Software Maintenance and Process Improvement by CMMI

Ann-Sofie Jansson
Abstract

Software Maintenance and Process Improvement by CMMI

Ann-Sofie Jansson

Software systems are integrated to a wide extent in society. As they are implemented the need for software maintenance arises. The effort and expenses spent on software maintenance are of great magnitude. Software organizations are trying to evolve to be more efficient and improve results upon their clients and stakeholders. A promising way of achieving this is adoption of a process improvement model.

This thesis investigates the support for software maintenance in the process improvement framework CMMI (Capability Maturity Model Integration) created by the SEI (Software Engineering Institute). Methods used are a review of the CMMI documentation and a case study of an accredited CMMI organization. The review was performed by reading and searching for maintenance support in the documentation. Quotes were explicit support was found is presented with discussion concerning the possible benefits maintenance work could achieve. The case of the thesis is a large software company with CMMI accreditation. There were 23 employees answering a questionnaire about maintenance and CMMI.

The review gives the conclusion that benefits are to gain from CMMI adoption for organizations performing software maintenance but that the support is not that detailed and the benefit come on a general level not explicitly to the core tasks of software maintenance. The case study shows difficulties concerning maintenance and potential improvements. The changes due to CMMI have introduced increased work load but also improvements in configuration management, documentation and more stable processes. There have not been many previous studies done on software maintenance and CMMI. The new results from this thesis are the review of the CMMI framework that explicitly tell weather the support exists and to what extent. The case study of the thesis show new qualitative results in improvement of software maintenance and process changes due to CMMI.
Populärvetenskaplig sammanfattning


I programvaruindustrin används olika processförbättrings modeller som har till syfte att skapa stabila processer som gör att man uppnår effektiviseringar. En processförbättrings modell som är utvecklad av Software Engineering Institute (SEI) är Capability Maturity Model Integration (CMMI). Denna modell och dess föregångare är ganska väl spriden i den internationella programvaruindustrin och dess målgrupp är organisationer som utvecklar och underhåller programvarusystem.

Syftet med detta examensarbete är att undersöka om och i vilken grad CMMI stöttar underhållsarbete av programvarusystem. Alltså om organisationer som utför detta underhållsarbete kan få fördelar av att använda CMMI. Detta har gjorts genom att först gå igenom dokumentationen av CMMI och leta efter explicit stöd till underhållsarbete. Denna genomgång resulterade i en tabell över vilka av de olika delarna av CMMI som explicit vänder sig till underhållsprojekt. Resultatet är att en generell processförbättring kan förbättra betingelserna för underhållsarbete. Det explicita stödet för just underhållsarbete är dock begränsat. Av de 22 process områdena i CMMI adresserar 11 av de underhållsaspekter direkt.

Den andra delen av uppsatsen består av en fallstudie av ett programvaruföretag som utvecklar och underhåller stora programvarusystem och har infört CMMI. Anställda fick svara på en enkät om underhållsarbete och förändringar och konsekvenser av CMMI. Fallstudien visar på svårigheter med underhållsarbete och vad som skulle kunna göras för att underlätta detta. De övergripande resultaten pekar på att det finns förbättringar att vänta vid införande av CMMI. Det krävs dock resurser i form av tid och personal för att klara av det genomgripande processarbetet.
Table of Contents

1 Introduction ........................................................................................................................................... 4

2 Software Maintenance ............................................................................................................................ 6
   2.1 Fundamental Concepts ..................................................................................................................... 6
   2.2 Categories of software maintenance ............................................................................................... 8
   2.3 Assessing software system maintainability .................................................................................... 11
   2.4 Maintenance and documentation ................................................................................................. 11

3 Capability Maturity Model Integration ................................................................................................ 12
   3.1 Background .................................................................................................................................. 12
   3.2 Extension of use of the CMMI in industry ..................................................................................... 12
   3.3 CMMI ........................................................................................................................................... 13
      3.3.1 Continuous Representation and Staged Representation ......................................................... 13
      3.3.2 The model components ........................................................................................................... 13
      3.3.3 Capability and maturity levels ................................................................................................. 14
      3.3.4 The process areas ..................................................................................................................... 15

4 Software maintenance and CMMI ......................................................................................................... 17
   4.1 Method .......................................................................................................................................... 17
   4.2 Findings ........................................................................................................................................ 17

5 Review of CMMI documentation to find support for software maintenance ............................... 19
   5.1 Method .......................................................................................................................................... 19
   5.2 Results .......................................................................................................................................... 19
   5.3 Discussion/Summary ..................................................................................................................... 24

