An Evaluation of a Maintenance Model
- A comparison with theory and results from case studies

Erik Björling
Anna Hoff

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden
This thesis is submitted to the Department of Software Engineering and Computer Science at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering, specialization Management. The thesis is equivalent to 20 weeks of full time studies for two students.

Contact Information:
Authors:
Erik Björling
E-mail: erik@bjorling.com

Anna Hoff
E-mail: pt97hol@student.bth.se

External advisor:
Lars-Göran Cronberg
SchlumbergerSema AB
Phone: +46 (0) 455 32 63 00
larsgoran.cronberg@sema.se

University advisor:
Conny Johansson
Department of Software Engineering and Computer Science
conny.johansson@bth.se

Department of
Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

Internet : www.bth.se/ipd
Abstract

This master thesis was performed in cooperation with SchlumbergerSema. During the project we identified several maintenance methodologies and studied the characteristics of both the ISO and IEEE standard. The base for our evaluation of the CURE maintenance model (developed by SchlumbergerSema) were both the result of our case study that comprised interviews from five maintenance projects as well as maintenance literature available. Both the interviews and the literature studies resulted in lists of requirements that each area make on a maintenance model. We compare the CURE model to the requirements found within these two areas. Based on the result of the comparison we give our recommendations for maintenance in general, maintenance within SchlumbergerSema as well as specific recommendations for the CURE development team. Our conclusions drawn from our work were mostly positive about CURE. However we have suggested several issues for further development such as e.g. certification to a standard. Other conclusions are that no matter what model you choose as a maintenance model, make sure that you implement the model fully. A major pitfall is to allow it to become “just a fancy book on the shelf”.

Keywords:
CURE, Software Maintenance, Maintenance Models,
ISO 9000-3, IEEE Std.1219, Case Study.

Acknowledgement

We wish to send our gratitude to those who have supported us during this research:
Lars-Göran Cronberg, SchlumbergerSema, for the support, enthusiasm and contributions regarding knowledge and experience within maintenance.
Conny Johansson, Blekinge Institute of Technology, for the support, guidance and constructive criticism.
Per-Erik Rönnbäck, SchlumbergerSema, for the comments and education on CURE.
Ronneby Brunnspark and Karlskrona archipelago for the stimulating environments.
CONTENTS

1 INTRODUCTION ........................................................................................................................................... 1
  1.1 THE HISTORY OF MAINTENANCE ........................................................................................................ 1
  1.2 THE DEFINITION OF MAINTENANCE ................................................................................................... 2
  1.3 BRIEF DESCRIPTION OF SCHLUMBERGERSEMA ........................................................................... 4
  1.4 MAINTENANCE MODELS .................................................................................................................... 5
    1.4.1 The RDF model ............................................................................................................................ 5
    1.4.2 CURE ........................................................................................................................................ 8
    1.4.3 The English model .................................................................................................................... 12
  1.5 STANDARDS ......................................................................................................................................... 13
    1.5.1 IEEE ........................................................................................................................................ 13
    1.5.2 ISO ........................................................................................................................................ 15
  1.6 SUMMARY ............................................................................................................................................ 16
    1.6.1 History and definitions ............................................................................................................... 17
    1.6.2 SchlumbergerSema ................................................................................................................. 17
    1.6.3 RDF – model ............................................................................................................................ 17
    1.6.4 CURE ........................................................................................................................................ 17
    1.6.5 The English model .................................................................................................................... 17
    1.6.6 Standards ................................................................................................................................ 18

2 PROBLEM DEFINITION ............................................................................................................................... 19
  2.1 SCHLUMBERGERSEMA RELATED PROBLEMS ................................................................................ 19
  2.2 GENERAL PROBLEMS ...................................................................................................................... 19
  2.3 THE PROBLEM DEFINITION FOR THIS THESIS .............................................................................. 19
  2.4 SUMMARY OF PROBLEM DEFINITION ............................................................................................... 20

3 DESCRIPTION OF RESEARCH METHODOLOGY .................................................................................... 21
  3.1 INTRODUCTION ................................................................................................................................... 21
  3.2 PRE-STUDY .......................................................................................................................................... 21
    3.2.1 Literature studies ....................................................................................................................... 21
    3.2.2 Internal education .................................................................................................................... 21
  3.3 CASE STUDIES .................................................................................................................................. 22
    3.3.1 Background ............................................................................................................................. 22
    3.3.2 Purpose ................................................................................................................................... 22
    3.3.3 Preparations ............................................................................................................................ 23
    3.3.4 Collect data ............................................................................................................................. 23
  3.4 INTERVIEWS ..................................................................................................................................... 23
    3.4.1 The classic interview .............................................................................................................. 23
    3.4.2 Maintenance apprehension ..................................................................................................... 24
    3.4.3 Re-usability of the metrics measurement questionnaire ......................................................... 24
  3.5 ANALYZE DATA .................................................................................................................................. 24
    3.5.1 Matrix analysis ....................................................................................................................... 25
    3.6 CATEGORIZE DATA ....................................................................................................................... 25
    3.7 EVALUATING DATA ......................................................................................................................... 26
  3.8 SUMMARY OF METHODOLOGY .......................................................................................................... 27

4 RESEARCH .................................................................................................................................................. 28
  4.1 OVERVIEW OF THE CASE STUDY .................................................................................................... 28
  4.2 RESULTS/FINDINGS REGARDING GENERAL SOFTWARE MAINTENANCE ISSUES ....................... 29
    4.2.1 Identifying categories .............................................................................................................. 29
    4.2.2 Results regarding the Need for Software Maintenance .......................................................... 29
    4.2.3 Results regarding Definitions of Software Maintenance ...................................................... 31
    4.2.4 Results regarding Changes in Software Maintenance since the 1980’s .................................. 31
    4.2.5 Results regarding Apprehension about Software Maintenance ........................................... 32
    4.2.6 Results regarding CURE-knowledge ..................................................................................... 33
  4.3 RESULTS/FINDINGS REGARDING THE MAINTENANCE PROJECTS ISSUES .................................. 33
    4.3.1 Description of systems participating in the case study ............................................................ 33
6.1.6 How can maintenance models be tailored? ................................................................. 61
6.1.7 How can maintenance models be adapted (in full, as guidelines)? ....................... 62
6.2 OBJECTIVES ......................................................................................................................... 62
6.2.1 Identify existing maintenance methodologies available in the industry and document
their characteristics by comparison ............................................................................................... 62
6.2.2 Identify the possibility to tailorize, the maintenance methodologies found, for the public
sector. ......................................................................................................................................... 62
6.2.3 Identify whether the developers and the customers share the same view of software
maintenance. ............................................................................................................................... 62
6.2.4 Identify the viewpoint of software maintenance by standards organizations such as ISO. 62
6.2.5 Present a software maintenance package suitable to the market. ............................. 62
6.2.6 Present a suitable usage, tailoring and modification of CURE (SchlumbergerSemas
internal initiative of software maintenance) mainly for customers in the public sector .......... 63
6.3 RESEARCH ............................................................................................................................ 63
6.3.1 Interviews ....................................................................................................................... 63
6.4 RECOMMENDATIONS ......................................................................................................... 64
6.5 FUTURE RESEARCH .......................................................................................................... 65
6.5.1 Standards ......................................................................................................................... 66
6.5.2 Start to use a model .......................................................................................................... 66
6.5.3 Metrics ............................................................................................................................ 66
6.5.4 Securing knowledge ......................................................................................................... 66
6.5.5 Release handling .............................................................................................................. 66
7 APPENDIX A ............................................................................................................................ 67
8 REFERENCES ........................................................................................................................... 69
### 1 INTRODUCTION

The maintenance of a software system is the largest part of its lifecycle and in most cases also the most expensive. The maintenance phase begins when the software system is handed over from the software development organization to the recipient organization, but the planning of the maintenance should take place earlier.

The level of maintenance needed differs between systems and is governed by factors such as system size, complexity of the system, business criticalness of the system, number of users dependent of the system and security aspects.

A common problem is that the customer and supplier do not share the same view of software maintenance. Some customers believe that they have bought a flawless system that will run forever without any supervision or service. One significant reason for why software maintenance has to exist is that the environment around the system, both within the technical domains and also within the main business, changes over time. The system has to be updated to handle new requirements from the users and the main business. In order to have a structured and efficient handling of those system updates, a separate organization should be formed, the maintenance organization.

The maintenance organization works with all aspects of the system from the moment when the development is finished until the system is closed for usage, usually a couple of years.

The maintenance organization consists of personnel both from the supplier of maintenance services and personnel from the customer organization. The work conducted within the maintenance organization is either controlled by a maintenance model or structured individually from system to system.

On the request of SchlumbergerSema (described in subchapter 1.3) this thesis investigates maintenance models and compares them with the requirements on a maintenance model from a case study and from literature.

#### 1.1 The history of maintenance

Ever since the start of software development the work of maintenance has been present to some extent. When Royce presents the first model of a software’s lifecycle in the 70’s there were no trace of maintenance [Bergvall, p. 19]. As the awareness, about the value of the systems and amount of resources used for maintenance, increased the need for a structured way to conduct maintenance were discovered. Therefore you will be able to find maintenance as a part of the later versions of Royce’s model for the system lifecycle.

The interest for maintenance has never been high and therefore the publications within the area have been few, in relation to other related areas. In the late 70’s Lientz and Swanson presented the first Doctor of Philosophy-thesis on maintenance [Bergvall, p. 22].

Among some others, the research of Bendifallah and Scacchi (1978 – 1984) [Bergvall, p. 27] is mentioned to have contributed to the area of maintenance. Their research consisted of two case studies of two word-processing systems at two academic institutions. Through their research they managed to draw conclusions about how and why maintenance activities were conducted [Bergvall, p. 28].

During the 80’s the concept of maintenance were introduced in Sweden by RDF (RiksDataFörbundet) [Bergvall, p. 10, 23] in association with SIS (Standardiseringskommissionen i Sverige) [RDF 26:1, p. 3]. During 1980 – 1987 they also conducted a research project to develop a maintenance model that would solve some of the problems discovered when lacking a defined maintenance model. The result in 1987 was a model that was an extended version of the model presented by Lientz and Swanson in the late 70’s [Bergvall, p. 19, 22, 27]. This model presented by RDF has given birth to several
different tailorized models and can therefore be recognized as the base-model of maintenance.

During 1984 – 1987 another similar project was conducted. This time by RRV (RiksRevisionsVerket) [Bergvall, p. 31]. The project studied 20 public authorities. The findings of the project resulted in the main conclusion that routines for maintenance should be developed along with the system development. They also produced a collection of criteria for "good" maintenance [Bergvall, p. 32].

In 1988 Parikh presented a model somewhat different from the model presented by RDF. This model was completely circular. By doing this he pointed out that all the knowledge built up during the development phase, has to be take care of in the maintenance phase [Bergvall, p. 21]. Another big difference were that the RDF only studied maintenance on existing systems, as Parikh on the other hand concentrated on systems yet to be built [Bergvall, p. 21].

Finally, in 1992 the RSF (Referensgruppen för systemförvaltning) admitted a definition on maintenance [Bergvall, p. 23]:

"Maintenance is all activities conducted in order to administrate and manage a system in operation to make sure that the system during its whole lifetime efficiently contributes to fulfill the goals of the business."

When comparing the international research on maintenance to the research conducted in Sweden, we discover two directions. While the international researchers mainly studied maintenance on the actual system, the Swedish researchers concentrated more on the connection between the system and the business [Bergvall, p. 34].

1.2 The definition of maintenance

When studying books and articles about maintenance you will discover that every author has at least one definition of maintenance. Sometimes they mention even more. Although all of them are different and choose different approaches, the base of them is the same. The most significant variation of all the definitions is what phases they include within the term. Some definitions contain every activity ever conducted after the system has been put into operation. Some say that improvements of the system belong to development rather than maintenance. Others define maintenance restricted to only containing corrections of the system. One common thing is that the process of maintenance is iterative [Bergvall, p. 10]. Some representative definitions:

Bergvall and Welander: "The work of continuously changing and managing information systems in the purpose of securing the benefit of the system within the business" [Bergvall, p. 18].

Leintz and Swanson: Adjustments, improvements and corrections [Bergvall, p. 22].

RDF (RiksDataFörbundet) had the same definition as Leinzt and Swanson, but they added the term of sanitation [Bergvall, p. 22].

RSFs (Referensgruppen för SystemFörvaltning) definition is mentioned in subchapter 1.1.

Parikh listed a lot of activities and in the end of the list he added etc. His definition was much like the one stated by the RSF [Bergvall, p. 23].

IEEE: "Modification of a software product after delivery, to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment" [Takang, p. 3].

According to RDF [RDF 26:1, p. 10] the first definitions made were mostly concerned with fault corrections. Later on the concept of improvements were added to most definitions. If
you study the Swedish terms of maintenance you will find big changes as new definitions were made.

Both RDF and Bergvall agree that an exact definition of maintenance might not be that important [RDF 26:1, p. 18][Bergvall, p. 16]. They both found it more important that you have a clear definition agreed upon within the organization. However, RDF talks a lot about the importance of spreading the definition. That is why they think it would be better to define maintenance by using simpler and unambiguous words than defining it exactly. They hope that it will help the concept of maintenance to gain better appreciation and being more spread within businesses.

Since most definitions of software maintenance are quite similar, we wanted to find one that we felt would be the best instead of making our own. Therefore our definition of maintenance is the same as the one developed by RSF (Referensgruppen för SystemFörvaltning) and also used in CURE:

“Maintenance is all the activities performed in order to administrate and manage a system in operation, in a matter that during its lifetime it efficiently contributes to achieve the goals of the main business.” [CURE]

The reason for us deciding to select this definition is that it contains all the parts we wanted to put within the definition. It is also written in a simple language and it is easy to understand. We also feel that the definition also states why software maintenance is necessary.
1.3 Brief description of SchlumbergerSema

This description of SchlumbergerSema is a (by us) shortened version of the corporate presentation available to the public at the corporate website [SchlumbergerSema].

SchlumbergerSema has more than 30,000 employees in 65 countries. In the Scandinavian countries SchlumbergerSema has about 2,100 employees in 30 cities.

SchlumbergerSema is focused in the areas IT-consultant services, system integration, managed services, computer and business security and IP-network services. Their customers are found within the sectors telecom, energy, finance, transport and public sector.

SchlumbergerSema is one business segment within the corporation Schlumberger Limited with more than 80,000 employees working in about 100 countries. Schlumberger has other business segment such as products and services for the oil and energy markets. Schlumberger Limited has built a strong corporate foundation on three key values: people, technology and quality, and profit.

SchlumbergerSema had revenue of more than $3 billion in 2001.

Prior to becoming SchlumbergerSema in 2001, Sema Group as it was called was a British-French corporation. Prior to Sema Group the company had its roots in a Swedish public sector IT-service company called Stadskonsult and DAFA. The company managed consulting, software development and software maintenance for Swedish organizations in the public sector.

SchlumbergerSema strives to be the one and only supplier of IT-services that the customer will need. SchlumbergerSema works with large, complex and critical systems involving the latest software, hardware and network solutions.

One of the latest contracts won by SchlumbergerSema is as provider of IT-systems for the Olympic Games spanning from 2002-2008.
1.4 Maintenance models

In this subchapter a summary of several maintenance models are found. The maintenance models described in this subchapter are the ones that we have studied and referred to during our case study.

1.4.1 The RDF model

The model presented in 1987 by RDF is divided into eight minor models. The different parts are related to each other to a various extent. They are therefore ordered into three groups within which the relations are stronger [RDF 26:1, p. 23]. The first group is the central models that consist of System model (1), Life cycle (2) and decision model (3). The second group is called complementary models and consists of maintenance activities (4) and roles (5). The last group is the system concepts that contain staff categories (6), system hierarchy (7) and function types (8). All the groups and their part models are illustrated in figure 1.1 below [RDF 26:1, p. 27].

![Figure 1.1 - The eight parts of the RDF maintenance model.](image)

<table>
<thead>
<tr>
<th>System layer</th>
<th>System level (system terms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Information usage</td>
<td>Main business</td>
</tr>
<tr>
<td>2. Manual data administration</td>
<td>Information system</td>
</tr>
<tr>
<td>3. Usage software</td>
<td>Databases</td>
</tr>
<tr>
<td>4. General support software</td>
<td>Surrounding software</td>
</tr>
<tr>
<td>5. System software</td>
<td>Computer system</td>
</tr>
<tr>
<td>6. Computer hardware</td>
<td>Computer equipment</td>
</tr>
</tbody>
</table>

*Table 1.1 – Description of the system model*

1.4.1.1 System model

The system model is a description of the system to be maintained and the business surrounding it. It is structured in six layers. The lowest numbered layer is the one closest to the business (information usage) and the highest (computer hardware) is closest to the hardware. For a layer to be efficient enough the layer just below has to work. The lowest numbered layer is the one most important for the efficiency of the business.

The six layers are: Information usage (1), manual data administration (2), usage software (3), general support software (4), system software (5) and computer hardware (6).

[RDF 26:1 chapter 4]
1.4.1.2 Lifecycle

The horizontal line of development, maintenance and closure is the main line of a systems lifecycle. The vertical line of improvement, maintenance and operation is an iterative line. The system will probably spend most of its life in one of these three last states. The differences between these three states are somewhat floating, but the line is drawn depending on the size of the activities. In the operation stage the activities consists mostly of minor corrections and daily tasks. Other activities like major corrections and minor improvements are conducted in the maintenance state and finally major improvements are done in the improvement state.

There are also differences in who is conducting the activities. In operation it is the system administrator and in maintenance, the maintenance group and in improvements there are developers.

