-Cabinet	Monitors	Operator Controls	X-ray Tube	R-Cabinet	X-ray Detector	Xtravision
1	Number of Hierarchic levels	3	2	1	2	3	1	1	2
2	Number of components	3	3	1	1	3	2	2	1
3	Relations between components	3	3	1	1	3	1	3	2
4	Number of design updates / changes	2	3	2	3	2	3	1	1
5	Criticality of the component	2	3	1	1	2	1	3	1
6	Technology level	2	3	1	2	2	1	3	2
7	Number of specialists involved	1	3	1	1	3	1	2	2
8	Degree of control software	3	3	1	2	3	1	2	3
Average		2,38	2,88	1,13	1,63	2,63	1,38	2,13	1,75
Rank		6	8	1	3	7	2	5	4

### Results questionnaire rater 2:

Subsystems		Product characteristics							
		Table	M-Cabinet	Monitors	Operator Controls	X-ray Tube	R-Cabinet	X-ray Detector	Xtravision
1	Number of Hierarchic levels	3	2	1	1	3	2	2	1
2	Number of components	3	2	1	1	3	2	2	1
3	Relations between components	3	1	2	2	3	1	2	1
4	Number of design updates / changes	1	3	1	1	2	3	2	1
5	Criticality of the component	2	3	1	1	2	2	3	1
6	Technology level	3	2	1	1	3	2	3	1
7	Number of specialists involved	3	2	1	1	3	2	2	1
8	Degree of control software	2	3	1	1	2	2	2	3
Average		2,50	2,25	1,13	1,13	2,63	2,00	2,25	1,25
Rank		5	4	1	1	6	3	4	2

### Average results from raters 1 and 2:

Subsystems		Product characteristics							
		Table	M-Cabinet	Monitors	Operator Controls	X-ray Tube	R-Cabinet	X-ray Detector	Xtravision
1	Number of Hierarchic levels	3	2	1	1,5	3	1,5	1,5	1,5
2	Number of components	3	2,5	1	1	3	2	2	1
3	Relations between components	3	2	1,5	1,5	3	1	2,5	1,5
4	Number of design updates / changes	1,5	3	1,5	2	2	3	1,5	1
5	Criticality of the component	2	3	1	1	2	1,5	3	1
6	Technology level	2,5	2,5	1	1,5	2,5	1,5	3	1,5
7	Number of specialists involved	2	2,5	1	1	3	1,5	2	1,5
8	Degree of control software	2,5	3	1	1,5	2,5	1,5	2	3
Average		2,44	2,56	1,13	1,38	2,63	1,69	2,19	1,50
Rank		6	7	1	2	7	4	5	3

**Overview of complexity rankings:**

	Ranking rater 1	Ranking rater 2	Average ranking	Complexity level
<b>Subsystem:</b>				
Monitors	1	1	1	Low
R-cabinets	2	1	2	Low
Operator Controls	3	2	3	Low
Xtravision	4	3	4	Low
X-ray detector	5	4	5	High
Table	6	4	6	High
X-ray tube	7	5	7	High
M-cabinet	8	6	7	High

**16-B Contingency tables and calculated chi-squares:**

**Contingency table complexity with failure classes “low” and “high”:**

		Failure classes		Total
		Low	High	
Complexity	Low	9	5	14
Classes	High	7	9	16
Total		16	14	30

**Chi-Square Tests**

	Value	df	Asymp. Sig. (2-sided)	Exact Sig. (2-sided)	Exact Sig. (1-sided)
Pearson Chi-Square	1,265 <sup>b</sup>	1	,261		
Continuity Correction <sup>a</sup>	,575	1	,448		
Likelihood Ratio	1,276	1	,259		
Fisher's Exact Test				,299	,225
Linear-by-Linear Association	1,223	1	,269		
N of Valid Cases	30				

a. Computed only for a 2x2 table

b. 0 cells (,0%) have expected count less than 5. The minimum expected count is 6,53.

**Contingency table complexity with failure classes “low”, “middle” and “high”:**

		Failure classes			Total
		Low	Middle	High	
Complexity	Low	6	4	4	14
Classes	High	6	6	4	16
Total		12	10	8	30

# Appendix 17 Categorization of HW subsystems

Unit	Complexity Class	Failures
PBL20 Table	high	
PBL20 M-Cabinet	high	
PBL20 Monitors	low	
PBL20 Operator Controls	low	
PBL20 X-ray Tube	high	
PBL20 R-Cabinet	low	
PBL20 X-ray Detector	high	
PBL20 Xtravision	low	
PBL40 Table	high	
PBL40 M-Cabinet	high	
PBL40 Monitors	low	
PBL40 Operator Controls	low	
PBL40 X-ray Tube	high	
PBL40 R-Cabinet	low	
PBL40 X-ray Detector	high	
PBL40 Xtravision	low	
PBL10 Table	high	
PBL10 M-Cabinet	high	
PBL10 Monitors	low	
PBL10 Operator Controls	low	
PBL10 X-ray Tube	high	
PBL10 R-Cabinet	low	
PBL10 X-ray Detector	high	
PBL30 Table	High	
PBL30 M-Cabinet	high	
PBL30 Monitors	Low	
PBL30 Operator Controls	Low	
PBL30 X-ray Tube	high	
PBL30 R-Cabinet	low	
PBL30 X-ray Detector	high	
	<b>Mean</b>	
	<b>SD</b>	
	<b>Variance</b>	

Unit	Complexity Class	Failures
PBL20 Table	High	
PBL20 M-Cabinet	High	
PBL20 X-ray Tube	High	
PBL20 X-ray Detector	High	
PBL40 Table	High	
PBL40 M-Cabinet	High	
PBL40 X-ray Tube	High	
PBL40 X-ray Detector	High	
PBL10 Table	High	
PBL10 M-Cabinet	High	
PBL10 X-ray Tube	High	
PBL10 X-ray Detector	High	
PBL30 Table	High	
PBL30 M-Cabinet	High	
PBL30 X-ray Tube	High	
PBL30 X-ray Detector	High	
	<b>Mean</b>	
	<b>SD</b>	
	<b>Variance</b>	
PBL20 Monitors	Low	
PBL20 Operator Controls	Low	
PBL20 R-Cabinet	Low	
PBL20 Xtravision	Low	
PBL40 Monitors	Low	
PBL40 Operator Controls	Low	
PBL40 R-Cabinet	Low	
PBL40 Xtravision	Low	
PBL10 Monitors	Low	
PBL10 Operator Controls	Low	
PBL10 R-Cabinet	Low	
PBL30 Monitors	Low	
PBL30 Operator Controls	Low	
PBL30 R-Cabinet	Low	
	<b>Mean</b>	
	<b>SD</b>	
	<b>Variance</b>	