6 Case study ............................................................................................................................................ 27
   6.1 Description of the case .................................................................................................................. 27
      6.1.1 The projects ............................................................................................................................ 27
   6.2 Method .......................................................................................................................................... 28
   6.3 Data presentation and analysis .................................................................................................... 29
      6.3.1 Demography ............................................................................................................................ 30
      6.3.2 CMMI .................................................................................................................................... 30
      6.3.3 Difficulties during software maintenance work ....................................................................... 31
      6.3.4 The most difficult type of changes and why ........................................................................... 31
      6.3.5 What would help to better perform maintenance tasks? ....................................................... 31
      6.3.6 Maintenance answers in Likert scale ..................................................................................... 32
      6.3.7 General maintenance answers in Likert scale ...................................................................... 32
      6.3.8 What is successful maintenance and which are the most common errors? ......................... 32
   6.4 Results .......................................................................................................................................... 32

7 Conclusions .......................................................................................................................................... 34
   7.1 Part I the Review ........................................................................................................................... 34
Appendix A: Questionnaire

List of tables
Table 4.1 .................................................................................................................. 15
Table 5. 1 .................................................................................................................. 19
Table 5. 2 .................................................................................................................. 20
Table 5. 3 .................................................................................................................. 20
Table 5. 4 .................................................................................................................. 20
Table 5. 5 .................................................................................................................. 21
Table 5. 6 .................................................................................................................. 22
Table 5. 7 .................................................................................................................. 22
Table 5. 8 .................................................................................................................. 22
Table 5. 9 .................................................................................................................. 24
Table 5. 10 ................................................................................................................. 24
Table 5. 11 ................................................................................................................. 24
1 Introduction

Software has become a highly valuable asset in business and economic environments today. The information technology sector is maturing and software systems are entering the maintenance phase. This requires disciplined and scheduled maintenance work efforts (Singh et al. 2007) and the need for software maintenance is increasing rapidly (Bhatt et al. 2004).

Maintenance of software systems is a large post of expense in software development. Studies have shown that more than 50% of all programming activity on a software system is done after the implementation (Coleman et al., 1994; Holgeid et al. 2000; Lehman and Belady 1985; Lientz 1983; Nosek and Prashant 1990; Pfleeger 1987; Zelkowitz 1978). The estimated magnitude of software maintenance costs is somewhere between 40-80% of total software life-cycle costs according to studies conducted by Alkhatib (1992), Kemerer (1995), Schrank et al. (1995).

CMMI-Capability Maturity Model Integration is a tool for implementing best practises for activities concerning products and services in organizations (Chrissis et al. 2007 p.3-4). It was the Software Engineering Institute (SEI) that first began the work with Capability Maturity Models, and published a book about the subject in 1995. The CMMs have meant success for many organizations; they have gained increased productivity and quality, and more accurate estimates on time and resource consumption (Chrissis et al., 2007. p.8-9). The CMMs was developed to be applied to different specific areas. Three of these models, The Capability Maturity Model for Software (SW-CMM), The Systems Engineering Capability Model and The Integrated Product Development Capability Maturity Model were used as source material to develop the CMMI. The CMMI development team at SEI combined these models and built a framework that can be used in multiple disciplines and that has a flexibility to support different approaches (Chrissis et al. 2007 p.12). In this thesis the focus is on CMMI for development.

The most important benefits of software process maturity models and standards within organisations are often said to be customer satisfaction, quality improvement and rework reduction (Ferreira et al. p.1). The SEI has reported that 2000 organizations have performed CMMI appraisals and 70 000 professionals have attended the CMMI introduction course (Brief history of CMMI Software engineering institute Carnegie Mellon www.sei.cmu.edu/about/press/CMMI-ACQ/CMMIhistory.pdf 2007-11-14).

Although software maintenance is a wide and well researched area, the role of process improvement and CMMI in maintenance have been little investigated. Conducted studies (April et al.2005; Jung & Goldenson 2003 etc.), have pointed at both negative and positive results when improving software maintenance processes by using the CMMI. Thus there is a need for further exploration of if and how process improvement models like the CMMI could improve results in software maintenance.
This thesis aims to investigate software maintenance and the CMMI in two ways. Firstly, by an analytical review of the CMMI framework from the perspective of software maintenance. The review is based on the following questions:

- Does the CMMI provide explicit support for improving software maintenance processes?
- Does the CMMI framework address unique software maintenance aspects?
- Does a higher CMMI maturity or capability level in an organization mean more mature software maintenance processes?

Secondly, software maintenance and CMMI will be studied through a case study. The case is a large software corporation that deployed CMMI. Data about this organization’s experiences with software maintenance in relation to CMMI is analysed. Twenty three participants answered a questionnaire consisting of questions about software maintenance and CMMI. The case study aims at answering the questions:

- What are the main difficulties during software maintenance?
- What changes could improve and make maintenance work easier?
- What changes adopted through CMMI have influenced the maintenance work, positively or negatively?

This thesis focuses on CMMI and the maintenance processes during the maintenance phase. This in contrast to maintenance aspects during development phase which also could benefit maintenance work.

The remainder of this essay is organized as follows. Section 2 describes software maintenance, important concepts and categorizations. Section 3 presents the content and levels of the CMMI. Section 4 gives an overview of previous studies of CMMI and software maintenance. Section 5 presents the review of the CMMI documentation, with findings and discussion about those. Section 6 describes the case study, method, answers and results. Section 7 concludes the thesis, discusses results and suggests plans for future work.
2 Software Maintenance

In this chapter there will be given a brief overview of the field of software maintenance and important concepts will be accounted for.