[RDF 26:1, chapter 5]

1.4.1.3 Decision model

The purpose of this decision model is not that it has to be followed in every detail. For minor changes it might be easier to skip some of the steps since it would otherwise be a lot of
unnecessary work. The model is made to be a general model to fit most kinds of systems and businesses. It consists of seven iterative steps (the system is not considered to be a step, rather an object). Each step might be conducted by different or the same person as long as it is the “right” person. It is important that a person with the right knowledge and authorities does each step since the output from one step is the input to the next.

[RDF 26:1, chapter 7]

1.4.1.4 Maintenance activities

Maintenance was originally only conducted within layer three to six (see table 1.1) but has today spread to consist all levels. There are four types of maintenance activities: corrections, adjustments, improvements and sanitation.

Correction activities consist of fault corrections in the system and it is the activity that is the most obviously necessary.

Adjustment activities can be of two types. The first is the kind of adjustments conducted upon the system to fit changes within the business. The other kind is adjustments of the business to fit the system usage. Both are conducted to enable the business to gain more from the usage of the system.

Improvements are somewhat like adjustments. These activities are conducted to improve the system by further development. Its goals are to make the system fit and effective for the business.

Finally, sanitation. This activity concerns the system. It identifies unnecessary parts and functions of the system and removes them in order to make the system more fit and efficient.

[RDF 26:1, chapter 6]

1.4.1.5 Roles

RDF defines four hierarchically ordered roles. In the top layer (general business) we find the system owner. His or hers main responsibility is to make decisions about the life of the system. He or she also defines the framework for the maintenance concerning budget. Directly beneath him there is a system maintainer (preferably from the company using the system). His or her job is to make decisions within the frames set by the system owner. He or she also has to provide the system owner with qualified information for him or her to base his or her decisions upon. Beneath the system responsible there are two roles: system administrator and technical system administrator. These two roles execute the decisions made by the system responsible.

[RDF 26:1, chapter 8]

1.4.1.6 Staff categories

In this part of the model, different staff categories are defined. They are tightly connected to the layers described in the system model (subchapter 1.4.1.1). They are divided into two groups: user-staff and system-staff. Within the first group we find four kinds of users. They all use the system in their own way. Each type of user is connected to one of the layers (1-4) in the system model. The second group, the system-staff, is those who maintain the system. They are divided into 2 layers. Both connected to one of the two remaining layers (5-6) within the system model.

[RDF 26:1, chapter 9]
1.4.1.7 System hierarchy
This hierarchy that consist of four levels is most generally defined. That is why it will fit almost any system or business, despite size or technology. The four levels are total-system, system area, system and part-system. The total-system term is a collective term for all systems within a business. The system area term means all systems that are related to each other in some way. The system term is a basic term for a specific system and a part-system is a separate part of an existing system.

[RDF 26:1, chapter 10]

1.4.1.8 Function types
In the RDF-model an alternative view of the system model is presented in the form of function types. The purpose for this is that every activity within maintenance shall be able to fit in one of those types.

<table>
<thead>
<tr>
<th>Function types</th>
<th>System layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information usage</td>
<td>Information usage</td>
</tr>
<tr>
<td>Manual data administration</td>
<td>Manual data administration</td>
</tr>
<tr>
<td>Data administration</td>
<td>Usage software</td>
</tr>
<tr>
<td>1. Data entry</td>
<td>Surrounding software</td>
</tr>
<tr>
<td>2. Data storage</td>
<td>System software</td>
</tr>
<tr>
<td>3. Data transfer</td>
<td>Computer equipment</td>
</tr>
<tr>
<td>4. User interface</td>
<td></td>
</tr>
<tr>
<td>Computer operation</td>
<td>Usage software</td>
</tr>
<tr>
<td></td>
<td>Surrounding software</td>
</tr>
<tr>
<td></td>
<td>System software</td>
</tr>
<tr>
<td></td>
<td>Computer equipment</td>
</tr>
</tbody>
</table>

Table 1.2 – Description of function types

[RDF 26:1, chapter 10]

1.4.2 CURE
The reference material for this subchapter is the CURE maintenance model documentation and internal education material from SchlumbergerSema.

1.4.2.1 Purpose
The main purpose for the development of CURE (Complete, Useful, Rational and Efficient) is to bring order to maintenance. The model is developed by SchlumbergerSema (North of Sweden). Before this model was developed and brought into use, ad-hoc solutions were made almost every time a new maintenance deal was made. To ease the startup of a new maintenance project CURE provides guidelines and checklists. It clarifies roles and responsibilities as well as it provide routines for change request handling.
1.4.2.2 Lifecycle
The lifecycle of the CURE model comprises all stages from product delivery to system termination. Although system termination is not specified within CURE.

![CURE Lifecycle Diagram](image)

As the picture above shows, CURE considers the total lifecycle of a system to be circular. CURE also divides the two first steps after development into several more detailed parts described below.

Business evolvement is a process to determine the current state of the business that the system is meant to support and what the future might hold in terms of changes or stability of the current state. Then it is time to actually develop the system fit to prior analysis of the business.

1.4.2.3 Transition (Planning)
As the title of this subchapter hints, this stage is about planning the actual work of the maintenance organization. It comprises of seven steps defined in CURE. All steps are about planning tools such as organization, contracts, plans and budgets.

CURE suggests that a budget not only concerning the financials to be developed. It should contain information about personnel, tailoring of guidelines and checklists, education plans, meetings and of course financial information such as costs.

1.4.2.4 Transition (Implementation)
The implementation stage is not about conducting the actual maintenance. It is about implementing the maintenance strategy into the organization. Just as the planning stage this stage is defined in seven steps. Among these steps you can find the signing of budgets, contracts and plans, the tailoring in detail of guidelines and checklists and not to forget, the education of the people involved in the organization according to the education plan defined at the planning stage.

At this stage CURE provides detailed guidelines for constructing both a maintenance contract and a maintenance plan.
1.4.2.5 Roles and responsibility
CURE defines eight different roles. These roles are divided into two organizational groups (customer/supplier).

<table>
<thead>
<tr>
<th>Customer:</th>
<th>Supplier:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• System owner</td>
<td>• Contract responsible</td>
</tr>
<tr>
<td>• System maintainer</td>
<td>• Technical system maintainer</td>
</tr>
<tr>
<td>• Personal integrity supervisor</td>
<td>• Operational responsible</td>
</tr>
</tbody>
</table>

Figure 1.5 – Roles in the maintenance organization

These roles should be looked upon as guidelines and not mandatory. The role of the operational responsible for instance is located within both groups. Depending on where the operational response is located within the organizational, the role is used.

The responsibilities associated with each role are defined in CURE at three levels. The strategic level is located at a higher level of the organization such as system owner and contract responsible. The middle level, tactical level, is where the decisions about the actual maintenance work are done. The third level is the operational level. At this level the actual maintenance tasks are performed. CURE provides several responsibility matrices for each level.

1.4.2.6 Organization
CURE provides seven different organization types. Each type is defined using charts and has their own benefits, drawbacks and situations that they are suited for. This is one point where CURE might be modified to suit the existing organization.

1.4.2.7 Meetings
CURE defines three types of meetings. Each meeting is in detail described regarding attendants and what’s on the agenda.

The attendants at the management meetings are the system owner, system maintainer, contract responsible and technical system maintainer. The meeting concerns budgets, plans and contracts.

At preparation meetings the attendants are the system maintainer and the technical system maintainer. At these meeting maintenance actions, release planning and spent costs is discussed among other issues. These meetings are preparations for the maintenance meetings.

The maintenance meeting concerns the system maintainer, technical system maintainer, super user and the operational responsible. The agenda is mostly about the current status of operation, finances, planning and prioritization of maintenance tasks.

CURE has defined guidelines for writing the agendas for each meeting.
1.4.2.8 Maintenance (change request handling)

The process of change request handling consists of seven steps. The process is run every time the customer or the supplier wants to suggest a fix, change or update of the system.

![Diagram of routine for change request handling]

Figure 1.6 - Routine for change request handling

The complete arrows symbolize the normal flow and the dotted arrows symbolize the flow when emergency change requests appear.

Each step is described in CURE at a relatively detailed level. Each step is assigned to one of the roles presented earlier except the step of actually deciding whether the change request is necessary or not and how to implement it. Decisions about change requests have to be made at a maintenance meeting to ensure that it is the right change for the system.

CURE provides a number of templates and guidelines for documents that will ease the continuous maintenance. Some of these templates describe maintenance management schedules, maintenance journals, maintenance reports and operational reports.

The maintenance management schedule gives a good overview of the maintenance planning over time. It shows all planned meetings, maintenance periods, maintenance and operational report deliveries.

The maintenance journal is a document that keeps all incoming change requests. It shows detailed information about every order such as date, priority, status, description and more.

In short, the maintenance and operational reports are descriptions of what has been done and how much resources have been used. They also report a problem – consequence – action analysis.

1.4.2.9 Maintenance (period ending)

At the end of each maintenance period (six or twelve months) CURE suggests that a structured period ending is done. The process suggested by CURE consists of seven steps. Six of the steps are analysis steps for various parts of the continuous work such as organization, operation and change requests. The seventh step is to write a report containing the results of the analysis. CURE also provides a template to ease this step.
1.4.2.10 Maintenance (renegotiations)
Based on the results of the recently ended maintenance period the customer and supplier discuss the future of the system, whether to terminate the system or continue the maintenance.

After these decision the steps suggested by CURE is similar to the ones at the startup. CURE suggests that plans are updated, the budget is updated, contracts renewed, tailorization of processes and routines and the need for education are reviewed.

1.4.3 The English model

1.4.3.1 Purpose
This is a model created to support the maintenance of a specific product. It is documented in detail describing exactly how a maintenance issue should pass through the maintenance organization. This model has also been ISO9001 certified.

1.4.3.2 Lifecycle
The model consists of no general lifecycles. It concentrates more on describing minor lifecycles in detail. The most general is the lifecycle of a support problem. The complete model is very strict with clearly defined routines to follow. Below is the lifecycle of a support problem:

1. The customer sends a support problem to the supplier.
2. The supplier now has x hours (depending on the contract) to respond. In this time they have to analyze and estimate resources for the problem.
3. The customer evaluates the supplier respond and gives ”go”/ ”no go” for the supplier to take actions upon the problem. The customer can also put the problem on hold if it e.g. seems too expensive at the time.
4. If the customer gives a ”go”, the support problem is put into a request log and a deal is made.
5. As the deal is made the supplier starts to design the support problem solution. When done, it gets reviewed and if not approved, reworked.
6. The supplier produces a unit test plan to test the solution. Again it is reviewed/reworked.
7. The supplier implements the solution. Again it is reviewed/reworked.
8. The solution is tested using the unit-testing plan written earlier. Again it is reviewed/reworked.
9. The supplier alters all documents to match the new solution. And a final review/rework is performed.
10. The solution is handed over to the customer for an acceptance testing and the customer gives a ”ok”/”not ok”.

At all times where review is not passed, a problem report is created.

1.4.3.3 Organization
The organization consists of two parts: the support department and the development department. These two departments have no greater cooperation. Their areas are well defined and do not overlap.
1.4.3.4 Roles
In the quality plan the roles within the organization are described with their responsibilities and duties.

- Resource Manager – monitors resources, assists in career development, responsible for training budget, contacts with other maintenance projects and plans and predicts future resource needs.
- Chief architect – decision-maker on technical issues, provides technical leadership, reviews technical documentation, bid reviews for improvements and review and test of code.
- Quality manager – ensures the coherence with ISO9001 among all activities.
- Operational director – overall leadership, progress monitoring, monitoring financial status, approving plans, reports upwards and downwards, strategy decisions and coordination of customer satisfaction issues.
- Project manager – project planning, monitor project progress, monitor project’s financial status, approving project expenses, staffing and resourcing projects and project leadership/coaching.

1.4.3.5 Routines
This model has routines for configuration management, documentation, backup, security, meetings, reporting, risk-management, quality-recording, purchasing, resourcing and financial procedures.

1.5 Standards
Standards for maintenance are not as widespread as other software standards, although it is slowly growing [Takang, p. 163]. The two most known standard organizations are ISO (International Standard Organization) and IEEE (Institute of Electrical and Electronic Engineers). They both have standards that concern maintenance to some extent. IEEE Std 1219 is a well-limited part of the complete IEEE standard collection that explicitly concerns maintenance. Sections of other standards can be mapped to this standard according to the table 1.3 below:

<table>
<thead>
<tr>
<th>ISO 9001</th>
<th>ISO 9000-3</th>
<th>IEEE Std 730</th>
<th>Other IEEE standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.13, 4.19</td>
<td>5.10</td>
<td>3.4.3</td>
<td>1219, 1044</td>
</tr>
</tbody>
</table>

Table 1.3 – Related maintenance standards [Moore, p. 103]

1.5.1 IEEE

1.5.1.1 IEEE Std. 730
This standard’s main concern is to construct a Software Quality Assurance Plan (SQAP). The only part of this standard that is directly connected to maintenance is that the SQAP may contain a maintenance plan. It also describes what this plan should contain i.e. “instructions for software product support and maintenance, such as procedures for correcting defects, installation of enhancements, and testing of all changes”. [IEEE Std. 730.1-1995 section 3.4.3.4]

The section also referrers to IEEE Std. 1219-1992, described below.
1.5.1.2 IEEE Std 1219

This standard consists of 39 pages explicitly dedicated to the management and execution of maintenance activities. The standard is applicable without constraints regarding size, criticality, complexity or application. The standard is divided into three parts. Part 1 contains the scope of the standard, terminology and references to other standards. Part 2 contains definitions and acronyms that are not found in other standards. The third part contains all the mandatory requirements that need to be met, in order to be certified according to the standard. The standard also contains two not mandatory annexes. The annexes are guidelines to conducting better maintenance. [IEEE]

The standard defines seven phases that a maintenance model should contain (problem identification, analysis, design, implementation, system test, acceptance testing and delivery). Each phase is well defined concerning input, process control, output and metrics. The metrics variable is also provided defined factors.

**Problem identification** - At this first phase the problem/modification is identified and given a unique identifier. The process of the phase is also to classify the type of maintenance activity needed, analyzing the problem to determine whether to accept, reject or further evaluate it. If it is accepted is has to be evaluated in terms of resource costs, time as well as developers. Then the problem is prioritized so that more important matters are dealt with first. The last step is to schedule the maintenance request. The result of each step is the output of this phase.

**Analysis** – The input of this second phase is the output of the former phase as well as system and project documentation. This step has two main activities, the feasibility analysis and the detailed analysis. The feasibility analysis concerns the impact of the modification, whether there are any alternative solutions, safety and security aspects, human factors, costs and not least the value of the benefit of making the modification. The detailed analysis concerns how to implement the modification. The analysis examines what parts that will be modified, safety and security issues, drafts a test strategy and an implementation plan for the modification. The output of the phase is reports for both analyses, test strategy, implementation plan, preliminary implementation list and updated requirements.

**Design** – The input of this phase is the output of the analysis phase, system and project documentation and system source code. The issues of this phase is to identify affected modules, create test cases for the new design, create regression tests, update requirements and update modification list. The output of this phase is mostly updated and revised plans and lists such as modification list, test plans, implementation plan and a list of constraints and risks.

**Implementation** – At this phase it is time to actually implement the modification according to all the prior analyses and design. The input of the implementation phase is the same as the output of the design phase as well as source code, system and project documentation. There are four defined activities within this phase. The first one is coding and unit testing. Thereafter integration, risk analysis and test readiness review according to IEEE Std. 1028-1988. The output of the implementation phase is the updated software, updated design-, test- and user documentation, updated training material and a test readiness review report.

**System test** – System testing is also defined by IEEE Std. 610.12-1990. Beside the test readiness report from the implementation phase, the input consists of a lot of documentation and the updated system. The documentation shall consist of system test plans (IEEE 829-1983), system test cases (IEEE 829-1983), system test procedures (IEEE 829-1983), user manuals and the design. The activity of the phase is to test the system. Defined tests are functionality tests, interface testing and regression testing. After these are conducted there shall be a test readiness review. The output of the phase shall be a fully tested and integrated system, a test report and the test readiness report.
Acceptance testing – This phase shall be conducted by anyone else but the supplier. Acceptance testing is also defined in IEEE Std. 610.12-1990. The input to the phase is the tested and fully integrated system as well as the readiness review report, acceptance test plans, cases and procedures. During this phase acceptance testing at a functional level is conducted as well as interoperability testing and regression testing. The three outputs of the phase is a new system base line, functional configuration audit report (IEEE Std. 1028-1988) and an acceptance test report (IEEE Std. 1024-1987).

Delivery – To this phase there are only one input, the fully integrated and tested system represented by the new baseline. Within this phase a physical configuration audit is performed. The user community shall also be notified about the new version of the system. The system shall be installed and if needed the users shall be trained. The system shall also be formed as an archival version for backup use. The two reports that form the output of this final phase is the physical configuration audit report (IEEE Std. 1028-1988) and the version description document.

These seven phases share many similarities to traditional system development. Although this standard is briefly described in this subchapter it gives a hint of the extension of the standard.

[IEEE Std 1219-1992]

1.5.1.3 IEEE Std 1044
This standard is said to give some support to maintenance [Moore, p. 45]. It concerns classification for software anomalies [IEEE Std 1044.1-1995].

1.5.2 ISO

1.5.2.1 ISO 9000-3
The ISO 9000-3 standard, section 5.10, is an implementation of the ISO 9001 standard section 4.19 on software.

The ISO standard 9000-3 section 5.10 concerns maintenance. Compared to the IEEE standard 1219 the ISO standard is very briefly described. The terms used in the ISO standards differs some from the terms used in most maintenance literature and the IEEE standard.

In contrast to IEEE standard, ISO do not describe a procedure in detail to follow. Instead the ISO standard describes parts of a maintenance project such as the maintenance plan and briefly its content, the need for a maintenance contract and organization. The standard also defines types of maintenance activities such as problem resolution, interface modification and functional expansion. About the documentation of the maintenance activities, ISO states that there should be records and reports. Both the supplier and the customer should agree upon the content of the reports. Last, ISO states that plans for releases of new versions of the system should be made and agreed upon by both parties.

The ISO standard is short and brief but covers most parts of the maintenance project/process.

The general requirement of ISO 9000-3 is that the supplier should supply and maintain routines for maintenance. The maintenance activities suggested by ISO 9000-3 are:

- **Problem resolution** – As problems arise they shall be analyzed and corrected. If they cause operational disturbance, emergency solutions might be done and a more permanent solution can be implemented at a later stage.

- **Interface modification** – These modifications might be required as other parts of the system are changed, such as hardware or other components controlled by the system.

- **Functional expansion** or performance improvement (further development) – The customer might want to improve/add functionality or performance to the system.
ISO 9000-3 defines what parts that should be maintained. These parts are program(s), data, specifications, user documents and supplier documents. Which parts that should be maintained and during what period of time should be defined by the maintenance contract. ISO 9000-3 also request that the maintenance work is defined through a maintenance plan. The plan should specify:

- The scope maintenance
- The initial status of the product to be maintained
- A support organization
- Maintenance activities
- Maintenance records and reports

The supporting organization consists of representatives both from the customer and the supplier. The organization shall be flexible enough to make decisions, as the maintenance has to go outside the plan as unexpected issues arise. They should also be able to gather resources that are needed for the maintenance.

According to ISO 9000-3 all maintenance activities should be documented according to the maintenance plan. All documents should also be preserved. The section within the maintenance plan that concerns documentation should set rules for structure, content and distribution. ISO 9000-3 states that the following parts should be present in the maintenance documentation:

- A list of received requests from the customer and their current status.
- The organization responsible for the handling of incoming requests of problem resolution, interface modifications or further development.
- Priorities to the maintenance activities.
- The result of the maintenance activities.
- Statistical data on requests and maintenance activities.

These reports can be used as a base for the evaluation and improvement of the software and the quality.

ISO 9000-3 also provides guidelines for releases of the system. Both the customer and the supplier should agree upon routines for when and how to release a new version of the software. According to ISO 9000-3 these routines should contain:

- Rules for when to solve the problem with a patch and when a complete new release of the system is necessary.
- Description of various types of releases. How they affect the customer’s business and when they can be implemented in time.
- Routines for how the supplier shall inform the customer of planned new releases.
- Routines for affirmation that the implemented release did not contribute to any new problems.
- Documentation that describes what changes that has been made and where.

[ISO 9000-3]

1.6 Summary

- Maintenance is the state in which a system will spend most of its time. The state begins directly after delivery and runs until the system is thrown away. Because of the long time a system is maintained, this phase is the most expensive.
The level of maintenance is based upon the size, complexity, business criticalness, number of users dependent of the system and security aspects.

Too often, the customer does not share the suppliers’ view of maintenance. Customers tend to regard the system as complete after delivery and therefore maintenance is unnecessary expenses.

1.6.1 History and definitions

- In the 70’s there were not many traces of maintenance.
- During the 80’s the concept of maintenance were introduced in Sweden by RDF (RiksDataFörbundet).
- In 1992 maintenance were defined by RSF (Referensgruppen för SystemFörvaltning) : "Maintenance is all activities conducted in order to administrate and manage a system in operation to make sure that the system during its whole lifetime efficiently contributes to fulfill the goals of the business."
- Since then many definitions has been made. Almost all of them share the same base though.
- We suggest the definition stated by CURE and RSF since it is simple and describes maintenance well without leaving any parts out.

1.6.2 SchlumbergerSema

- This master thesis project was conducted in cooperation with SchlumbergerSema

1.6.3 RDF – model

- The RDF model, developed by RDF (RiksDataFörbundet), is divided into eight minor parts. These eight models are divided into three groups.
- The first group is the central models that consist of System model, Life cycle and decision model.
- The second group is called complementary models and consists of maintenance activities and roles.
- The last group is the system concepts that contain staff categories, system hierarchy and function types.

1.6.4 CURE

- The model is developed by SchlumbergerSema (North). Before this model was developed and brought into use, ad-hoc solutions were made almost every time a new maintenance deal was made.
- The lifecycle of the CURE model comprises all stages from product delivery to system termination. Although system termination is not specified within CURE.
- CURE defines both the transition phase from delivery and the maintenance itself. CURE also defines roles, organizations and documentation.

1.6.5 The English model

- The English model is developed by Sema Group in the UK.
- It is compliant with ISO9001
- It is developed for a specific product with detailed role descriptions and routines.
1.6.6 Standards

- We have looked at two major standards, ISO and IEEE. Both concern maintenance.

- IEEE Std 1219 concerns the execution and management of maintenance. The standard is applicable to most projects. It consists of three parts, scope and references to other standards, definitions and acronyms and the mandatory part of the maintenance process. It also provides guidelines to better maintenance. The mandatory process consists of seven phases such as problem identification, analysis, design, implementation, system test, acceptance testing and delivery.

- The IEEE Std 1044 standard is said to give some support to maintenance. It concerns classification for software anomalies.

- ISO 9000-3 section 5.10 is an implementation of the ISO 9001 standard section 4.19 on software. Compared to the IEEE standard 1219 the ISO standard is very briefly described. The ISO standard describes parts of a maintenance project such as the maintenance plan and briefly its content, the need for a maintenance contract and organization. The standard also defines types of maintenance activities such as problem resolution, interface modification and functional expansion.
2 PROBLEM DEFINITION

2.1 SchlumbergerSema related problems
The initiator of this master thesis, SchlumbergerSema, has during the years identified some problems when making maintenance agreements with their customers. The main problem is SchlumbergerSema-internal and is that every maintenance agreement differs from the others because they are individually formed by the account manager and/or his fellows. The differences in the agreements both regard the financial agreement as well as the maintenance arrangements. The consequences of these differences are that it’s hard to move employees between maintenance projects, hard to grasp the overall financial picture over projects and internal loss of potential co-ordination benefits between projects.

2.2 General problems
The problems that SchlumbergerSema experiences can easily be elevated to a general level due to that they exists to some extent in all software development organizations. In order to get an organization to work in a uniform way with software maintenance they need to follow a model. Today numerous models exist such as CURE and RDF described in chapter 1, where many of them share the same origin.

Even if a software development organization follows a model in their maintenance work some problems still exists. There will always be a difference in the level of maturity among the customers and the ability for the customers to adopt the modeled way to work. Another issue is that different software systems have varying levels of maintenance need due to factors as size, criticalness and complexity of the system.

The last two problems shows that even if a software development organization works by a maintenance model that model must be flexible and dynamic in order to fit the needs of the organization and all it’s customers.

One fundamental issue regarding software maintenance models and many other areas as well is the question if you should push all your customers towards working by your own model or if you should work ad hoc, handling each project individually. Are your efforts as a supplier pushing your idea of maintenance towards every customer organization more efficient than just working the easy ad hoc-way?

2.3 The problem definition for this thesis
Among the SchlumbergerSema and the general problems we have chosen to work with the following issues: which model is best suited for SchlumbergerSema and how can this model be tailorized to work smoothly with customers in the public sector?

SchlumbergerSema works today by some different but similar models, which is a problem within the organization regarding terminology, the ability to smoothly move employees between projects and weak financial overview of the maintenance agreements.

SchlumbergerSema works mainly with large customer organization and many in the public sector where they have experienced a need for a specialized maintenance model packaged for those customers. Could this public sector-package be a tailorization of the chosen model?

Prior to starting this research project we have identified seven questions in our research proposal that this thesis will provide answers to:

- Have the system maintenance knowledge from 1980’s been forgotten?
Is there a difference in the need or characteristics of system maintenance depending on the code type of the system (old Cobol systems vs. modern object oriented systems)?

Which system maintenance methodology is best suited for different code types (see above)?

Is there a difference in the need or characteristics of system maintenance depending of the customer type or organization (public vs. private or customer market e.g. telecom, business administration etc.)?

Which system maintenance methodology is best suited for different organizations (see above)?

How can maintenance models be tailorized?

How can maintenance models be adapted (in full, as guidelines)?

The objectives for this research, also listed in our research proposal, are:

- Identify existing maintenance methodologies available in the industry and document their characteristics by comparison.
- Identify the possibility to tailorize, the maintenance methodologies found, for the public sector.
- Identify whether the developers and the customers share the same view of software maintenance.
- Identify the viewpoint of software maintenance by standards organizations such as ISO.
- Present a software maintenance package suitable to the market.
- Present a suitable usage, tailoring and modification of CURE (SchlumbergerSemas internal initiative of software maintenance) mainly for customers in the public sector.

2.4 Summary of Problem Definition

Many of the problems experienced by SchlumbergerSema within the domain of software maintenance are general problems experienced by most software development organizations. Among those problems we have chosen some that we will investigate through a case study and literature studies.
3 DESCRIPTION OF RESEARCH METHODOLOGY

3.1 Introduction
This subchapter will describe the research methods that we will use during this master thesis. When conducting a research project it is necessary to thoroughly plan the strategies that will be used in order to be able to collect data of good quality and data that will give you a correct basis for the analysis.

After the data is collected an analysis is performed that will eventually lead to some conclusions and recommendations. To be able to carry out an analysis, domain knowledge is needed and therefore, as in most research projects, the first method used is literature studies.

All the separate methods combined in the master thesis are called the methodology. Many of the methods in the methodology are ordinary and used in most research projects, but the case study that is one of the main methods that we are going to use is the one that is most tailorized for our needs.

3.2 Pre-study
Pre-studies in this research project consist of getting to know people that work in the domain, interview them, read literature and participate in courses on the subject. The following subchapters will describe the pre-study in this research project furthermore in detail.

3.2.1 Literature studies
As in almost every research project we must study a lot of literature in order to obtain domain knowledge, vocabulary and knowledge about prior research. The domain knowledge will be needed throughout the entire project, during interviews, analysis of responses, making conclusions and writing future recommendations.

The knowledge about prior research is essential due to the fact that we wouldn’t like to perform a survey very similar to what someone else might have done in the past. Studying old research on the topic is interesting in many ways because it often contains information about what they felt was good and bad methods or approaches. Most research material contains information and recommendations about how the authors would perform the research if they could do it over again i.e. learn from others mistakes.

3.2.2 Internal education
For us as researchers to obtain the current system maintenance knowledge of SchlumbergerSema we will need to participate in their internal courses on the subject. SchlumbergerSema has developed a maintenance model CURE, that they have begun to use in the northern parts of Sweden. We will participate in an introductory course in CURE to get a basic knowledge about it and its procedures.

CURE or a tailorization thereof might be the future for SchlumbergerSema in other parts of Sweden as well.

In the early contacts with SchlumbergerSema we will try to get in contact with employees that have worked with system maintenance for a long time. These contacts will give us a good start in the search for accurate literature and connections to other interesting people and resources in the domain.
3.3 Case Studies

The case study is the most important phase of this research project and it is important that it is performed in a good way in order to collect and analyze information correctly. This phase is also the one with most contacts with other parties like numerous development teams, sales managers, key account managers and customers. The case study comprises investigation of maintenance agreements, system specifications and last but not least interviews of involved personnel.

In order to get the best results out of the case study it is very important to think through the purpose and the goals of the case study.

These issues are described in the following subchapters.

3.3.1 Background

We have chosen to collect up-to-date information about software maintenance from the industry by performing case studies of at least 5 software development projects delivered by SchlumbergerSema. The reason for choosing case studies as our data collection methodology is that we will be able to spend more time with each project. Spending more time with each project enables us to review project documentation more thorough and interview more personnel and each customer [Patel, p. 62]. This approach allows us to investigate each project from more angles and meet the interviewees in their natural environment and grasp the atmosphere within each maintenance organization. This technique is called descriptive studies by Patel [Patel, p. 62].

We are going to interview many different roles in each project to get a better and more accurate understanding of how the maintenance is agreed upon, set up and run. The roles within each project at SchlumbergerSema that we are going to interview are the salesman, the project leader, system maintenance manager and a developer. For each project we are also planning to interview the customer.

3.3.2 Purpose

The overall purpose of the case study is to collect information from the customers about how they experience maintenance in general and from SchlumbergerSema in particular. Other secondary purpose is to gain knowledge of how the system maintenance is marketed, sold and delivered by SchlumbergerSema.

We have identified some areas that the case study will be aimed at trying to reveal some specific information. These areas are:

- Re-use of code or components. Identify the level of re-use of code and components between projects. Does system development consider sharing or re-using components or code in order to facilitate for system management.
- Maintenance tools. Are there any tools used in the system maintenance work. Tools for change requests, maintenance logbook or customer satisfaction.
- Maintenance metrics. Is there a way to measure the performance or customer satisfaction? Is there a way to measure the level of maintenance need for each system or customer?
- Allocation of resources for maintenance. How much resources are used for maintenance. Is this measurable and used as a basis for new agreements?
- Size of project/need for maintenance. How large and critical is the system, in development hours, users, load, and uptime. How does this affect maintenance needs?
- How is system knowledge maintained? How do personnel at SchlumbergerSema maintain an appropriate level of knowledge of delivered systems? How is this cost financed?
• Origin of maintenance model. Does the maintenance agreements and maintenance organizational structure look alike between projects? Does the maintenance model have any origin?

3.3.3 Preparations

Before the actual case studies start we are going to develop a questionnaire, see below, for the interviews and make a light pre-study of each customer participating in the case study. We will also collect and review documents regarding the projects prior to the interviews.

3.3.4 Collect data

As a part of the case study we will collect and review numerous documents regarding the projects to gain more specific knowledge about the arrangement between SchlumbergerSema and the customer. The documents that we will try to collect if existing are:

• Technical environment. This document will show if the system is operated within SchlumbergerSema or at the customer’s site.
• System specification. This specification will give us basic knowledge about the system architecture, usage, connectivity to other system etc.
• Maintenance agreement. This document will show the financial agreement between SchlumbergerSema and the customer. Most interesting here is to see if this agreement is separate from the system development agreement or integrated. This document will also reveal the maintenance organization and obligations for each party. Specific details that we will investigate are when in the sales/development phase that this agreement was made and for how long it will be valid.
• Maintenance logbook. If a logbook is kept this will show the frequency of change requests, systems stops or other errors.

3.4 Interviews

After having studied the collected documents regarding a software development project in the case study it is time to interview some key personnel. The interview will consist of approximately 30-40 questions with open-ended questions and a list of maintenance metrics to be discussed and evaluated. The idea is that the entire interview will take maximum 60 minutes per person.

We have chosen to try to meet every interviewee for a face-to-face interview due to that you easily can give explanations to unclear questions and ask for clarifications on unclear answers. In a face-to-face interview you will receive answers of much higher quality than interviews or questionnaires performed by phone or mail [Patel, p. 102].

3.4.1 The classic interview

The main purpose of the classic interview is to collect information, experiences and ideas from personnel working with software maintenance in the industry. Some other aspects that we will investigate during the interviews are how the customer looks upon SchlumbergerSema and its software maintenance services:

• Are the services marketed well?
• Do the customers understand from the beginning that they will need to purchase these services?
• Does SchlumbergerSema sell these services along with system development?
• Do the customers believe that these services are expensive?
• How is SchlumbergerSemas software maintenance services compared to other suppliers?
We will also collect administrative data about maintenance organization and its workload. We believe that it is necessary to have this data when we are comparing the different projects. Some examples are: the frequency of maintenance meetings, the frequency of change requests, the size of the maintenance organization, the expected lifetime of the system, the uptime-requirements on the system, the number of users etc.

As mentioned above we are going to interview numerous roles for each project in the case study, the salesman, the project leader, the technical maintenance manager, one developer and the customer. We have decided to use the same questions throughout the entire case study. This means that all projects and all roles that we will interview will answer the same questions. There are a couple of reasons for this: easy to identify differences between projects and easy to identify differences in the apprehension of maintenance between different roles.

3.4.2 Maintenance apprehension
The apprehension of software maintenance between different roles in the organization and projects is by us believed to differ a lot. Software maintenance is in the belief of the majority of personnel working in the software engineering field something that is both time consuming and boring. We are anxious to see how this belief corresponds to reality and if it differs among project roles.

One part of the interview will cover this issue and hopefully give us some understanding of the origin of the bad vibes associated with maintenance.

Apprehension is generally something that has very loose roots and often established without any substantial knowledge. Knowledge about system maintenance is another aspect that we will cover in the interviews to determine how the knowledge is spread among the different roles in the organization and among the customers.

3.4.3 Re-usability of the metrics measurement questionnaire
One specific benefit achieved by discussing metrics regarding measurements of software maintenance efficiency is that when a relevant set of metrics are chosen, a questionnaire can be developed collecting data around those metrics. The questionnaire should have well defined questions and measurable answers because that would enable re-use of the questionnaire in the future in some appropriate interval e.g. every 6 months.

Performing an evaluation on a system in maintenance today and bringing its strengths and weaknesses to light is appealing. Naturally you as a supplier will try to resolve and reduce the weaknesses found. After 6 months you can perform another evaluation with the same questionnaire and you will now be able to measure the differences and identify trends since the last evaluation.

With regularly measurements you as a development organization you will also have the possibility to compare maintenance projects with each other to find differences in efficiency, personnel needs and costs.

As mentioned above, one of our purposes with the interviews is to try to identify some metrics in software maintenance that will enable more accurate measures of its effectiveness.

3.5 Analyze data
When the interviews are finished we will analyze the responses both from the interviews and from the discussions on maintenance metrics. The things that we are looking for are especially all the points mentioned above under purpose.

Some aspects are more important than others e.g. if any project/department have packaged maintenance services and market this as a separate service.
Another characteristic that we will focus on is the origin of the maintenance model that different project/departments use. Is the model built from another model or is it remade every time?