2.1 Fundamental Concepts

**What is software?** Grubb and Armstrong use the definition below.
“The programs, documentation and operating procedures by which computers can be made useful to man.” (Grubb & Armstrong 2003 p.6)

It is important to stress that software is not just the set of programs but also the documentation and the procedures. Software includes apart from the source and object code, documentation of all the aspects of the program and procedures in order to get the program running and operating. The documentation usually exists of requirements analysis, design documents and system and user manuals (Grubb & Armstrong 2003 p.7).
To maintain software is thus to maintain not only the program code but also to maintain the documentation and procedures.

The term **maintenance** can be defined as:
“The act of keeping an entity in an existing state of repair, efficiency, or validity; to preserve from failure or decline.” (Grubb & Armstrong 2003 p 6)

Maintenance takes place in multiple places in the society and is important for keeping technical artefacts functioning. It would be a material disaster if no maintenance work was performed. And we all in some way trusts that things around us will work. For example the train to work and the cashier machine at the grocery store. Maintenance is needed for the many technical artefacts in our surroundings. This is also true for software but the difference between maintenance of any technical artefact and maintenance of software is that software maintenance not only repairs, it should evolve the system to be able to fulfil new requests as they occur.

IEEE (Institute of Electrical end Electronics Engineers) defines **software maintenance** as:
“The 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.”
(Pankaj 2004 p. 1)

The above definition is very broad and well spread. It clearly states that the maintenance phase starts after the delivery of the software system to the organization who ordered it and that maintenance is all about modifications. Maintenance work exists in the context of an existing running system, this sets limitations on which changes can be done. If the system that is to be maintained has problems in architecture, design, unconventional class-division etc then the maintenance workers has to handle that. They can not easily change these conditions.
Software maintenance can be thought of in two different settings. One aspect is maintenance during the development phase. The other is maintenance after the software is implemented. It is essential for software maintenance to be considered during the development phase, this is when the ground is laid and decisions at this time are crucial for the entire life time of a software system. However, the main maintenance work is done after the implementation. Both aspects are thus important for the overall picture of software maintenance. To connect these aspects Lehman’s third law is interesting.

“The third law is, perhaps the most interesting and the most contentious of Lehman’s laws. It suggests that large systems have a dynamic of their own that is established at an early stage in the development process. This determines the gross trends of the system maintenance process and limits the number of possible system changes” (Sommerville 2007).

Lehman realises the importance of building systems that can change and that maintenance always has to be performed in the frames of an existing system that clearly sets boundaries.

“A program that is used undergoes continual change or becomes progressively less useful” Lehman’s first law of software evolution. (Lehman 1980)

Lehman’s statement above is fundamental. The important thing to notice is that a software system is not a static phenomenon when it is implemented. This also implies that developing a system is to lay the ground for a continuous evolution of a software system, and that software maintenance is crucial to keep a software system “alive”, which is running smoothly and evolving over time.

**Software Process**

“A software process is a set of activities that leads to the production of a software product. These activities may involve the development of software from scratch in a standard programming language like Java or C. Increasingly, however, new software is developed by extending and modifying existing systems and by configuring and integrating off-the shelf software or system components.” (Sommerville 2007 p.64)

In this thesis the processes are of interest and Sommerville express it as above. Processes are activities in a certain mix and order that lead to a product or service.

**Systems development processes**

“The tasks undertaken to construct a …system, and the management of this effort, by a group of stakeholders with agendas, who engage in transactions over time within an institutional context by applying structure to their work with a set of tools and methodologies, and who judge outcomes of their efforts and act accordingly.” (Jeffery 2005 p.45)
Above Jeffery also takes in the central stakeholders and the management of the processes leading to an implemented software system. He also mentions the tools and methodologies that are used. The CMMI is a model to help set up sufficient process and keep them running and efficient over time.

**Software maintenance process** can be defined as:
“The series of actions taken to effect change during software maintenance”
(Grubb & Armstrong 2003 p. 60)

When software systems are installed and running in an environment, sooner or later new requirements emerge. To continue being useful a system needs to be able to change and/or add functionality for these new requirements. For software to be maintainable it needs to be adaptable to changes in a cost-effective way that does not increase the probability of errors (Sommerville 2007 p.49). A system with maintainability as a critical requirement should have an architecture with wide division into parts with self-contained components that easily can be modified (Sommerville 2007 p.243). The well spread software process model, the waterfall model or the lifecycle model describes the essential parts of the software development process. Operation and maintenance is the last phase in the waterfall model and the longest. Apart from implementing new requirements maintenance is early in the phase about correcting errors not discovered earlier and improving the implementation (Sommerville p.67).