3.5.1 Matrix analysis
We are planning to make a matrix analysis of the results from the interviews with the projects on one axis and roles on the other axis. Conducting the analysis in this matter enables us to see differences in attitude towards maintenance among for example the salesmen. It will be interesting to see how maintenance services are packaged, marketed, sold and agreed upon by a number of salesmen in the same organization.

On the other axis the benefits will be that the analysis will reveal how maintenance is apprehended by all roles in one project. For example: what worked well and what did not and what are the lessons learned within each project.

3.6 Categorize data
Most of the responses that we will receive from the interviews with text-answers will not need further categorization more than putting them in the correct stack e.g. ideas, problem areas, good practices etc. Some of the questions are more specific and will be kept together with the answers from all interviews e.g. “What factors determines the need for software maintenance?”

We will use many of the questions and their corresponding answers in the process of recommending a new maintenance model that will incorporate many of the good practices and ideas discovered during the case study, e.g. “Do you have any ideas on how to make software maintenance more popular?”
3.7 Evaluating data

We have chosen to evaluate the data from the case study and the literature studies as in figure 3.1. A description of the numbered activities is found in table 3.1 below the illustration.

Figure 3.1 – Overview of the methodology

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The general knowledge of software maintenance from literature studies. Chapter 1.</td>
</tr>
<tr>
<td>2</td>
<td>The educational material of the maintenance model CURE.</td>
</tr>
<tr>
<td>3</td>
<td>Requirements on a good maintenance model from literature. Chapter 4.</td>
</tr>
<tr>
<td>4</td>
<td>All the material from the interviews.</td>
</tr>
<tr>
<td>5</td>
<td>A list of problem areas with software maintenance models identified by the case study. Chapter 4.</td>
</tr>
<tr>
<td>6</td>
<td>A comparison between the case study’s requirements of a good maintenance model and CURE. Chapter 4.</td>
</tr>
<tr>
<td>7</td>
<td>A comparison between the literature’s requirements of a good maintenance model and CURE. Chapter 4.</td>
</tr>
<tr>
<td>8</td>
<td>A comparison between the case study’s requirements and the literature’s requirements of a good maintenance model. Chapter 4.</td>
</tr>
<tr>
<td>9</td>
<td>Our recommendations within software maintenance. Chapter 5.</td>
</tr>
<tr>
<td>10</td>
<td>The conclusions of this thesis. Chapter 6.</td>
</tr>
</tbody>
</table>

Table 3.1 – Explanation to figure 3.1
3.8 Summary of Methodology

- In this methodology chapter we have described how we will perform our research starting with literature studies, preparation of case study, case study, analysis of data, recommendations and finally with writing the conclusion of the thesis.

- We have in detail described how we will prepare and perform our case study with at least five software systems delivered by SchlumbergerSema.

- Within these projects we have described that we would like to interview the five roles: the salesman, the project leader, the technical maintenance manager, one developer and the customer.

- We have described how we afterwards will analyze the responses and other collected data.

- The last subchapter illustrates how the evaluation of all data will be done leading to recommendations and conclusions. The evaluation consists of lists of requirements that will be compared with each other and the maintenance model CURE.
4 Research

4.1 Overview of the Case Study

During the mid-phase of this master thesis we interviewed fourteen persons involved in five different projects. The interviews were performed with personnel working in different roles in the projects such as customer representatives, account managers and maintenance managers. The five projects were chosen in mutual agreement with SchlumbergerSema. The reasons for choosing these projects are because they represented different business segments within SchlumbergerSema’s operations in southern Sweden and that they have either been interested in or used SchlumbergerSema’s maintenance services.

The individuals within each project were chosen in such a way that we would get one person from each role (customer, account manager and maintenance manager) within each project. Unfortunately this could not be fully fulfilled and the distribution of roles interviewed is described below. The reasons for us not fulfilling our initial objective regarding the distribution of roles are that every project didn’t work in the same organizational structure, we were not allowed to interview one customer and some were unwilling to participate.

Roles in the case study:

- Four persons were account managers
- Six were maintenance managers
- Three were customer representatives
- One was a consultant

Gender distribution in the case study:

- Six were women
- Eight were men

Of the companies that we met:

- One was in the public sector
- Four were private corporations
- One of the companies were in the IT-sector
- Four of the companies had their main business in totally differing areas

The projects that we studied had very differing characteristics, some were old systems that had been in maintenance for years and some were brand new with not yet signed maintenance agreements. The software systems in the case study were also of differing kind:

- A time reporting system
- An intranet solution
- A quotation/sales-management system
- A billing system
- A compensation-management system

The questionnaire that we used is found in Appendix A. During the interviews we both tried to get a picture of the interviewee’s general knowledge and apprehension of software
maintenance but we also tried to collect ideas about improving its efficiency and the public’s apprehension.

Another part of the interview was to collect information about the actual project that either was in maintenance or just about to enter the maintenance phase. We asked questions about which maintenance model they were using and the benefits and drawbacks thereof. We asked questions about how they thought about their maintenance organization, decision-making, responsibility assignments and change/error handling/tracking.

The results and findings from the interviews are described in the following subchapters.

4.2 Results/Findings regarding general software maintenance issues

In order to find results and identify findings in the responses from the interviews we had to identify categories and to group the answers together within them. Some of the categories in this subchapter have been identified by as few as one interviewee.

4.2.1 Identifying categories

Since the interviews are based on open-ended questions the answers we received were text based and sometimes long descriptions and sometimes as short as a yes or no. We were interested in lowering the number of different answers regarding one question. After we read all the answers we identified some categories of answers and then counted the ones that fell into each category.

Some questions had answers that could be rated from good to worse and around some questions we could identify different areas such as e.g. economical, security and critical aspects where we placed the answers in the corresponding bucket and counted them afterwards.

4.2.2 Results regarding the Need for Software Maintenance

One of the fundamental questions regarding software maintenance is to explain or express the need of it. When is software maintenance needed and when is it not? One question closely related to this one is the one concerning what factors to assess when determining the need for software maintenance.

Almost every response indicates that you need some level of software maintenance with every system. Some special cases are mentioned when it might not be necessary, such as low frequently used systems or stand-alone systems with no network capabilities.

The next question in the questionnaire the interviewee was asked to present relevant factors to evaluate when determining the level of software maintenance.

The results from the case study on this matter were categorized in four areas. Within each area the findings regarding important evaluation factors are listed. Of the 14 persons interviewed we received 27 responses in this area. Table 4.1 below is sorted in such matter that the most popular responses are at the top and the least popular are at the bottom.
<table>
<thead>
<tr>
<th>Category</th>
<th>Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size aspects</strong>, mentioned 12 times</td>
<td>System size</td>
</tr>
<tr>
<td></td>
<td>Development time</td>
</tr>
<tr>
<td></td>
<td>Number of users</td>
</tr>
<tr>
<td></td>
<td>Complexity</td>
</tr>
<tr>
<td><strong>Critical aspects</strong>, mentioned 7 times</td>
<td>Business criticalness</td>
</tr>
<tr>
<td></td>
<td>Integration with other systems</td>
</tr>
<tr>
<td><strong>Economical aspects</strong>, mentioned 6 times</td>
<td>Size of investment</td>
</tr>
<tr>
<td></td>
<td>Lifetime of the system</td>
</tr>
<tr>
<td></td>
<td>Customer’s financial situation</td>
</tr>
<tr>
<td></td>
<td>Rate of planned future changes/upgrades</td>
</tr>
<tr>
<td></td>
<td>Uniquely developed system or regular product</td>
</tr>
<tr>
<td><strong>Security aspects</strong>, mentioned 2 times</td>
<td>Reliability of system</td>
</tr>
<tr>
<td></td>
<td>Integrity of data</td>
</tr>
</tbody>
</table>

*Table 4.1 – Factors affecting the need of maintenance*

Two persons interviewed described a great metaphor when comparing maintenance of software systems to the maintenance of cars. A car owner that has the appropriate knowledge, resources, time and tools can service his car by himself. If he or she lacks some of these requirements the car can be sent away for service at a car repair shop. The car engineer conducts the maintenance of the car according to the detailed description in the car’s service-plan.

If you compare this to the maintenance of software systems the service-plan is the maintenance agreement and the car repair shop is the supplier with technicians and
consultants that has knowledge about your system. As an owner of a system you can choose to perform the maintenance of the system within your own organization if you have knowledge, resources, time and personnel. Otherwise you have to buy the maintenance from a supplier.

The ownership of a modern car is one of the best working maintenance solutions in society. A modern car indicates to their owner when the car needs service in intervals calculated both in time and distance traveled since the last service.

The two persons pointing out this metaphor emphasized that it is very important that the owner of the software system understands the need for maintenance just as naturally as they understand that their cars needs service.

4.2.3 Results regarding Definitions of Software Maintenance

One of the hardest questions in the questionnaire was for the interviewee to define the term software maintenance. Most often mentioned in the responses where that software maintenance was the effort to make sure that the software system supported the main business needs of the corporation.

A couple of other definitions included the efforts to:

- Cost effectively extend the lifetime of the system
- Organize and plan resources for long-term maintenance
- Ensure security of the system and integrity of the data
- Maintain and manage system knowledge
- Undertake overall commitment to ensure system availability

Some of the persons referred to the definitions by CURE and RDF.

CURE’s definition of software maintenance is:

“Maintenance is all the activities performed in order to administrate and manage a system in operation, in a matter that during its lifetime it efficiently contributes to achieve the goals of the main business.”

RDF’s definition of software maintenance is:

“Maintenance is a term for the activities needed in order to administrate and manage a system in operation that is not a part of the system operation or of the type further development.”

4.2.4 Results regarding Changes in Software Maintenance since the 1980’s

Half of the interviewed persons believed that knowledge about software maintenance has decreased or been forgotten since the 1980’s and the other half believed that it was the same or refined. Not that many of the interviewees were actually working with software maintenance or working at all during the 1980’s. The answers from the ones that do not have any real experience from the 1980’s were based on the general apprehension of the history.

From an interviewee that was actually working with software maintenance during the 1980’s we got the following interesting statement: “Since the 1980’s the need for software maintenance has increased. The reasons for this are higher level of complexity in the systems, higher requirements on the systems (24 hours/day, 7 days/week) and more technologies and products involved in each solution.”
Packaging and marketing of software maintenance services is one aspect that was bad in the 1980’s and still is. There are no standard agreements and every deal is tailored regarding level of service and pricing."

From another interviewee working with maintenance in the 1980’s we got the following statement: “Software development has developed tremendously since the 1980’s but the software maintenance has not developed much over the years. Due to this fact software development is moving very fast today with developers going from project to project. Software maintenance has not been able to keep up and therefore looked upon as sometimes old and stiff”

4.2.5 Results regarding Apprehension about Software Maintenance

In the subchapter regarding maintenance apprehension among the interviewed persons we have categorized the answers within the areas personal apprehension, personal knowledge development and the organizations apprehension.

Within the area of personal apprehension about software maintenance we found that:

- 55% thought it was interesting
- 33% thought it was fun
- 12% thought it was boring

Activities to improve the general apprehension of software maintenance were collected in the interviews. The most mentioned activity was to change the terminology regarding maintenance and instead calling it on-going development, refinement or further development. This is also supported by the literature [Bergvall p. 107] where a comparison is made to other business areas where the terminology was changed and the status of that area was improved.

Another activity that was mentioned was to increase the status of maintenance work within the organization e.g. same salary/benefits as the developers, emphasize the responsibility of maintenance roles and rewards based on customer satisfaction etc.

Two other activities in order to improve the general apprehension of software maintenance was mention by some interviewees: commitment and support from corporate management and continuous development/refinement of the maintenance model used.

In the area of personal knowledge development the interviewed persons responded that they experience personal development working with software maintenance. Most of the responses indicated that working in a maintenance role is very knowledge demanding. It is both demanding regarding all technically related areas such as the systems, products, operating systems etc. but also regarding the customer and the business domain of the customer.

Activities to improve the apprehension about software maintenance in this area are: continuous development of the maintenance model, continuous strive towards improved goals, to have varying work tasks and to participate in courses in order to be up to date in the industry

In the area of organizational apprehension of software maintenance all of the responses indicated that software maintenance is necessary and that this must be emphasized towards everyone in the organization as well as customers.

One activity mentioned in the responses in order to improve the organizations apprehension of maintenance is to emphasize the profits and benefits of properly working software maintenance both for the supplier as well as the customer.
4.2.6 Results regarding CURE-knowledge

One fourth of the interviewees had never heard of or had no knowledge about CURE prior to this case study. The rest had knowledge ranging from 2 hour self studies up to participating in the 2-day course and real life experience.

Most of the comments regarding CURE were positive and just a few were slightly negative. The positive comments were:

- Good description of roles
- Good templates
- Efficient
- Good as a reference
- Easy and clear to present
- Good responsibility matrix
- Still room for tailorization and individual solutions

The negative comments were:

- Must still be tailorized to every organization
- No computer-based tools, yet

4.3 Results/Findings regarding the maintenance projects issues

This part of the results presentation will focus on the characteristics identified when we studied the five systems and the personnel working with them.

4.3.1 Description of systems participating in the case study

We have together with SchlumbergerSema decided that systems and their customers participating in the case study should be anonymous to the public. In this report the participating systems will be referred to as maintenance projects A, B, C, D and E. SchlumbergerSema handles the software maintenance for all these systems.

4.3.1.1 Maintenance project A

The system in this project is an intranet solution developed for a large Swedish corporation with business and users all over the world. The software maintenance model used is parts of CURE. The maintenance agreement is signed.

The persons working in the maintenance of this system enjoy that the information channels are defined, good templates and good organizational structure on both sides. One drawback mentioned in the maintenance of this system is the lacking of an appropriate error reporting and change request system.

4.3.1.2 Maintenance project B

The system in this project is a system that manages applications from applicants, monitors the applications during the decision-making and handles payment of the final compensations. This system is developed for a Swedish public authority with users all over Sweden. The software maintenance model used is based on the customer’s earlier organization and influenced by CURE. The maintenance agreement is being developed.
In this maintenance organization both the customer and SchlumbergerSema are taking an active part in the development and tailorization of the maintenance model.

Positive comments mentioned about the maintenance of this system are good relation and dialog between the supplier and the customer, full traceability of requests and good tools. One drawback mentioned in this maintenance organization is that it might be too bureaucratic when dealing with small tasks.

4.3.1.3 Maintenance project C
The system in this project is a system for handling quotation and sales for a medium sized Swedish corporation with business and users in northern Europe. The software maintenance model is not outspoken, only some low-level routines are established. The maintenance agreement is not signed.

Regarding this system the customer has not understood why they need maintenance and therefore no agreement has been signed. Maintenance activities such as support, error reporting and change requests are handled the easy way by phone and efforts by consultants selling hours. This system is quite stable and not that many requests are handled every year.

4.3.1.4 Maintenance project D
The system in this project is a billing system for telecom services used by a large telecom operator. The system is used for billing in three northern European countries. The software maintenance model is a model developed in England, which is ISO-certified. The maintenance agreement is signed.

The maintenance of this system is a 24/7-solution (24 hours/day and 7days/week) with support and agreed response times for different priority classes.

The maintenance model used is very detailed with routines and meetings. The model is well worked-in among the involved personnel especially on the customer’s side.

One drawback mentioned is that the model might be too bureaucratic when handling easy cases and task.

4.3.1.5 Maintenance project E
The system in this project is a time-reporting system for the personnel developed for a large Swedish corporation operating in northern Europe. The software maintenance model used is very close to CURE. The maintenance agreement is signed.

The maintenance of this project uses a lot from CURE but the role descriptions are refined and developed further to fit in with the rest of the customer’s organization. The persons working with the maintenance enjoy the templates and the guidelines that CURE comprises.

SchlumbergerSema has presented CURE to this customer and they have worked together with the adaptation of CURE to the customer organization. This introduction has given both organizations the same and shared view of software maintenance, which enables good and efficient cooperation.
4.3.2 A summary of the maintenance projects

<table>
<thead>
<tr>
<th>Maintenance model used?</th>
<th>Project A</th>
<th>Project B</th>
<th>Project C</th>
<th>Project D</th>
<th>Project E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CURE light</td>
<td>CURE and some more</td>
<td>None</td>
<td>English</td>
<td>CURE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maintenance agreement formed?</th>
<th>Yes</th>
<th>Under construction</th>
<th>Yes</th>
<th>Yes</th>
<th>Yes</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Maintenance agreement signed?</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
<th>Yes</th>
<th>Yes</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Good statements:</th>
<th>Templates, structure, information-channels</th>
<th>Relation, dialog, traceability of requests, good tools</th>
<th>Good routines, well worked-in</th>
<th>Templates, guidelines, shared view of maintenance</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bad statements:</th>
<th>Bad change-request routine</th>
<th>Too bureaucratic for easy or urgent tasks</th>
<th>Different view on maintenance</th>
<th>Too bureaucratic for easy or urgent tasks</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>When was maintenance sold?</th>
<th>In the end of the development</th>
<th>Initiated by the customer at beginning</th>
<th>Attempts after development</th>
<th>Together with sale of product</th>
<th>In the middle of the development phase</th>
</tr>
</thead>
</table>

Table 4.2 – An overview of the maintenance projects

4.3.3 Conclusion of maintenance projects issues

As a conclusion of this subchapter we have found that SchlumbergerSema does not need any tailorization of the maintenance model. SchlumbergerSema can work by CURE and does not need to tailorize anything within the model. The customers and their habits from prior maintenance projects is the major source of the need to tailorize the maintenance model.

4.4 Comparison between CURE and the case study’s requirements on a software maintenance model

In this subchapter we have made a comparison between CURE and the requirements from the case study regarding maintenance models. In the subchapters below we have described how we identified those requirements and why we have chosen to compare them with CURE. The areas covered in this subchapter were presented by one or more interviewees. The result of the comparison will be defined as level of match. There are three levels: high, medium and low. The match will be high if it is a perfect match, medium if the requirement is filled to some extent and low if no match is found within CURE. The results are summarized in a table in subchapter 4.4.3.