### 2.2 Categories of software maintenance

Software maintenance can be described more commonly as the general process of changing a software system after delivery to customer by the development team. A common division of the maintenance activities are done into the following categories (Sommerville 2005 p.493; IEEE 1990).

**Categories of software maintenance:**

- **Maintenance to repair software faults (corrective changes).** The faults in a software system can be of different character. There are coding errors, design errors and requirements errors. The code errors demand least resources and requirements errors most. The coding errors can be corrected by a programmer with understanding of the system. The design errors may take longer time and are more expensive. The requirements errors are the most demanding because the entire system is build to match some requirements and if these are wrong the system may need extensive redesign.

- **Maintenance to adapt the software to a different operating environment (adaptive changes).** Sometime in the life time of a system the hardware, the platform operating system or some other software that is connected to the system probably changes. Then the system needs to be changed to be able to cooperate with the new environment.

- **Maintenance to add or to modify the system’s functionality (perfective changes).** This is the most demanding type of maintenance. When something in the context of a system changes, due to organisational or business
development, the requirements on the system often change drastically. The modifications needed then are of large proportions.

- **Preventive Maintenance** is not in Sommerville’s list but in many articles this list is extended by this aspect. Preventive maintenance improves future maintenance and enhancement and aims to detect problems before they occur. (Singh et. al 2007)

Surveys by Lientz and Swanson (1980) and Nosek and Palvia (1990) have suggested that about 65% of all maintenance activity is concerned with changing requirements and with following modifications and supplements to the systems functionality, which is maintenance to add or modify system’s functionality, perfective changes. Maintenance due to environmental changes, adaptation to a different operating environment takes up 18% of all maintenance activity and 17% is about correcting faults. According to Sommerville (2007 p.493) are those numbers still roughly correct. One important aspect is that it is not correcting faults that are expensive and main focus for maintainers; it is evolving and changing the systems to cope with new requirements and environments. There are of course deviations among different kinds of systems.

April et al. claims that the technical support for software maintenance is satisfactory. There are efficient tools available. It is the managerial aspects that leave a lot more to desire. The managerial aspects are the reason that the cost of maintenance is too high, the maintenance work takes too much time and it is a problem to handle change requests in an appropriate way (April et al. 2005 p.198).

Software maintenance takes place in a context that consists of a variety of elements. The **different interfaces the maintainers have to collaborate with can be grouped as follows** (April et al. 2005 p.199):

- Customers and users of software maintenance
- Upfront maintenance and helpdesk
- Computer operations department
- Developers
- Suppliers

April et al. group **maintenance processes** into three classes:

- **Primary processes** (software maintenance operational processes)
- **Support processes** (supporting the primary processes)
- **Organizational processes** (offered by other parts of the organization e.g. training, finance, human resources, purchasing etc.)

The primary processes begin with pre-delivery and transition processes. This is the structures and coordinated process by which the maintainer follows the work of the developer with the new software. This is not a momentary event but a process. When the maintainer has taken over responsibility of the implemented software the Event and Service Request Management process steps in to handle daily operation of maintaining the running system. Requests about problems, modifications and support must be assessed and either addressed, rerouted or rejected.
Software development and software maintenance are two different types of activities with different characteristics. Software maintenance is essentially an evolutionary process (Bhatt et al. p.1). The most obvious difference is that software development is conducted on a blank canvas, while software maintenance always is bound by an already existing design and implementation. Maintenance personnel must conduct an impact analysis before doing any work on a system. Because of the system being implemented and used by people, organization of the maintenance work demands a lot of consideration to not disturb the running system and its users. Software maintenance is a key discipline because it is the force by which systems remain operational and efficient. This is extremely important in the world of today where the reliance of software is widely spread (Grubb & Armstrong 2003 s.9). Software maintenance is a very large post of expense in total software costs for a lot of organizations (Alkhatib 1992: Kemerer 1995; Schrank et al.1995).

Grubb and Armstrong (2003 p.42) suggest three overall solutions to the problem of maintenance:

- Budget and effort allocation. Assign maintenance resources to develop more maintainable systems
- Replacements of systems that are out of date and are no longer efficient
- Improvement of existing systems

As stated above, software maintenance is a complex and important subject. All software systems will face problems of different magnitude and character and if not solved large values are lost. In the society of today many organizations are dependent on software systems to be reliable at all times. To be efficient and follow an organizations development also the software must develop. The maintenance process is the set of activities performed to get the changes in place. There is, of course, as many different ways of performing software maintenance as there are people setting out for the task. It is usually prolifically to use a well proven model to be guided by. This is why models like CMMI are used and can provide possibilities to support organizations strive to become more effective.

Software maintenance is the work done on a software system after being built and implemented by the development team. Software maintenance is not just the program, the code and correcting it. It is about the system being in a context, for the people working with maintenance there is a lot more to deal with than just coding. Maintenance is, for example, also to interact with stakeholders, customers, users, to decide and evaluate requests of change.