This subchapter has been reviewed and approved by the CURE-development team.

4.4.1 Case study’s requirements

We have in the material from the case studies identified ten areas of requirements that the interviewees has on a software maintenance model. The requirements have either popped up as an idea of improvement to an on-going maintenance project from involved personnel or as something that not works well today. One question in the questionnaire was how the interviewee would like maintenance to be working. Some of the requirement areas were found in the answers to this question.

4.4.2 The Comparison with CURE

As an assignment from SchlumbergerSema we have compared the requirements on a software maintenance model from the case study with CURE. SchlumbergerSema were interested in getting their new maintenance model, CURE, evaluated with the latest requirements of the actual industry.
4.4.2.1 Well-defined and clear role descriptions
Many of the interviewees mentioned that part of a maintenance model that they found most useful was the part describing the roles within the maintenance organization. CURE has seven proposed maintenance organization structure with clearly but very generally described roles. The description of roles is the part that is most tailorized within the industry today.

The benefit of having only general description of roles is that the model is more suitable for many types of organizations and corporations.

Within CURE the roles are connected to responsibilities within a large matrix. Within the matrix the responsibilities are thoroughly described. Due to the connection between roles and responsibilities it might be implicated that the roles are thoroughly described as well, even though it might not be obvious to a new CURE user.

The organizational structures proposed in CURE are very well connected to the roles.

**Level of match:** High

4.4.2.2 Efficient error/request handling systems
CURE describes a routine for the handling of requests from the organization using the system. The routine is very easy to grasp and it is obvious that it would be an efficient way to work if it was implemented correctly. Within CURE there are also templates that would ease the effort in creating a maintenance agreement with the customer. What seems to be lacking in CURE in this area is a guide about how to implement the routines between the customer and the supplier. One positive aspect with CURE’s request handling routine is that it contains shortcuts through the routine for a smooth handling of easy and/or urgent tasks.

Many organizations find it necessary to tailorize this part of the model to fit in with the rest of their organization. Some reasons for this are that the organization has e.g. old habits with personnel used to review some specific issues within IT-systems. Some examples of this are an introduction of new roles such as a security manager or a personal integrity supervisor.

As mentioned in the header of this subchapter the interviewees are asking for a system. CURE only contains description of routines, templates for agreement and other information on paper. The interviewees are asking for a computer-based error reporting and change request system.

**Level of match:** Medium

4.4.2.3 Templates
CURE contains templates for all the necessary agreements and the reoccurring reports. Prior to use some templates a discussion with the customer has to take place in order to choose organizational structure for the maintenance. Another crucial task is to make the financial agreement with the customer.

After the maintenance agreement is formed the templates can be used without tailorization. The templates are a huge strength within CURE because it gives the persons trying to start to work according to CURE a great support and structure.

**Level of match:** High

4.4.2.4 Traceability of submitted errors and requests
Traceability of all error reports and change requests are crucial for the maintenance to work over time. Traceability is mentioned in CURE in the form of a unique identifier on every change request and error report. This unique identifier is also present in the examples and templates. As mentioned in the subchapter 4.4.2.2 everything is on paper and the interviewees are longing for a computer-based system.
One template presented in CURE is the maintenance journal which is a great example showing traceability of all received requests and error reports.

**Level of match**: Medium

### 4.4.2.5 Guidelines
As we interpret the term guideline within software maintenance we have identified two separate parts. The first type of guidelines regards the startup-process, when you try to work according to CURE. The other type of guidelines deals with the tasks for an on-going maintenance organization.

CURE does not contain guidelines that will help you to start to work according to it. Besides the model CURE there are two ways to begin working according to it, the first one is to hire CURE-consultants and the other is to participate in CURE educational courses.

The maintenance model CURE contains a lot of guidelines for the regular maintenance work, when you work in a CURE-way, it’s easy to continue doing so. The guidelines assist the user of the model at many different levels, what kind of meetings to have, when to have the meetings, what to discuss and decide on each meeting and who should attend the meetings. The guidelines cover many other routines and process all over the area of maintenance work.

As the guidelines within CURE are designed today, an organization will need help to get started working according to CURE but when they are up and running the guidelines will keep them on track.

**Level of match**: Medium

### 4.4.2.6 Defined information channels
The information channels within the maintenance organization are well defined horizontally but not mentioned vertically. Horizontal communication occurs between the customer and the supplier at each level in different maintenance activities mentioned in CURE such as maintenance meetings etc. Vertical communication occurs between the different levels within both the customer and the supplier organizations and this is not mentioned in CURE and is assumed to be handled within the line organization on each side.

**Level of match**: High

### 4.4.2.7 Appropriate level of bureaucracy
In this case we define bureaucracy as the resistance and struggle when making decision.

CURE is equipped with a responsibility matrix were the responsibilities of each role within the organization are described. If this information is public knowledge within the maintenance organization everyone understands the process of making decisions of different significance.

CURE’s decision-making model contains a lot of steps to be strictly followed but it’s also equipped with a shortcut for easy and/or urgent tasks.

**Level of match**: High

### 4.4.2.8 Appropriate level of administration
In this case we define administration as the creation of reports and other documents.

CURE indicates that a maintenance report shall be created after each maintenance period e.g. every six months. This is by many, believed to be very unnecessary and boring. The benefits of creating this report are for new employees to learn from, for management to identify
trends since the last report and as a basis for writing future maintenance agreements. CURE has templates for these kinds of reports and CURE also defines which role that should write which report.

**Level of match:** High

### 4.4.2.9 Maintenance services integrated in the sales process of system development

Every newly developed system needs some level of software maintenance and it is therefore crucial for the supplier’s sales personnel that normally sell system development projects to learn to also sell maintenance.

CURE does not define when/where in the system development process to sell maintenance services. Within CURE it’s very well defined what to do when maintenance is sold to a customer but not how to sell it.

Even though a closer integration with system development is required in the arguments above it is good that CURE is kept separate from the system develop process. The reason for this is that CURE is a pure maintenance model used by the maintenance organization and should function with any system development model that is used by the system development organization.

**Level of match:** Medium

### 4.4.2.10 Well-packaged maintenance service in order to ease marketing and sales

We have identified three different types of packages that can be sold, CURE education, CURE-consultant services and maintenance according to CURE.

Today only the first one is packaged and easy to market and sell. The CURE education consists of three different kinds of courses about the model, which are all specified and priced.

The second type of package is a startup package with CURE-consultant services that will help a customer to establish a maintenance organization together with a supplier. This package is today sold to some customers buying large systems by SchlumbergerSema but is not yet sold as a separate product to corporations having systems developed by others than SchlumbergerSema.

The third type of package that the interviewees want is the maintenance services package. Today every maintenance agreement is individually arranged by the account manager, which makes it hard to market those services. The agreements sold are to customers that are also buying a new system. If well-defined maintenance packages, with different service levels, would be defined and priced it will be much easier to sell maintenance service. As described in the English maintenance model in chapter 1, different service levels are defined in a couple of separate packages. The packages in the English model differ in e.g. maximum response time (on a request), support service hours (only daytime to 24/7) and of course price level.

**Level of match:** Low
4.4.3 Results of comparison between CURE and case study’s requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Level of match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear role descriptions</td>
<td>High</td>
</tr>
<tr>
<td>Routines</td>
<td>Medium</td>
</tr>
<tr>
<td>Templates</td>
<td>High</td>
</tr>
<tr>
<td>Request traceability</td>
<td>Medium</td>
</tr>
<tr>
<td>Guidelines</td>
<td>Medium</td>
</tr>
<tr>
<td>Information channels</td>
<td>High</td>
</tr>
<tr>
<td>Bureaucracy</td>
<td>High</td>
</tr>
<tr>
<td>Administration</td>
<td>High</td>
</tr>
<tr>
<td>Sales process</td>
<td>Medium</td>
</tr>
<tr>
<td>Package/business friendliness</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 4.3 – Overview of the comparison

To summarize the conformity between CURE and the case study’s requirements it can be determined that CURE supports all of the requirements within the maintenance organization and the daily work conducted there. The kinds of requirements that CURE does not fulfill among the ones presented by the case study are the ones that could be categorized as external, such as a well-packaged for the marketing and sales personnel.

4.5 Comparison between CURE and literature’s requirements on a software maintenance model

As in the subchapter above we have made a comparison between CURE and the requirements regarding maintenance models. This time we compare the model with requirements from the literature. In the subchapters below we have described how we identified those requirements and why we have chosen to compare them with CURE. The areas covered in this subchapter are pointed out by one or several authors. The result of the comparison will be defined as level of match. There are three levels: high, medium and low. The match will be high if it is a perfect match, medium if the requirement is filled to some extent and low if no match is found within CURE. The results are summarized in a table in subchapter 4.5.3.

This subchapter has been reviewed and approved by the CURE-development team.

4.5.1 Literature’s requirements

To capture these requirements we had to read a lot of books and articles. None of them explicitly covered the question regarding ‘What is a good maintenance model?’ We therefore had to elicit the information we needed by noting what the authors pointed out to be most important or interesting.

The result is a list of requirements containing 13 different areas. The requirements concern the overall organization with its processes as well as single documents and guidelines. The literature studied for this part is Pigoski, Brandt, Bergvall, Takang and Révay. They are all listed in more detail in the reference chapter (chapter 8).
4.5.2  The Comparison with CURE
This comparison is a complement to the comparison above to fully evaluate CURE as a maintenance model. As in chapter 1, we use the internal CURE material from SchlumbergerSema.

4.5.2.1 Defined and documented routines (e.g. Decision routines, request handling and error reporting routines)
The area that most of the authors to some extent mentioned as important was that the model used by the maintenance organization is clearly defined and documented. In contrast to the other authors, Révay and Takang meant that the process has to be defined in several steps and phases to provide a “road map” for the maintenance project. Révay also means that an iterative process is preferable.

CURE provides a structured process in several steps. Each step is defined and supported by guidelines and templates. CURE is also iterative since it is divided into maintenance periods.

Level of match: High

4.5.2.2 Flexible process that can be tailorized to fit the situation
All authors agree that a software system is always changing. Therefore Takang, Révay and Bergvall points out that a model has to be tailorized to fit the situation. A static model created for today is soon to be obsolete.

CURE can be thought of as most flexible since it is very generally defined. CURE also provides several suggestions for organizational structure. Since CURE can be seen as a collection of guidelines it is easy to select the appropriate level of the model.

Level of match: High

4.5.2.3 Standardized documentation templates
Order and structure is always spoken of as necessary for a successful project of any kind. Therefore it is not that surprising to find Pigoski and Révay to point this out as necessary. To gain this they believe that all the documents produced within maintenance organizations should follow an agreed standard. If the model itself does not provide this, much time might be spent on creating standards.

CURE provides a lot of clear and well-defined templates to ease the creation of various documents. If these templates are followed there will be no problems in keeping the consistency, structure and standard between the various documents. Among others, CURE provides templates such as: maintenance contract, maintenance plan, period-ending report, transition plan and meeting protocol.

Level of match: High

4.5.2.4 Maintenance plan
According to authors of software maintenance literature such as Pigoski and Révay, the maintenance plan is one of the most central documents. That is why the need for support in the construction of a plan is important.

Since CURE is divided into maintenance periods, a maintenance plan will be created/modified at regular intervals. CURE provides templates for the creation of a maintenance plan at a detailed level.

Level of match: High
4.5.2.5 Planning of the transition from development to maintenance

The transition in software maintenance is described as the phase where the newly developed system is handed over from the development organization to the maintenance organization.

Révay and Pigoski mention the importance of including the transition phase to the system lifecycle. Including the transition phase into the maintenance model will only provide a structured start of the maintenance project.

The transition phase is included in the CURE system lifecycle and defined in two phases, planning and transition.

**Level of match: High**

4.5.2.6 Separate planning/management organization

Both Révay and Pigoski believe that the best way to manage a maintenance project is to have a separate group for activities regarding planning and management.

CURE provides the concept of a steering committee. Participants of this committee are gathered from both the customer and supplier. The task of the steering committee is to define the boundaries in both work and resources for the maintenance period. The committee also has the final saying to extensive decisions.

**Level of match: High**

4.5.2.7 Capture the evolutionary nature of software system

Earlier in this report we have used the metaphor of maintaining cars, see subchapter 4.2.2. A major difference between cars and software is that a car does not evolve like a software system. After providing some kind of service to you car it has not changed much. It still has four wheels, a steering wheel and an engine. Software on the other side might evolve and change its structure at a greater extent. Takang, who means that this should not be forgotten, points this out.

Since CURE provides iterative maintenance periods, the evolution of the system is always regarded. All changes made during a period will be counted in during the creation of the maintenance plan of the next maintenance period.

**Level of match: High**

4.5.2.8 Individual-independent project

With individual-independent we mean that a role or task should not be dependent on a specific employee with unique skills and experience. More than one employee should possess those skills and experiences.

Révay mentions an area that none of the other authors discuss. Révay identifies individual-dependency of a maintenance project as a drawback. He means that the maintenance project should ensure that no competence is lost if a project member should leave the project. A good maintenance model should provide a tool for securing the competence within the organization.

CURE provides the organization with a template for an educational plan where the steering committee shall decided how many persons that should be educated in different areas. CURE also provides a template for resource contracts where it’s defined how much resources the maintenance organization can demand from the line-organization. Other tools within CURE on this subject are maintenance journals and maintenance reports.

**Level of match: Medium**
4.5.2.9 Follow-ups
In order to make more accurate future maintenance plans the current plan has to be followed up. Révay points this out. Not only the maintenance plan has to be followed-up. The continuous work has to be followed-up in order to keep track of changes, used resources and cost.

Using period-ending reports and other reports defined in CURE will structure the follow-up procedure. The follow-up issue is also to be defined as a section in the maintenance report according to CURE.

**Level of match:** High

4.5.2.10 Project structured organization with documented goals
According to Révay and Bergvall, a maintenance project should be run as a separate project with own goals. The goal should be defined both short term and long term.

CURE can be regarded as a single standing project since it is defined with its own management, goals and plans. These goals and plans can very much be in line with the overall business goals and plans but should also be defined as system/project specific goals. CURE also points out that the goals should be long-, short-term and defined in the maintenance plan.

**Level of match:** High

4.5.2.11 Responsibilities/roles/organization defined and documented
Both Bergvall and Brandt have spoken for structure and order and not least the requirement of well defined roles, organizations and responsibility definitions. These definitions should be well documented to prevent misunderstandings and give all involved a good picture of the divided responsibilities.

Regarding organizations, CURE provides several defined role structures. Each structure has its own benefits and drawbacks. Because of the variations it should not be too hard to find one that fits the existing organization with only minor modifications. The roles listed by CURE are somewhat vague. That is one small drawback for CURE. The responsibilities of each role on the other hand are well defined by a well-structured responsibility matrix.

**Level of match:** High

4.5.2.12 Business-friendliness
Bergvall dedicated a complete chapter to this area. Bergvall means that in order to work professionally towards customers the model used by the supplier has to have some degree of business-friendliness. By this she means that both parts have to be satisfied with the deal and not feel cheated. Just adding fancy titles and roles does not solve this.

CURE is maybe not as well packaged as one could wish for. On the other hand the model is quite easy to grasp generally and that could provide some degree of comfort to the model. Because of this comfort the model might help the supplier to gain more long-term customers since they might feel safe and secure working with the supplier.

**Level of match:** Medium

4.5.2.13 Project placement and project boundaries
The last area identified by Bergvall and Révay is the positioning of the project within the organization and the need for clear project boundaries. It is important to know the boundaries for the project both in budget as well as types of tasks.
As mentioned above CURE is not regarded as a well-packaged model to the customer. Internally within SchlumbergerSema on the other hand CURE might be seen as a better-packed model. The reason is the experience of the user. Within CURE it is the responsibility of the steering committee to determine what is maintenance and what is further development.

**Level of match:** Medium

4.5.3 Result

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Level of match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routines</td>
<td>High</td>
</tr>
<tr>
<td>Tailorization</td>
<td>High</td>
</tr>
<tr>
<td>Templates</td>
<td>High</td>
</tr>
<tr>
<td>Maintenance plan</td>
<td>High</td>
</tr>
<tr>
<td>Transition</td>
<td>High</td>
</tr>
<tr>
<td>Planning organization</td>
<td>High</td>
</tr>
<tr>
<td>System nature</td>
<td>High</td>
</tr>
<tr>
<td>Individ independency</td>
<td>Medium</td>
</tr>
<tr>
<td>Follow-ups</td>
<td>High</td>
</tr>
<tr>
<td>Organization</td>
<td>High</td>
</tr>
<tr>
<td>Clear role descriptions</td>
<td>High</td>
</tr>
<tr>
<td>Package/ business friendliness</td>
<td>Medium</td>
</tr>
<tr>
<td>Project boundaries</td>
<td>Medium</td>
</tr>
</tbody>
</table>

*Table 4.4 – Overview of the comparison*

The comparison above shows that CURE meets most of the requirements that authors of software maintenance literature puts on a software maintenance model. Most requirements were general and about defining and documenting routines, documents and roles. Some of the requirements were however not fully met by CURE. Those requirements were; clear project boundaries, routines for securing competence within the project and the package of CURE towards customers.

With the result of 10 requirements met against 3 not fully met, we would say that CURE is a model that several of these authors should find appropriate to use.

4.6 Comparison between case study’s requirements and literature’s requirements on a software maintenance model

In this subchapter we have compared the requirements on a software maintenance model stated by the literature and the case study. The areas covered in this subchapter are derived from the list of requirements from the case study and the literature in the subchapter 4.4 and 4.5 above.
4.6.1 The reasons for comparing case study’s requirements with literature’s requirements

The reasons for this comparison are to broaden the analysis and to identify if and why the two sources of requirements differ. It is also interesting to find out which type of model requirements that differ and which are similar.