As April et al. suggests the maintenance processes are different from the development processes; there are unique aspects of maintenance that should not be neglected. Sommerville and IEEE’s definitions of software maintenance are very simplified but important in order to get a scope of what the field includes.
There are a lot of possible approaches on how to improve software maintenance processes. As Grubb and Armstrong suggest, the field of maintenance needs to be highlighted to get resources. The aspect of software maintenance should be considered during the development process and tools to improve the maintenance process should be evaluated.

2.3 Assessing software system maintainability

Assessing maintainability is a prediction; studies have shown that a combination of expert evaluation and formal methods give good results. There are many factors that concern maintainability. The technology and environment for maintenance is changing continuously therefore it is not realistic to have the same factors to represent maintainability at all times. There is no exact way of measuring maintainability; it can only be done indirectly. Structural code measures can be used to measure well defined criteria. When using experts to measure maintainability the definitions inevitable becomes vague but can detect things that only a human mind can see, like choices of concepts etc. (Anda 2007)

2.4 Maintenance and documentation

To be able to do maintenance work successfully the maintainer must understand how making changes will affect the system, how the system is build and what all the different parts are doing and how they are connected. If up-to-date documentation is lacking, this is a serious problem for the maintainers. (de Souza et al. 2005)
3 Capability Maturity Model Integration

This chapter gives a background description of the CMMI. Information about the extension of use of the CMMI in industry is presented. The content of the CMMI is also described.

3.1 Background

The CMMI was created to be a model to help organizations improve development and maintenance processes and deliverable quality. The CMMI aims at leading the way from ad-hoc processes to mature, capable processes by performing activities guided by directions in process areas and five levels of maturity or capability. The key elements of each level are called Process Areas (Miller et al. 2002 p.1473). More capable process produce, shorter development cycle times, systems without defects, predictable process performance, and adapted products and services that better meet customers’ requirements is the goal with adopting the CMMI. The CMMI aims at providing organizations with abilities to better plan and manage their development and maintenance (Miller et al. 2002 p.1474).

The SEI published in 1995 the book, *The capability Maturity Model: Guidelines for Improving the Software Process*. This was the start for CMMs (Capability Maturity Model) for software organizations. To evolve and improve the models for businesses three different CMM models were combined to the integrated one, CMMI. The capability model for software (SW-CMM), the systems engineering capability model (SECM) and the integrated product development capability maturity model (IPD-CMM) were combined to CMMI.

In *CMMI- Guidelines for Process Integration and Product Improvement*, in the SEI series Chrissis et. al describes the CMMI framework in detail. In the introduction to the book the purpose of CMMI for development is said to be “to help organizations improve their development and maintenance processes for both products and services” (Chrissis et. al 2007 preface). This point out that maintenance is important in the CMMI and that maintenance is supposed to be improved by using the CMMI. Business objectives are motivators for using the CMMI. “Process helps an organization’s workforce meet business objectives by helping them work smarter, not harder, and with improved consistency.” (Chrissis et al. 2007 s.5)

CMMI has become a well known and respected process improvement model especially for determining the organizational maturity in product or software development (Jokela et al. 2003). The best practices in the CMMI models have been extensively evaluated by users. From the beginning each new version has incorporated improvements from thousands of change requests from the public (Chrissis et al. 2007 p. 18).

3.2 Extension of use of the CMMI in industry

Since 1991 and the first version of the CMM for software (SW-CMM) 2400 organizations have been assessed on the five level CMM-scale and more than 30 000
people have been trained in the principles and techniques of CMM. In the year 2000 SW-CMM was upgraded to CMMI with the aim to bring the CMM-approach into line with international industry standard (SEI, Carnegie Mellon, Public Relations, http://www.sei.cmu.edu/about/press/stats.html 2007-11-14). Figures from 2003 shows that 87 organizations have performed CMMI appraisals from the SEI (Zubrow 2003). In March 2007 a number of nearly 2000 organizations have reported appraisals results to the SEI. More than 70 000 professionals have attended the Introductory course to CMMI offered by the SEI and partners (Brief history of CMMI Software engineering institute Carnegie Mellon www.sei.cmu.edu/about/press/CMMI-ACQ/CMMIhistory.pdf 2007-11-14).

3.3 CMMI
In this section the content of the CMMI will be described.

3.3.1 Continuous Representation and Staged Representation

Organizations have to choose from one of two paths to follow when starting using CMMI, the continuous representation or the staged representation. The two paths offer a slightly different approach to take on CMMI. If choosing the continuous representation there is a lot of flexibility and doing things in different rates, and is suitable if it is clear which processes that are problematic. With the continuous representation the organization is going through capability levels. The staged representation is a systematic, structured approach that leads from one stage making sure the ground is laid for the next. The maturity levels contain specific process areas that need to develop and gain certain maturity to be able to go to the next level. This path is more fixed and is suitable to an organization that does not know where to start the improvement work (Chrissis et al. 2007 p.23)

3.3.2 The model components

The model components of the CMMI are grouped into categories that reflect how they are supposed to be interpreted; there are required, expected and informative components (Chrissis et al. 2007 p.31). The required components in the CMMI are the specific and generic goals that represent what an organization must do to satisfy a process area. Expected components include the generic and specific practices and are a guide to what an organization need to implement to achieve the specific and generic goals. Informative components are detailed information on, for example, work products, sub practices, amplifications, generic practice titles, goal and practice notes, and references.

The term process area is a key element in the CMMI. A process area is a collection of related practices in an area that is considered crucial to improve the specific area. The CMMI for development version 1.2 includes 22 process areas, for example configuration management, measurement and analysis and organizational process definition. For every process area there is a purpose statement. This is an informative component. The goals, are of course, of major importance. The specific goals are characteristics that must be present to have satisfied a process area. The specific goals are not the same for two different process areas, whereas the generic goals can be the same for multiple process areas. The generic goals function is to describe what must be in place for the process
areas to be institutionalized. With the specific and generic goals comes a specific and generic practice. The practices are activities considered important to accomplish the goals. (Chrissis et al. 2007 p.31-36)

3.3.3 Capability and maturity levels
Organizations using the continuous representation are guided by the capability levels. All the CMMI models reflect capability levels. Each capability level has implemented in a process area its associated generic goals and practices. For each process area the organization climbs the capability levels in order to improve and institutionalize the processes. **There are six capability levels numbered from 0-5:**

0. Incomplete
When at incomplete capability level, processes are incomplete; they are not fully performed and are missing generic goals and perhaps even some specific goals.

1. Performed
A performed process is a process that satisfies the specific goals of the process area. The generic practice of this capability level helps institutionalizing the improvements to make sure they are followed at all times.

2. Managed
Managed processes means that the processes are not just being performed but also managed. That is they have an infrastructure and are controlled, monitored and guided by important objectives of the organization. The process has a basic infrastructure and is planned and executed in line with organisational policy.

3. Defined
Defined processes follow standards, process descriptions and procedures for the project and are tailored from the organizations set of standard processes. The processes at this level are carefully described and documented for every aspect they include.

4. Quantitatively Managed
At level 4 the processes are controlled using quantitative techniques. From experience objectives are established and for a process, quantitative data can be analyzed to manage the process.

5. Optimizing
Improvement in capability level 5 is about understanding inherent variations in process and optimizing them with that knowledge. (Chrissis et al. 2007 p. 46-48)

The maturity levels support the staged representation. And as with capability levels all the CMMI models reflects maturity levels.

“A maturity level consists of related specific and generic practices for a predefined set of process areas that improve the organization’s overall performance. The maturity level of an organization provides a way to predict an organization’s performance in a given discipline or set of disciplines.” (Chrissis et al. 2007 p. 52)
There are five maturity levels:

1. **Initial**
   At maturity level 1 processes are ad hoc and uncontrolled. Organizations on maturity level 1 significantly exceed budgets and can not hold timeframes in schedules. They do get products finished but it all depends on heroic achievements of employees and successes are hard to repeat. (Chrissis et al. 2007 p. 52)

2. **Managed**
   At level 2 processes and practices based on policies are established. Projects are monitored and managed according to skills of employees, documentation, stakeholders etc. At stressful times practices are retained and no chaos is to emerge. (Chrissis et al. 2007 p.52)

3. **Defined**
   At maturity level 3 processes are described in more detail compared to level 2. The processes are more sophisticated; more organized and have an organizational identity. The organization has its own set of standard processes that are profoundly grounded in the organization. (Kulpa & Johnson 2003)

4. **Quantitatively Managed**
   At maturity level 4, the organization establishes quantitative objectives for quality and process performance and uses them as criteria in managing processes. The quantitative objectives are based on information on what customers, end users, organization and implementers need. This way quality and process performance is managed and understood in statistical terms. (Chrissis et al. 2007 p. 54)

5. **Optimizing**
   “Maturity level 5 focuses on continually improving process performance through incremental and innovative process and technological improvements.” (Chrissis et al. 2007 p. 55)

3.3.4 **The process areas**

There are 22 process areas in the CMMI. A process area is build around goals connected to a specific area, for example configuration management. Each process area has specific practices that are supposed to fulfil the goals when implemented collectively. The 22 process areas are in alphabetical order:

- Causal analysis and resolution (CAR)
- Configuration management (CM)
- Decision analysis and resolution (DAR)
- Integrated project management + IPPD (OID + IPPD)
- Measurement and analysis (MA)
- Organizational Innovation and deployment (OID)
- Organizational process definition + IPPD (OPD + IPPD)
- Organizational process focus (OPF)
- Organizational process performance (OPP)
- Organizational training (OT)
- Product integration (PI)
- Project monitoring and control (PMC)
- Project planning (PP)
Process and product quality assurance (PPQA)  
Quantitative project management (QPM)  
Requirements development (RD)  
Requirements management (REQM)  

Risk management (RSKM)  
Supplier agreement management (SAM)  
Technical solution (TS)  
Validation (VAL)  
Verification (VER)

The process areas in CMMI are grouped into four categories:
  • Process Management
  • Project Management
  • Engineering
  • Support

In chapter five the review of the documentation of the process areas of the CMMI is presented. The CMMI mostly consist of descriptions of the process areas, their goals, specific and general practices.
4 Software maintenance and CMMI

In this chapter studies published about the CMMI or the SW-CMM and maintenance are presented. Very few articles about the CMMI and software maintenance have been published.