4.6.1.1 Characteristics of the case study’s requirements on a software maintenance model

The characteristics of the requirement areas mentioned by people in the industry are that they are all detailed. The viewpoint of the areas has in almost all cases arisen from personal experience of real life problems. Individuals working with maintenance easily connect the problems within the industry to what they are missing in the maintenance model.

All the requirement areas identified by the case study could be grouped together and be described as activities to create a smoother working process within software maintenance.

4.6.1.2 Characteristics of the literature’s requirements on a software maintenance model

The characteristics of the requirement areas mentioned in the literature regarding software maintenance are all comprehensive and all embracing. The areas are identified through scientific studies within the industry and the working experience of the authors.

All the requirement areas identified in the literature could be grouped together and be described as activities to create more order and structure within software maintenance.

4.6.1.3 Similarities

When we reviewed the two lists of requirement areas we identified some similarities. Those five are:

- Business-friendliness within maintenance is mentioned in the literature and well-packaged maintenance services are mentioned from the case study. These two aspects are very closely connected since a well-packaged service is easier to sell and market, therefore it’s also business-friendly.

- Defined and documented maintenance routines are mentioned in the literature and this can easily be compared with the requirement from the case study regarding an efficient change request system as one of those routines. A change request and error reporting systems follows well-defined and documented routines.

- The responsibilities, roles and organization should be defined and documented is another requirement from the literature that is also found in the needs from the case study as well-defined and clear role descriptions.

- The literature indicates that the maintenance process shall be flexible so that it can be tailored to fit the situation of the organization. The interviewees require appropriate level of administration and bureaucracy, which are characteristics of a flexible maintenance process that could fit most organizations and also could change over time.

- Standardized document templates are mentioned in many literature sources and described as one of the most useful tools used in the industry.

4.6.1.4 Differences

About half of the requirement areas in the two lists could not be interconnected because they are created with different viewpoints. The requirements from the literature are identified with
a scientific viewpoint reviewing the entire domain and making sure that all parts of the model works at all times. The requirements from the case study are identified with a problem-solving viewpoint as a proposed solution to a problem that someone has been caught in.

4.6.2 Results of the comparison between case study’s requirements and literature’s requirements on a software maintenance model

The case study is more likely to identify urgent matters as tools for the daily work, guidelines about how to get started with the model and structured communication with the customer. These kinds of requirements are good to demand from a model in order for the model to contribute to an easier working environment for the involved personnel.

The kinds of requirements that are not presented by the case study are of the long-term kind e.g. individual-independent roles etc. Those long-term kinds of requirements are omitted by the case study due to that they mostly look for profits or benefits in the short term. The long-term requirements are instead presented in the literature where the authors have many years of experience working with maintenance. The authors of the maintenance literature have been able to identify the long-term requirements over a longer period of time and in many cases they have worked in many different maintenance organizations and even numerous corporations.

4.7 Comparison between CURE and standards

Both the ISO standard and the IEEE standard are described in more detail in chapter 1. Both comparisons below are made on a high level, meaning that we do not look at any details.

4.7.1 The Comparison with IEEE

4.7.1.1 The process shall contain seven defined steps

The IEEE standard 1219 describes seven phases that should be defined in order to work according to the standard. These phases are problem identification, analysis, design, implementation, system testing, acceptance testing and delivery.

Within CURE we find some steps that correspond to these phases:

<table>
<thead>
<tr>
<th>IEEE</th>
<th>CURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem identification</td>
<td>Order</td>
</tr>
<tr>
<td>Analysis</td>
<td>Preparation, decision</td>
</tr>
<tr>
<td>Design</td>
<td>Implementation</td>
</tr>
<tr>
<td>Implementation</td>
<td>Implementation</td>
</tr>
<tr>
<td>System testing</td>
<td>Implementation</td>
</tr>
<tr>
<td>Acceptance testing</td>
<td>Delivery testing, delivery approval</td>
</tr>
<tr>
<td>Delivery</td>
<td>Production</td>
</tr>
</tbody>
</table>

Table 4.5 – Overview of the comparison

As we see it, CURE includes all the steps to some extent at a high level. If we were to look more at the details within each phase, we would probably find that IEEE and CURE do not match. This conclusion is drawn since IEEE requires a lot of reports, reviews and matrices, while CURE is more of a non-bureaucratic model with fewer reports, reviews and matrices. Which one that is the best choice is not determined within this research.
4.7.2 The Comparison with ISO 9000-3 section 5.10

4.7.2.1 Maintenance routines
ISO defines three categories of maintenance activities that should be a part of the maintenance routines. These three activities are problem resolution, interface modification and further development.

We find that CURE contains all these three activities. This seems obvious to us since problem resolution and further development is the core activities of maintenance and that interface modifications are sometimes necessary due to the other two activities.

4.7.2.2 Documentation
ISO recommends that several various documents should exist. These documents are e.g. Maintenance contract and maintenance plan. All maintenance activities should also be documented. However, ISO do not in detail describe what these documents should contain but both the customer and the supplier should agree upon all documents.

CURE provides good templates for both the maintenance contract and the maintenance plan. If the maintenance journals provided by CURE are used properly, then all maintenance activities will be well documented.

4.7.2.3 Organization
ISO states that the maintenance organization should contain representatives from both the customer and the supplier. The organization shall also be flexible for urgent decisions.

CURE provides several organization structures with representatives from both sides. CURE has also defined a shortcut in the maintenance request process in order to gain flexibility to manage urgent requests.

4.7.3 Result
The result of this comparison is that CURE is more like ISO than IEEE. This is quite obvious even though the comparisons were made without respect for details.

The match between IEEE and CURE could be interesting for further research since the standard is very detailed.

The match between ISO and CURE were quite good. ISO, however, is a very general standard that comprises many solutions. A further development of CURE in order to certify the model according to the standard would be a good idea.

4.8 Summary

4.8.1 Case study
- The case study consisted of 14 persons selected from five different projects. All five projects were chosen in mutual agreement with SchlumbergerSema from within their organization. The roles were both from the customer and supplier side.
- During the interviews we tried to collect opinions about both the interviewee’s knowledge and apprehension of maintenance as well as ideas about how to improve its efficiency and the public apprehension of maintenance.

4.8.2 Results regarding general maintenance issues
- As we started analyzing the answers from the interviews, we had to categorize the answers into groups.
Regarding the need for maintenance, almost all interviewee’s found it necessary to some extent at all times. As we asked what factors that affected the needs, we received 27 answers from the 14 interviewed persons. We categorized the answer with in four groups: size aspects (12 answers), critical aspects (7 answers), economical aspects (6 answers) and security aspects (6 answers).

Some interviewees mentioned a metaphor including car maintenance. Either you have the knowledge and resources to maintain the car yourself or you will have to let the car repair shop do it for you. All customers should feel that maintenance of systems is as natural as the service of the car.

When asked to define maintenance, most interviewee’s found it hard. The base of all answers was mostly the same although the words used were different.

Only half of all interviewees were old enough to actually have worked during the 80’s. Of this half there were two who meant that maintenance has changed since the 80’s. The changes were regarding the need, requirements, technology and perhaps maintenance has not been able to develop as much as software development since then.

Regarding the apprehension of maintenance only 12% found it boring. The solutions to improve the general apprehension were several. Some of them were: changing the terminology, increase salary and benefits, more support and commitment from management and further personal development.

Only one fourth of the interviewees had not heard of or had any knowledge of CURE. Most of them who knew CURE had only good comments. Some drawbacks were mentioned though such as the need for tailorization and the lacking of computer based tools.

### 4.8.3 CURE vs. the results of the case study

The ten identified areas from the case study are:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Level of match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear role descriptions</td>
<td>High</td>
</tr>
<tr>
<td>Routines</td>
<td>Medium</td>
</tr>
<tr>
<td>Templates</td>
<td>High</td>
</tr>
<tr>
<td>Request traceability</td>
<td>Medium</td>
</tr>
<tr>
<td>Guidelines</td>
<td>Medium</td>
</tr>
<tr>
<td>Information cannels</td>
<td>High</td>
</tr>
<tr>
<td>Bureaucracy</td>
<td>High</td>
</tr>
<tr>
<td>Administration</td>
<td>High</td>
</tr>
<tr>
<td>Sales process</td>
<td>Medium</td>
</tr>
<tr>
<td>Package/ business friendliness</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 4.6 – Overview of the comparison

As the table shows, CURE fills most of the requirements. However the requirement of packaging and business friendliness is not filled.

### 4.8.4 CURE vs. literature

Within this study, 13 different requirements were identified. Some of them are similar to the requirements from the case study. The requirements found are:
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Level of match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routines</td>
<td>High</td>
</tr>
<tr>
<td>Tailorization</td>
<td>High</td>
</tr>
<tr>
<td>Templates</td>
<td>High</td>
</tr>
<tr>
<td>Maintenance plan</td>
<td>High</td>
</tr>
<tr>
<td>Transition</td>
<td>High</td>
</tr>
<tr>
<td>Planning organization</td>
<td>High</td>
</tr>
<tr>
<td>System nature</td>
<td>High</td>
</tr>
<tr>
<td>Individ independency</td>
<td>Medium</td>
</tr>
<tr>
<td>Follow-ups</td>
<td>High</td>
</tr>
<tr>
<td>Organization</td>
<td>High</td>
</tr>
<tr>
<td>Clear role descriptions</td>
<td>High</td>
</tr>
<tr>
<td>Package/business friendliness</td>
<td>Medium</td>
</tr>
<tr>
<td>Project boundaries</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 4.7 – Overview of the comparison

The table shows that CURE meets most of the requirements. Some of the requirements were however not fully met by CURE such as clear project boundaries, routines for securing competence within the project and the packaging of CURE. It is our opinion that CURE is a model that several of these literature authors would find appropriate to use.

4.8.5 The case study vs. literature

- There were both differences and similarities between the both sets of requirements.
- The similarities were packaging/business friendliness, defined routines, roles/organization, tailorization and documentation.
- The differences were mostly that the requirements from the literature were based upon scientific history as the case study requirements were based on experienced problems.

4.8.6 CURE vs. standards

- The two standards that we examined were ISO 9000-3 and IEEE Std 1219.
- At a high level CURE matched the standards pretty well. Although if we were to evaluate CURE in detail according to IEEE, we believe that it would not fit the standard. ISO on the other hand that does not provide these details would fit better.
- We suggest that the developers of CURE should try to modify CURE in order to certificate the model according to the standard. That would give the model a good mark of quality.
5 RECOMMENDATIONS

5.1 Introduction
In this chapter we will give our recommendations within the domain of software maintenance based on the knowledge that we have gained from literature studies and from the industry through the case study.

The recommendations in this chapter are divided in three categories: general, SchlumbergerSema-specific and CURE-specific. The reason for the division is that we feel that some recommendations are suitable for all in the field of software maintenance, some are SchlumbergerSema-specific and some are recommendations regarding the maintenance model CURE and its development team.

The subchapter covering the general software maintenance recommendations contains information regarding: what to think about, what to avoid, what to watch out for and benefits gained from conducting software maintenance in a proper way.

The subchapter with recommendations to SchlumbergerSema and their maintenance work covers: choice of maintenance model, tailorization of CURE, benefits with CURE, drawbacks with CURE.

The subchapter with recommendations regarding the maintenance model CURE contains information for users of CURE and the CURE-development team. This subchapter covers the areas: benefits, drawbacks, further development and packaging.

5.2 General recommendations

5.2.1 Definition of Software Maintenance
In order to build a good basis of the customer-supplier relationship it is essential to define the terms involved in the relationship. In the case of making a software maintenance agreement it is therefore important to define the term software maintenance and make sure that everyone involved understands and share the same definition.

We define software maintenance in the same way as within the maintenance model CURE. The definition is formulated by the RSF (Referensgruppen för SystemFörsörjning) and is:

“Maintenance is all the activities performed in order to administrate and manage a system in operation, in a matter that during its lifetime it efficiently contributes to achieve the goals of the main business.” More about the definition of maintenance is found in subchapter 1.2.

5.2.2 How to handle Software Maintenance
This subchapter contains our general recommendations about how to handle software maintenance.

- In order to success as a supplier with software maintenance it must be an open attitude around it and the costs for it should not be hidden. The sales process for system development must be integrated with the sales process of maintenance services. See more in subchapter 4.4.2.10.

- The services, costs and activities within software maintenance must be packaged for marketing and sales. See more in subchapter 4.4.2.9.

- Use only motivated routines and make everyone involved aware about the benefits gained by working according to the procedures. If the understanding of each individual’s contribution to the success of the maintenance work is high the
motivation for performing the routines will also be high. See more in subchapter 4.4.2.7.

- Defined and documented routines and information channels will increase the level of order and structure in the daily work and therefore make the maintenance organization more efficient. See more in subchapter 4.4.2.6.

- Templates are a perfect tool that will bring order, structure and speed within the maintenance work. See more in subchapter 4.4.2.3.

- We believe that the most efficient way to handle software maintenance is to choose a model for a long-term perspective. If a new model or solution is used for every project, the solution will become very dependent on a few individuals that are almost irreplaceable. The individuals involved will possess knowledge about the solution, its terminology, agreement and routines that nobody else has. See more in subchapter 4.5.2.8.

- Do not hide software maintenance in the sales process of system development. This will only lead to trouble down the road when the customer finds out that it will not only cost money for the development but also afterwards. The best approach is to offer both the system development and an appropriate level of software maintenance at the start of the relationship between the supplier and the customer. See more in subchapter 4.4.2.9.

- There is a risk of a maintenance model being too bureaucratic with a lot of unmotivated routines, structures and documentation-habits. If a maintenance model is too thorough with tasks that the personnel conducting them cannot see the benefits from the project, customer and model will receive a lot of bad credit. See more in subchapter 4.4.2.8.

5.2.3 Pitfalls within Software Maintenance

There are some specific pitfalls within software maintenance that it’s very important to look out for.

One is that the supplier and the customer don’t share the same view of software maintenance in general and the project in particular. The most common difference within this particular area is that the customer doesn’t see any need of software maintenance at all due to the fact that they have purchased a working system. In this case the supplier wants some level of annual or monthly payment to cover costs for support, knowledge management and test environment.

This kind of difference in apprehension of the level of maintenance needed is often hard to solve after the system is delivered and should therefore be settled prior to delivery. A solution to this pitfall is to integrate software maintenance services in the sales process of system development. See more in subchapter 4.4.2.9.

Another pitfall that we have identified is that the model becomes just a nice-looking bunch of paper and is never really used. Many activities in the daily work are performed in an ad hoc kind of way and the routines defined in the model are not followed.

One solution to this pitfall is to educate all involved personnel in the model and to ensure that everyone understands the benefits by working according to the model.

5.2.4 Benefits of a well-functioning Software Maintenance

Well-functioning software maintenance is a great quality mark for the supplier. It is a huge sign of supplier-maturity to be able to show their customers how they would like the maintenance to be arranged together with examples of other well-functioning maintenance
projects. The supplier can also arrange courses in the software maintenance model towards the customers that will increase the supplier’s sign of quality and maturity.

If the maintenance works smoothly and efficiently the personnel will enjoy their positions within the maintenance organization more. This will increase their job satisfaction that will lead to a lower employee turnover and less trouble in the maintenance work. A lowered employee turnover and increased job satisfaction will, in our opinion, give maintenance work a higher status.

Well-functioning software maintenance will encourage a higher level of participation in the development of the maintenance model and its refinement from the personnel and lead to a greater job satisfaction.

5.2.5 To choose a Software Maintenance Model
This subchapter covers two significant questions: to have a maintenance model or not, to have one in a long-term perspective or a short-time perspective.

The most significant question is of course the problem regarding working with a model or working ad hoc with no model at all.

5.2.5.1 How to choose a model or no model
Even though it costs money, time and effort to implement and follow a maintenance model the benefits are numerous: structure, maturity, efficiency etc. There are in our opinion a few factors that would emphasize the use of no model at all, but first the reasons for using a model.

In our opinion the factors that would emphasize the usage of a model are size, lifetime, complexity and external dependencies. We believe that a model shall be used when a system is anything but very tiny, when the lifecycle of the system is not very short or when the system is complex or when a combination of those factors occurs. Another factor that affects our recommendation regarding having a maintenance model is the number of dependencies such as users or third parties. If a system has more than a handful of users or external parties dependent of the system a maintenance model should in our opinion be used to emphasize order, structure and routines.

In some special cases we recommend the usage of no model at all. In these cases the maintenance work will be solved in an ad hoc matter with temporary solutions every time a problem occurs. One reason for this approach is if the size of the system is tiny and the cost for setting up a maintenance organization is greater than the system itself. In this type of case we believe that it is a smoother solution for both parties that a consultant is paid for by the hour.

Another reason for choosing to use no model at all is if the lifetime of the system is very short. An example of such a system is a system for a one-day event like a results-tracking system for a sport event. See more in subchapter 4.2.2.

5.2.5.2 How to choose a model for the long-term perspective or the short-term perspective
The second significant question regarding software maintenance is the choice of a model in a long-term or short-term perspective.

With a long-term perspective we mean that an organization chooses a model that they will work according to for a couple of years. In a short-term perspective we mean that an organization will choose a maintenance model individually for each project going from development to maintenance.

The benefits with the long-term perspective are:
- A sign of quality as a supplier to have a defined maintenance routine.
- Easy for personnel to move between projects.
- Easy for sales personnel to have a good understanding about what they are selling.
- Efficiency within the maintenance organization due to established routines and processes.

If an organization has decided to choose a model for a long-term commitment they should choose a model that is very general in order to fit most customers, systems and projects. The chosen general model can be tailorized to fit every project and customer organization.

Another aspect that is important when an organization choose a model for a long-term commitment is education in the model’s theory and other material for all involved personnel. One person should be appointed to be responsible for the maintenance model and its refinement within the organization. This model-specialist should keep track of all tailorizations done to the model in order to recommend the tailorizations to future projects.

The benefits of a short-term perspective are:
- Less supplier influence on the customer
- The maintenance process will be similar to prior customer maintenance projects
- Appropriate solution for each maintenance project

If a supplier has decided to choose a model for each maintenance project in mutual agreement with the customer they should choose a model that is more specific and comprises specific tools and templates for the actual system.

If a model is chosen for each project it could be more suited for the specific system and less for the organization.

### 5.3 SchlumbergerSema-specific recommendations

#### 5.3.1 Selection of long-term or short-term choice of maintenance model

After considering the subchapter 5.2.5.2 above, we would suggest the selection of a long-term model. Since SchlumbergerSema has provided themselves with CURE it is most natural that they use it. CURE is general enough to fit a wide range of customers and projects with minor tailorizations. The benefits of choosing a long-term model above a short-term seems to be greater. Not only does a long-term model save resources in form of education costs and efficiency losses, but it also provides a uniformity allowing staff to go from one project to another with less transition time. This uniformity also provides a good base for collection of statistical data and opinions in order to develop the model and the efficiency of the maintenance projects. As we see it, there are far more benefits than drawbacks from using a long-term model. See more in subchapter 4.2.2 and 4.2.3.

The drawbacks are however inevitable. The model does not always suit every customer. In some cases the project has to come up with their own solution. This could be due to the characteristics of the system, rules and regulations or the internal structure of the customer’s organization. One might think that the uniformity of all projects would damp the development of the model as no new routines are made or no new tools created. But the development of the model will always continue as long as tailorizations have to be made. Every customer or supplier that encounters the model will probably question various parts. If these opinions are collected as mentioned above there will be a good flow of new ideas. See more in subchapter 4.2.5.
If a long-term model is chosen, it must be maintained itself. Since the model is developed by SchlumbergerSema themselves, the maintenance should be easier. The contact between the model developers and the model users ought to be easier and the interest in developing the model further should be greater.

5.3.2 Tailorization of CURE

Both the authors of software maintenance literature and the people actually working within software maintenance organizations agree that no projects are exactly the same. Therefore it is important to find a model general enough to allow tailorization. CURE, that is a most general model, allows this tailorization necessary for the maintenance projects to run smoothly.

Our suggestion for tailorization will here answer three questions:

- Which parts of CURE are mandatory?
- Which parts of CURE are eligible?
- When should CURE be tailorized?

CURE consists of many various parts. The level of tailorization is illustrated in table 5.1 below.

<table>
<thead>
<tr>
<th>Part</th>
<th>Mandatory</th>
<th>Eligible</th>
<th>How?</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Life cycle</td>
<td>X</td>
<td></td>
<td>The life cycle could just as easy be linear instead of cyclic as CURE prescribes.</td>
<td>All projects do not have to result in the analysis of the business in order to develop a new system.</td>
</tr>
<tr>
<td>Process</td>
<td>X</td>
<td></td>
<td>The process is the backbone of the model. It should be kept as described by CURE: Transition (Planning, transition) and maintenance (change request, period ending, re-negotiations)</td>
<td>If the main process within CURE is kept within several projects, it is easy for project members to move between several projects. It also keeps the projects more comparable.</td>
</tr>
<tr>
<td>Contract</td>
<td>X</td>
<td></td>
<td>The structure of the contract should be the same and follow the contract template provided by CURE.</td>
<td>Not only does it give the salesman an easier task, but also eases the task of comparing projects in order to gain a better financial overview.</td>
</tr>
<tr>
<td>Plan</td>
<td>X</td>
<td></td>
<td>The structure of the maintenance plan should be the same and follow the maintenance plan template provided by CURE.</td>
<td>The benefits of having the same structure eases the work of concerned parts and specially those who are involved in other projects to some extent. It also ensures that the terminology within the project is the same.</td>
</tr>
<tr>
<td>Roles</td>
<td>X</td>
<td>The role names should be the same but which roles to use could be tailored to fit the current situation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---</td>
<td>------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Small projects do not need all roles and large projects or customers might need additional roles with specific responsibilities. Important is that no matter what roles are used; they should be well defined and documented.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistance</td>
<td>X</td>
<td>If the roles are tailorized in a way that roles are omitted or added, the responsibilities might have to be shifted around. If that is done it is important to ensure that no responsibility is omitted or duplicated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>As long as the roles are not set according to the CURE model, the responsibilities have to be tailored as well. This has to be done to ensure that no responsibilities are omitted or duplicated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td>X</td>
<td>The roles might be shifted around and new information channels might be added or omitted.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The tailorization of the organization follows by the tailorization of the roles. It could also be necessary due to the customers’ internal structure.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meetings</td>
<td>X</td>
<td>Meetings can be added or omitted. The content and participants of the meetings can be altered.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If roles or organization are tailorized and new channels are added or omitted the meetings might have to be tailorized concerning agenda, frequency, level of formality and participants.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Budget</td>
<td>X</td>
<td>The budget structure should be the same and follow the maintenance budget template provided by CURE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This part should be kept according to the template provided by CURE in order to trace resources within the supplier business and to give a better and more easily calculated base for financial prognoses.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change request</td>
<td>X</td>
<td>The change request process provided by CURE is simple, well defined and structured. It also allows shortcuts for emergency requests. It should be implemented as a whole.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>All maintenance projects should have a defined and structured process for change request handling. A structured process will, according to our study, ease the change request handling.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Follow-up</td>
<td>X</td>
<td>The amount of follow-ups and how it is done is well defined by CURE. Some minor projects might not need follow-ups.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The need for follow-ups is always present. However it should be at an appropriate level. Too much will result in unnecessary overhead.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
need all that and larger projects might need more due to additional departments, roles or other.

in boring bureaucracy and to little will result in losses of experience.

When should these tailorizations be used? We believe that the answer to that question is; whenever the basic CURE model does not fit the project. The amount of tailorization does not have to be huge every time a tailorization is made. In some projects only one or two minor changes might have to be made due to a specific circumstance such as customer organization, system characteristics or third part requirements. This does not mean that tailorizations should be done “just for fun”. Every tailorization has to be well considered since it might result in additional changes such as the relation between roles-responsibilities-organization.

5.3.3 Benefits from using CURE

If SchlumbergerSema would adopt CURE as a long-term model, there would be several benefits to gain. These benefits are discussed in this subchapter.

The greatest benefit of adopting CURE would be that it actually fulfills many of the requirements that people within the company as well as authors of software maintenance literature puts on a good software maintenance model. These requirements are discussed in more detail in chapter 4. This benefit is probably the most valuable since it satisfies the everyday requirements.

Other benefits are more of another kind. These benefits are more long-term such as statistical bases, evaluation and experiences. If CURE would be used within all or several projects within a company, it would create uniformity. Members of a software maintenance organization could easier move from one project to another without having to learn the model from scratch. People will not have to be educated in the model for every new project but merely enlightened about eventual tailorizations. See more in subchapter 4.5.2.8.

The usage of CURE within several projects will also provide a large statistical material and viewpoints about CURE. These two benefits are however not automatically gained. The follow-up guidelines of CURE have to be thoroughly conducted, the information has to be collected such as core data about the project and the opinions of the people actually using CURE. This information must also be analyzed and followed-up. Otherwise the collection and analyses will be in vain. The result of this can then be used to suggest further development of CURE and to make prognoses about efficiency, resources and financial budget.

5.3.4 Drawbacks from using CURE

Although there are benefits of using CURE there are also some drawbacks. It is not necessary that these drawbacks would not exist if CURE would not be used, but we feel that they are relevant to mention.

Most of the drawbacks of CURE could be omitted due to further development of the model. The main drawback as we see it is the fact that the model only provides procedures, guidelines and templates. What would have been a benefit of the model is the support of a computer-based tool for e.g. change requests handling. This issue was also mentioned by one of our interviewees. We believe that a tool like this could really ease that process. Another drawback that could be omitted by a further development by CURE is the fact that the model is not certified by any standard. The customer and others would recognize a certified model as a better model than one that is not. The certification is a quality sign. It gives the customer and probably the company managers, comfort in that this model is not just something
quickly thrown together. The third drawback of this kind is the fact that the model only exists in Swedish. If the future customer is English-spoken, the model has to be translated. It would be even more usable if it were available in other languages too. See more in subchapter 4.2.6.

5.4 CURE-specific recommendations

5.4.1 Benefits with CURE

There are many good things to say about CURE. The most positive thing is that it suits its purpose well. It meets many requirements from both the case study and the literature. This we have spoken of in the earlier chapters (chapter 4). Although it does not meet all the requirements we evaluate CURE to meet a majority of them. Another reason why CURE is appreciated by several that works within the software maintenance industry today is the amount of templates. The templates are a highly appreciated part of CURE. They provide structure as well as guidance both during the start-up phase of the maintenance project and during the actual maintenance process. Because of its general level, CURE is also easy to tailorize. The parts that we have identified as tailorizable are found in the table in subchapter 5.3.2. Last but not least since it has been used with success, it actually works.

CURE is today a licensed-based maintenance model where both the model and the education regarding the model are sold. CURE is cheap for large customers since the model is site-licensed in a matter that if an organization buys CURE they can use it for all their systems.

5.4.2 Drawbacks with CURE

The CURE maintenance model is expensive to buy for small customer organizations since it is site-licensed and meant to cover all software maintenance needs of the organization.

Since the solution to the drawbacks often is further development, we do not discuss this any more in this subchapter. Further development of CURE is covered in the next subchapter.

5.4.3 Further development of CURE

All of the drawbacks about CURE that we have encountered during this study have been evaluated. Below in table 5.2 some of them are described as suggestions to further development of CURE as a model.

<table>
<thead>
<tr>
<th>Improvement</th>
<th>What?</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role backup</td>
<td>Routines to ensure that the project is not dependent on one person with regard to his role. E.g. deputies to important roles.</td>
<td>If a person leaves the maintenance project, his place should be easy to fill. The project should not loose any time or resources due to this.</td>
</tr>
<tr>
<td>See subchapter 4.5.2.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Competence backup</td>
<td>Routines to secure the competence within the maintenance project. The routine should be general to cover all types of competence such as system knowledge, maintenance project knowledge or theoretical model knowledge.</td>
<td>This is just as important as the role backup. If a person should leave the maintenance project then his competence about the system, project or the model should be secured within the project. This does not only ease the role backup but also makes the project less individual dependent. Today, CURE has only a template for an educational plan.</td>
</tr>
<tr>
<td>See subchapter 4.5.2.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clear boundaries</td>
<td>Guidelines that helps determining whether a change request is a bug fix or a further development issue.</td>
<td>Since this was a problem identified within the literature and we thought that CURE did not meet this requirement it is brought up here. This would help both customers and suppliers to determine the level of effort needed. Perhaps it could be measured in terms of man-hours. The CURE recommendation is that the maintenance committee determines this.</td>
</tr>
<tr>
<td>Certification</td>
<td>Certification of CURE according to a standard.</td>
<td>In order to gain a sign of quality, CURE should be certified according to a standard such as ISO or IEEE. This will gain comfort in the model for all parts and raise it above other competitive models.</td>
</tr>
<tr>
<td>Translation</td>
<td>Translation of CURE into other languages than Swedish, e.g. English.</td>
<td>In order to apply the model on projects with a foreign customer or supplier the model has to be translated. This includes templates, model documentation and education. If it is to be done every time it will consume a lot of time and resources. Therefore a central translation will take less effort.</td>
</tr>
<tr>
<td>Tool</td>
<td>Computer-based tools for handling of change requests.</td>
<td>If the model should provide this kind of tool as a complement to the templates and processes, minimum effort will be spent on developing routines and tools for each project. The tool can also be made central and used by several projects and therefore lessen the costs of its development.</td>
</tr>
</tbody>
</table>

Table 5.2 – Recommendations for further development of CURE.
5.4.4 Package of CURE

Earlier we identified the package of CURE as somewhat weak. It can be divided into three different levels:

<table>
<thead>
<tr>
<th>Level</th>
<th>Target</th>
<th>What?</th>
<th>Why?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>The staff of SchlumbergerSema</td>
<td>The model is only packaged as a big lump.</td>
<td>Since the developers of CURE are located within the same company, the model seems more attractive and useful. It is packaged in a way that feels comfortable for the staff.</td>
</tr>
<tr>
<td>Educational</td>
<td>Suppliers</td>
<td>The CURE education is packaged in several levels. Education can be bought at various extents such as courses, hours or support.</td>
<td>Since it is packaged in levels, it is easy to get a grip of.</td>
</tr>
<tr>
<td>External</td>
<td>Salesmen and Customers</td>
<td>The model is only packaged as a big lump.</td>
<td>When selling CURE to a customer, it would ease the work of the salesman if there existed some kind of altered packages. The model could be packaged within a few standard solutions such as small, medium or large. Then the tailorizations would lessen and a customer will get what they want faster.</td>
</tr>
</tbody>
</table>

Table 5.3 – Packaging of CURE

Other issues about the package of CURE are that it would ease the selling of CURE if sales tools existed. By tools we mean presentation material (preferably showing the small, medium, large packages described in table 5.3), success stories to base arguments on and finally statistics. Provided success stories and statistics the salesman can easier make the customer understand the benefits and flexibility of CURE.

5.5 Summary of Recommendations

In this chapter we have given recommendations about software maintenance in general, SchlumbergerSema-specific and CURE-specific. The general recommendations cover areas such as maintenance tips, pitfalls, benefits and how to choose a maintenance model.

A summary of the general recommendations:

- An open attitude towards maintenance with visible and clear presentation of costs.
- The maintenance services must be packaged for marketing and sales.
- Use only motivated routines and procedures.
- Choose a model for a long-term perspective.
- Use and tailorize templates.
- Define and document information channels.
A summary of the pitfalls:

- The supplier and the customer do not share the same view of software maintenance.
- The maintenance model is not used in reality, just a nice-looking bunch of paper in a folder.

A summary of the benefits of well-functioning software maintenance:

- Sign of quality and maturity of the supplier.
- Increased job satisfaction among the employees.
- A higher status of maintenance work.

A summary of how to choose a maintenance model:

- The question regarding to have a model or no model at all. It’s our recommendation that you should have a model in all cases except for very tiny systems, systems with a short lifetime and stand-alone systems.
- The question regarding having a model in the short-term or the long-term perspective. The benefits with the long-term perspective are that it will enable personnel to move easier between projects, a sign of supplier quality and maturity, and efficiency due to well worked-in routines and processes. The benefits of a short-term perspective are that the maintenance process will be more similar to prior customer projects and a more appropriate solution for each maintenance project.

The SchlumbergerSema recommendations investigate how SchlumbergerSema shall choose a maintenance model and how they can tailorize it for their different needs. This subchapter also covers why/why not SchlumbergerSema shall choose CURE and the related benefits and drawbacks with such a decision.

- We recommend SchlumbergerSema to choose a model for the long-term perspective.
- We recommend that SchlumbergerSema choose the model CURE.
- We recommend what parts of CURE that can be tailorized by SchlumbergerSema.

The subchapter with recommendations regarding CURE is aimed at personnel working according to the maintenance model CURE and its development team. This subchapter identifies benefits and drawbacks with CURE and recommendations for future improvements to the model.

- We have identified that CURE fulfills most of the requirements from the industry and other literature.
- We see it as a drawback that CURE is not yet standardized by some authority e.g. IEEE, ISO. We recommend that the CURE-development team takes this action in order to spread the model and increase it’s sign of quality.
- We recommend that CURE should be packaged for sale and marketing.
6 CONCLUSION

The first two subchapters in this chapter are answers to our research question and objectives stated in the research proposal. The research proposal was formulated before this master thesis project begun. The next subchapters are our conclusions of this master thesis and finally recommendations for future research in the area of software maintenance.

6.1 Answer to research questions

In this subchapter we will answer the research questions that we posted in the research proposal prior to start this research project. The answers are based on the responses from the interviews and evaluation of the other material in the case study such as review of maintenance agreements etc.

6.1.1 Have the system maintenance knowledge from 1980’s been forgotten?

Only half of all interviewees were old enough to actually have worked during the 80’s. Of this half there were two who meant that maintenance has changed since the 80’s. The changes were regarding the need, requirements and technology. Perhaps maintenance has not been able to develop as much as software development since then. See subchapter 4.2.4.

6.1.2 Is there a difference in the need or characteristics of system maintenance depending on the code type of the system (old Cobol systems vs. modern object oriented systems)?

Yes, the need is greater due to several reasons:

- In the older types of systems every application was a separate solution and some users used one and others used another. Today most applications are integrated with each other in complex systems with a wide variety of users. The customers often do not understand that a requested change in one part of a system affects the entire solution. This fact puts pressure on the maintenance organization to coordinate changes and maintain an overall system view.

- Testing is more difficult due to system integration. Due to often very complex system integration, requirements on testing and an extensive test environment has increased. When errors happen it is also harder to locate the source of the error in a complex and integrated client/server system compared to an older and more structured system. These facts affects maintenance in a way that more time and resources has to be spent in testing and in the efforts of locating error sources than in older system types.

- Today most customers have higher expectations on fast deliveries. In the old days computer systems were considered to be something really advanced and therefore the customers had no problem with longer development times. This reason for the change of maintenance need is not directly connected to the change of system technologies but instead well connected to the changed look upon computer systems over time.

- The transition from older structured systems to more complex and integrated systems has developed the user environment tremendously with multiple applications and windows running simultaneously. In a modern client/server system the user often has full flexibility to jump around in the application or even between applications. These new abilities introduce new problems for the maintenance
organization regarding support. It is more difficult for the support personnel to understand what the user has done or where he or she is within the system.