4.1 Method

To find out what research has been done in the field of software maintenance and CMMI searching in the informatics databases of Inspec, IEEE Xplore and ACM Digital Library was performed. Table 4.1 presents the results of searching different databases. There were several hits when searching the ACM Digital Library, but a lot of these were book reviews and other irrelevant articles.

In my search for the literature about CMMI and maintenance I have used “BISYS ask”, the searching engine of the Universities of Norway.

<table>
<thead>
<tr>
<th>Date</th>
<th>Database</th>
<th>Keywords</th>
<th>No. of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-10-12</td>
<td>Inspec 2007</td>
<td>CMMI in abstract and maintenance in abstract</td>
<td>5</td>
</tr>
<tr>
<td>2007-10-12</td>
<td>Inspec 2007</td>
<td>CMMI in anywhere</td>
<td>11</td>
</tr>
<tr>
<td>2007-10-12</td>
<td>Inspec 2007</td>
<td>CMMI in title and maintenance in title</td>
<td>0</td>
</tr>
<tr>
<td>2007-10-12</td>
<td>IEEE Xplore</td>
<td>CMMI and maintenance, basic search</td>
<td>3</td>
</tr>
<tr>
<td>2007-11-07</td>
<td>ACM Digital Library</td>
<td>CMMI and maintenance in abstract</td>
<td>160</td>
</tr>
<tr>
<td>2007-11-07</td>
<td>ACM Digital Library</td>
<td>Process improvement maintenance</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 4.1

4.2 Findings

The Software Engineering Institute performed a study in 1994 (Herbsleb et al. 1994) which concluded that software process improvement can gain an organization. A number of businesses observed in this study have had noticeable benefits of CMMs.

The CMM for Software (SW-CMM) is proposed to be a process improvement model both for development and maintenance. Organizations doing only maintenance work has had difficulties with implementing SW-CMM (Drew 1992). There is also criticism about the SW-CMM not addressing maintenance directly (Kuilboer & Ashrafi 2000).

Niessink and van Vliet have done research on software maintenance and software development from the perspective of providing a service. They claim that maintenance is providing a service while development is product development and for that reason there is a need for different process improvement models for the two different types of activities (Niessink & van Vliet 2000).
Jung and Goldenson (2003) performed a study for the SEI in 2003 with the result that maintenance projects in higher maturity organizations on average have lower mean and variance in schedule deviations than comparable projects in organizations at lower maturity levels. This indicates thus that maturity has a positive effect on keeping maintenance projects on schedule. The research was done by a test of predictive validity on data from 754 maintenance projects in 441 organizations at maturity levels 1 through 3.

A study performed by Krishnan and Kellner (1999) using a combination of questionnaires and existing project metrics showed that SW-CMM based process maturity was associated typically with a reduction in delivered defects. This result was after correcting for size and personnel capability.

There has also been survey studies conducted with individuals from SW-CMM assessed organizations showing that higher maturity organizations are more likely to do well in subjective measures of performance, product quality, staff productivity, customer satisfaction, and staff morale (Herbsleb et al. 1994, Herbsleb et al. 1997).

Software organizations usually do not have defined processes for their software maintenance activities (April et al., 2005 s. 201). April et al. points out that much of software maintaining activities are unique and are not part of the software development processes. The Software Engineering Body of Knowledge (SWEBOK) and authors April et al. have identified some:

**Unique software maintenance activities** (April et al. 2005)

- Transition: a controlled and coordinated sequence of activities during which a system is transferred progressively from the developer to the maintainer
- SLAs (Service Level Agreement) and specialized (domain specific) maintenance contracts negotiated by maintainers
- Modification Request and Problem Report Help-Desk: a problem handling process used by maintainers to prioritize, document, and route the requests they receive
- Modification Request acceptance/rejection: Modification Request work over a certain size/effort/complexity may be rejected by maintainers and rerouted to a developer

April et al. (2005 p. 207) did a comparison between this unique software maintenance activities and the CMMI model content. They found that the CMMI does not explicitly address these topics. It solely focuses on project management and neglect or do not address for example:

- The concept of maintenance maturity.
- Sufficient inclusion of maintenance-specific practices as process-improvement mechanisms.
- Maintenance specific issues.
- Rejuvenation-related plans.
5 Review of CMMI documentation to find support for software maintenance

This chapter presents the review of the CMMI documentation, the methods, results and discussion.