6.1.3 Which system maintenance methodology is best suited for different code types?

The difference in the maintenance work between the two system types discussed in subchapter 6.1.2 could still be comprised within the same maintenance model. As we see it the maintenance models are very general and could therefore be used with both system types. The differences are in the level of effort spent on different parts in the model but all the activities were needed in the older systems and are still needed in more modern object oriented client server systems.

6.1.4 Is there a difference in the need or characteristics of system maintenance depending of the customer type or organization (public vs. private or customer market e.g. telecom, business administration etc.)?

The most important factors when determining the level of need for software maintenance are the business criticalness of the system and the size of the system. If business stops within the organization when the system stops, all interviewees agree that this system needs the highest level of software maintenance. See subchapter 4.2.2.

Companies in the public sector tends to be more required to follow laws, standards and regulations than companies in the private sector. The difference in characteristics of system maintenance between the two categories is that companies in the public sector tailorize the maintenance model to follow specific laws and regulations for public sector companies. See subchapter 5.3.2.

We have not identified any significant difference regarding the customer market e.g. telecom, business administration etc. The factors that affect the need or characteristics of maintenance are as described in chapter 4: size, business criticalness and complexity of the system.

6.1.5 Which system maintenance methodology is best suited for different organizations (see above)?

Companies in the public sector tend to choose a model that will comply with public sector laws and regulations. The model for public sector companies shall also fit all the systems within the organization and therefore be general. See subchapter 5.2.5.1 and 5.2.5.2.

Companies in private sectors tend to choose models that will benefit the specific system with tools, templates and routines. The private sector companies tend to choose a model for each system with more specific tools for the actual system.

6.1.6 How can maintenance models be tailorized?

The most frequent tailorization on maintenance models used within the case study was refinement of role descriptions and the introduction of new roles. The refinement was done in order to make the roles fit more to how the rest of the organization works or to fit in more with earlier maintenance work within the organization. See subchapter 5.3.2.

Some maintenance organizations had introduced new roles e.g. a security manager. An extra role beside the recommended ones within the model is often introduced due to organizational
history or that the specific task is very extensive or important that an extra employee is needed.

In a few cases the responsibilities connected to each role were modified but this is very closely related to the tailorization of role descriptions above.

6.1.7 How can maintenance models be adapted (in full, as guidelines)?

A maintenance model can be adapted at several levels. During this research project we have seen models adapted up to almost 100% and in some cases just as a reference material.

It is our opinion that the more of the model used the better for the maintenance organization. The model doesn’t do much good on just paper in a folder. Build a tailorization of the model for the specific maintenance project and follow it as much as possible. Analyze what parts of the tailorized model that are not used properly and re-tailorize to obtain a fully functioning maintenance model. See subchapter 5.2.3.

6.2 Objectives

6.2.1 Identify existing maintenance methodologies available in the industry and document their characteristics by comparison.

At the start of our research we identified several different maintenance methodologies such as CURE, RDF-model and the English-model. After having studied them we realized that they were very similar. Therefore we felt that a comparison of the models would not give any new findings. Sure they differ some but only at a detailed level.

6.2.2 Identify the possibility to tailorize, the maintenance methodologies found, for the public sector.

Since this research was done in cooperation with SchlumbergerSema, we concentrated mainly on the tailorization of CURE. The possibilities to tailorization of this model are described in subchapter 4.5.2.2, 4.5.2.7 and 5.3.2.

6.2.3 Identify whether the developers and the customers share the same view of software maintenance.

After conducting the interviews we found that this issue were not of any greater interest. It is most rare to find an organization where all involved share exactly the same view. Therefore we decided not to put any effort into finding out more about this.

6.2.4 Identify the viewpoint of software maintenance by standards organizations such as ISO.

Those standards that we concentrated on were ISO 9000-3 and IEEE Std 1219. They both concerned software maintenance. Both standards are described within subchapter 1.5. Other mentioned standards that did not get as much space as these two were ISO 9001 and IEEE Std 1044 and 730. The reason for this were that they did not directly concern maintenance but only minor parts or very briefly. ISO 9000-3 is also an implementation on software engineering of the more general ISO 9001. Therefore it felt overlapping to discuss them both in detail.

6.2.5 Present a software maintenance package suitable to the market.

The maintenance package that we studied was the package of CURE. Since we found that it almost did not exist, we gave the CURE development team some suggestions of how to
package CURE for various purposes. This is described more in subchapter 5.4.4. The issue was also raised within some interviews and the package of CURE was evaluated in subchapter 4.4.2.9, 4.4.2.10 and 4.5.2.12.

6.2.6 Present a suitable usage, tailoring and modification of CURE (SchlumbergerSema’s internal initiative of software maintenance) mainly for customers in the public sector.

As described above we have evaluated CURE regarding the possibility to tailorize the model. Since we did not find any major differences between businesses within the public sector and businesses within the private sector (subchapter 6.1), that issue was put aside. The tailoring of CURE is discussed within subchapter 5.3.2.

6.3 Research

By comparison of five different maintenance projects and interviews of 14 persons involved in those projects, we were able to elicit a lot of information about software maintenance.

In cooperation with SchlumbergerSema, we chose five different maintenance projects to study. The projects were all in their start-up phase or already running. The characteristics of the projects were spread over several kinds of industries, size and currently used model. See subchapter 3.3 and 4.3.

6.3.1 Interviews

The people to interview were as equally as possible, three from each project, selected from within these selected projects. We wanted to interview those within the project that could provide us with the most information about their maintenance model currently used. The roles were the customer, technical system maintainer and the customer responsible. Some deviations were made but in most cases this were the interviewed roles. The reason for selecting these roles was their knowledge of the current project and the model used.

During the interviews many interesting issues were raised. Some of the issues concerned general subjects such as the overall model and some were quite detailed such as the change request handling process. We were able to classify the answers into a few categories for each question or question area. Although they have different backgrounds and come from different projects, many opinions are shared. See subchapter 4.4. The three most “popular” issues were:

- Well-defined and clear role descriptions
- Efficient error/request handling system
- Templates

When we started to select what literature to relate to, we tried to select literature from the present as well as literature written in the 80’s. We wanted to see whether the issues had changed over time or if they are the same. The literatures that we selected were both from Swedish authors as well as American. See subchapter 4.5.

After a thorough study of the literature we were however able to elicit 13 issues. Several of those issues were comparable to those discovered within the interviews. The three most “popular” issues were:

- Defined documents and routines (All authors)
- Flexible process that can be tailored to fit the situation (3/5 authors)
- Standardized documentation templates (2/5 authors)
The third place was hard to determine since a lot of other issues were mentioned by 2 authors each. However we feel that this issue was strongly pointed out.

Table 6.1 below clearly shows the relations between the two studies. It also shows how well CURE meets the requirements of both the interviews and the literature.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Case study</th>
<th>Literature</th>
<th>CURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear role descriptions</td>
<td>X</td>
<td>X</td>
<td>High / High</td>
</tr>
<tr>
<td>Routines</td>
<td>X</td>
<td>X</td>
<td>Medium / High</td>
</tr>
<tr>
<td>Request traceability</td>
<td>X</td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>Templates</td>
<td>X</td>
<td>X</td>
<td>High / High</td>
</tr>
<tr>
<td>Guidelines</td>
<td>X</td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>Information channels</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Bureaucracy</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Administration</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Sales process</td>
<td>X</td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>Package/business friendliness</td>
<td>X</td>
<td>X</td>
<td>Low / Medium</td>
</tr>
<tr>
<td>Tailorization</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Maintenance plan</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Transition</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Planning organization</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>System nature</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Individ independency</td>
<td>X</td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>Follow-ups</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Organization</td>
<td>X</td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Project boundaries</td>
<td>X</td>
<td></td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 6.1 – Summary of the comparisons (see subchapter 4.4 and 4.5)

6.4 Recommendations

The recommendations that we give within chapter 5 are the main results of this research. It is divided into three different domains. Each domain has different assumed recipients. The general recommendations are made for anyone who comes in contact with software maintenance. The other two are somewhat related to the CURE model. The recommendations about maintenance at SchlumbergerSema are in the middle between the CURE recommendations and the general maintenance recommendations. It concerns how we believe that SchlumbergerSema should conduct software maintenance. The last part is direct recommendations for the developers of the CURE model.

The general recommendations about software maintenance concerned several areas such as maintenance strategies, how to conduct maintenance, how not to conduct maintenance, pitfalls, benefits of good maintenance and the selection of a model. See subchapter 5.2.

Concerning maintenance strategies we discussed long-term or short-term selection of a maintenance model. We state that a long-term selection is preferable due to its many benefits.

As the strategy is chosen, a model has to be selected. In all cases where a system is larger than tiny, we recommend a structured model over an ad hoc solution. As the project is
started the supplier has to keep an open and sincere dialog with the customer in order to make both parts comfortable. When adopting the model and searching for tailorizations all routines, roles and administrative work has to be motivated in order to keep the level of bureaucracy at an appropriate level. If these parts are neglected we believe that the supplier will fall into “bad light” and the organization will become slow and hard to work in. The greatest pitfall that we want to point out is the risk of the model becoming a nice bunch of papers in a folder. Meaning that the model is there on paper but not implemented within the organization.

Since we during the general recommendations recommended the choice of a long-term model, we recommend the same for SchlumbergerSema. In their case we also recommend the choice of model to be CURE. This choice is based both on our experiences from the research and the fact that it is their own model. The largest benefit of using CURE is that it meets a lot of the requirements put on a maintenance model both by the interviews and the literature. Not only does it meet those requirements, using the model all over the business would increase the base for further development of the model. The pure fact of the supplier using their own model could also gain the comfort of their customers. As with all models with benefits there are also some drawbacks. These drawbacks such as the fact that it only exists in a Swedish version, is not certified by any standard or that it lacks computer based tools, are drawbacks that could be omitted due to further development of CURE. Since a model that shall be used as a long-term model has to fit a lot of different projects, it has to be quite general. Being a general model requires the possibility to tailorize the model to suit various situations. CURE is a most flexible model that is easy to tailorize. The table in subchapter 5.3 points this out.

The CURE specific recommendations are meant for the developers of CURE. It is the result of a thorough evaluation of CURE. The major benefits that we have identified are the fact that CURE meets most of the requirements from the interviews and the literature. Despite from that, CURE possesses a lot of other benefits. The model consists of several templates for various documents that need to be constructed during a maintenance project. These templates help in that way that they provide a good support. Since they exist, administration will feel less overwhelming. A benefit that is not discussed that much is that for large businesses this model is to be considered as cheap. But on the other hand this is a drawback since it will be expensive for small businesses with few maintenance projects. See subchapter 5.4.

All those characteristics that we have found as drawbacks of CURE can be solved with further development. Some of these drawbacks are certification according to a standard, translation to other languages and procedures for role backup. The two first ones give the model a mark as a quality model. It can be used by more kinds of projects and customers. The role backup procedure concerns what happens if a person leaves a maintenance project. In order to spread the model to other suppliers and/or customers, it has to be packaged. We find that the education of CURE is packaged in several levels. The model itself on the other hand is not packaged at all. Packing the model into a few packages at various size and level could ease the sales of the model. Other tools for selling the model to customers such as presenting material, success stories and statistical data are missing from the model.

### 6.5 Future Research

During our research we identified several areas that would be interesting to study in more detail. Since we did not have the time or possibility to include all these areas, we wish to discuss some of them very briefly here.
6.5.1 Standards
Since we only looked at ISO and IEEE at a quite high level, we were not able to compare them to CURE in any detail. Studying each standard in detail and identifying more standards is one interesting issue. Why are they so different and are they a complement to each other?

Another interesting issue might be to evaluate several models according to the standards to see how many that actually is following the standard, to what extent and which parts.

6.5.2 Start to use a model
More research could be done in the area of actually implementing a model. How shall the implementation start? What steps are there? Can guidelines be developed to ease the process?

This research could lessen the risk of the pitfalls described in chapter 5 e.g. allowing the model to become a nice book on the shelf and nothing more.

6.5.3 Metrics
At a point during our research we started to investigate variables for efficiency measurements. However, we found that to be outside our scope. The interest both from SchlumbergerSema and us remained, therefore we find this to be a suitable issue for future research.

The questions that we asked ourselves as we studied this issue were: how can efficiency be measured? What should be measured? What is the best way to collect data? How should it be analyzed? Can projects using different models be compared?

6.5.4 Securing knowledge
We mention some about this during our comparison between CURE and the literature. How can knowledge be secured within the project? How must knowledge be maintained in order to stay accurate? What is knowledge? Who has knowledge and who should have it? Who should pay for the continuous transfer of knowledge, the customer of the supplier?

6.5.5 Release handling
It is sometimes hard to determine what should be a fix and what should be a new release. What are the metrics and the factors that determine what should be delivered as a bug fix and what should be delivered as a new version? Has these factors or metrics changed over the years going from large Cobol-based systems to modern object oriented?
APPENDIX A

Questionnaire for the master thesis

Interviewee
1. Name
2. Current maintenance project?
3. Which role do you have in the maintenance of the current project?
4. Which assignments do you have in the maintenance of the current project?
5. Which responsibilities do you have in the maintenance of the current project?
6. How many hours/week do you work with maintenance in the current project?
7. How many maintenance projects are you involved in?
8. How long experience do you have of software maintenance? (Years, months)
9. (Collect the maintenance project documentation)

Software maintenance, generally
10. How do you experience software maintenance?
11. Why?
12. Do you have any ideas about how to make software maintenance more popular?
13. How much education/knowledge do you have of maintenance? (theory, practice)
14. How much knowledge do you have about maintenance models? (RDF, CURE, etc.)
15. How much knowledge do you have about maintenance methods?
16. How much knowledge do you have about maintenance organizations?
17. How do you see the need of software maintenance?
18. Which aspects influence the need of software maintenance?
19. How would you define the term software maintenance?
20. Do you believe that knowledge about software maintenance from the 1980’s have been lost down the road?

Software maintenance in the current project
21. What maintenance model do you use?
22. What are the benefits with your model?
23. What are the drawbacks with your model?
24. What do you think about your maintenance organization?
25. What do you think about the distribution of responsibilities within your maintenance organization?
26. What do you think about the decision-making within your maintenance organization?
27. What do you think about change request handling within your maintenance organization?
28. Do you believe that everyone in your maintenance organization (both customer and supplier) share the same view of software maintenance?
29. Why/why not?
30. What set of tools do you use within the maintenance work?
31. How would YOU like the maintenance to work?
32. How susceptible for changes do you think that your maintenance organization is?

Software maintenance, with Sema-personnel
33. How was maintenance sold to the customer?
34. How was it received by the customer?
35. Have you heard about CURE?
36. What do you think about CURE?
Software maintenance, with non-Sema-personnel
37. How did you receive the offer from Sema?
38. How did you perceive their offer regarding maintenance? Why maintenance?
39. Have you heard about CURE?
40. What do you think about CURE?
41. How do you apprehend SchlumbergerSema as a software maintainer compared to others?

A foundation for identification of maintenance metrics
Purpose: together with the interviewee have a discussion regarding suitable metrics for measure of the maintenance process efficiency over time.

A basis for software maintenance process
Metrics of the system
1. Number of rows of code
2. Size of requirements specification
3. Number of rows of comments
4. Number of subsystems
5. Number of connections to other systems
6. Number of commercial products within the system
7. Number of environments (operating systems) within the system
8. Number of users (concurrent and total)
9. Number of user-interface views
10. Requirements on availability/up-time

Workload of the maintenance organization
The utilization of resources
11. Hours/month (week or year)
12. Number of personnel involved
13. Cost/year
14. Price/year

Quality of the system
15. Errors/year
16. Changes/year
17. Adaptations/year
18. Stability of requirements, requirement changes/year
19. Stability of design, design changes/year

Efficiency of the maintenance organization
20. Number of closed cases/year
21. Number of cases in backlog
22. Number of new releases/year
23. Amount of time from case reported to closed
24. Man-hours/case
25. The percentage of bug-fixes that passes testing

Other
So far just numbers, are questions with measurable/rated answers relevant to ask? For example:
26. How do you believe that decision-making works in your maintenance organization? Efficient, Alright, Bureaucratic, Useless, Don’t know
8 References

[Bergvall] Affärsämssig systemförvaltning
Malin Bergvall & Tommy Welander
ISBN: 91-88862-01-1
1996

[Brandt] Välja och förvalta standardsystem
Peder Brandt, Rolf Carlsson & Anders G Nilsson
1998

[CURE] Internal SchlumbergerSema documents and licensed-based CURE educational material
2001

[Dawson] The essence of Computing Projects – A Student’s Guide
Christian W. Dawson
2000

[Ely] Kvalitativ forskningsmetodik i praktiken,
Margot Ely et al.
1993

[IEEE] IEEE Standards Collection
1997

1997

[Moore] Software Engineering Standards
James W. Moore
ISBN: 0-8186-8008-3
1997

[Patel] Grundbok i forskningsmetodik
Rema Patel, Ulla Tebelius
1987

[Pigoski] Practical software maintenance
Thomas M. Pigoski
ISBN: 0-471-17001-1
1997

[RDF] Systemförvaltning, collection of 6 reports
ISBN: 91-86656-11-2, report 1
1987
<table>
<thead>
<tr>
<th>Avf深切</th>
<th>Titel</th>
<th>Författare</th>
<th>ISBN</th>
<th>År</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Smith]</td>
<td>Designing maintainable software</td>
<td>Dennis D. Smith</td>
<td>0-387-98783-5</td>
<td>1999</td>
</tr>
<tr>
<td>[Svensson]</td>
<td>Kvalitativa studier i teori och praktik,</td>
<td>Per-Gunnar Svensson &amp; Bengt Starrin</td>
<td>91-44-39851-4</td>
<td>1996</td>
</tr>
<tr>
<td>[Wallén]</td>
<td>Vetenskapsteori och forskningsmetodik,</td>
<td>Göran Wallén</td>
<td>91-44-36651-5</td>
<td>1993</td>
</tr>
</tbody>
</table>