5.1 Method
The review was done by one person in the time frame of a month. First there was conducted a fast reading of the book that was used, *CMMI Guidelines for Process Integration and product Improvement* (Chrissis et al. 2007). During the reading, search for the term software maintenance or maintenance in that sense was performed, to see if CMMI explicitly address software maintenance. The largest part of the book, about the process areas was read firmly twice. The main focus of the review was on the process areas because these describe how processes should be in order to follow CMMI guidelines and also how to work in order to progress through maturity or capability levels. The review has limitations because it was conducted by only one person in a limited time frame.

5.2 Results
The following process areas explicitly address software maintenance; *Configuration management, Integrated project management, Organizational innovation and deployment, Product integration, Requirements development, Requirements development, Risk management, Supplier agreement management, Technical Solution, Validation and Verification.*

**Process area:** Configuration management (CM)

**Maturity Level:** 2

**Quote**

Introductory notes:
“*The Configuration Management process area involves the following:*  
• Maintaining the integrity of baselines”

"CM should capture enough information to identify and maintain the configuration item after those who have developed it have gone." (Chrissis et al. 2007 p.191)

Specific Practice 1.2:
“Typical work products:
• Change request database” (Chrissis et al. 2007 p.196)

“Change requests address not only new or changed requirements, but also failures and defects in the work products. Change requests are analyzed to determine the impact that the change will have on the work product, related work products, budget, and schedule.” (Chrissis et al. 2007 p.198)

Table 5.1

The description of CM states that its purpose is to establish and maintain the integrity of work products. CM also includes maintaining the integrity of baselines (table 5.1).
Specific practice 1.2 has *change request database* as a typical work product. Dealing with change requests is a typical software maintenance activity. Table 5.1 presents the Configuration management process area that, as the quote states, addresses and is a tool for software maintenance.

**Process area:** Integrated project management (IPM)

**Maturity level:** 3

**Quote**

Specific goal 1 in this process area states:

“The projects defined process must include those processes from the organization’s set of standard processes that address all processes necessary to acquire or develop and maintain the product. The product related lifecycle processes, such as the manufacturing and support processes, are developed concurrently with the product.” (Chrissis et al. 2007 p. 224, added italic to maintain)

Table 5.2

The Integrated project management process area has the task of managing processes, choosing which processes to define, and involving relevant stakeholders. The quote in table 5.2 highlights that the processes and the work with IPM has the purpose of being helpful in both development and maintenance of a product.

**Process area:** Organizational innovation and deployment (OID)

**Maturity level:** 5

**Quote**

This process area has this quality and process-performance objectives among others:

“Shorter development or production time to change functionality or add new features, or adapt to new technologies” ( Chrissis et al. 2007 p.273)

“Reduce time to adapt to new technologies and business needs” (Chrissis et al. 2007 p.273)

Table 5.3

Typically software maintenance issues are to add new functionality and adapt to new technologies and business needs. The quote in table 5.3 is taken from the objective of the process area OID. If an organization follows the process area there is a possibility to reduce time spent on software maintenance concerning adaptation to new technologies and addition of new functionality.

**Process area:** Product Integration (PI)

**Maturity level:** 3

**Quote**

Specific practice 2.2:

“Management of the interfaces includes maintenance of the consistency of the interfaces throughout the life of the product, and resolution of conflict, noncompliance, and change issues.” (Chrissis et al. 2007 p.375)

Table 5.4
The quote in table 5.4 points at a very important issue of software maintenance. To be able to make changes in the system after delivery, to adapt the system to new technology etc demands that interfaces, resolution of conflict, noncompliance and change issues must be handled in a careful way and be prioritized.

**Process area: Requirements development (RD)**

**Maturity level:** 3

**Quote**

Introductory notes:
“In the case of a project that is focused on maintenance activities, the changes to the product or product components are based on changes to the existing requirements, design, or implementation. The requirements changes, if any, might be documented in change requests from the customer or users, or they might take form of new requirements received from the requirements development process. Regardless of their source or form, maintenance activities that are driven by changes to requirements are managed accordingly.” (Chrissis et al. 2007 p.465)

Specific practice 2.1:
“Establish and maintain product and product component requirements, which are based on the customer requirements.” (Chrissis et al. 2007 p.471)

“The modification of requirements due to approved requirement changes is covered by the “maintain” function of this specific practice; whereas, the administration of requirements changes is covered by the Requirements Management process area.” (Chrissis et al. 2007 p.472)

**Table 5.5**

The process area RD in table 5.5 gives clear guidelines to software maintenance projects. The quotes in table 5.5 explicitly address some of the most essential aspects of software maintenance namely the handling of the emerging of new requirements. The objective of specific practice 2.1 is quoted in table 5.5; it says that following the specific practice will benefit the maintenance of requirements. Handling of new requirements is a fundamental part of software maintenance and the process area RD clearly is meant to help set up process for this work.

**Process area: Requirements management (REQM)**

**Maturity level:** 2

**Quote**

The exact same statement as above in RD is given in this process area. (Chrissis et. al 2007 p.488 “In the case of a project that is focused on maintenance activities…”)

Specific goal 1:
“The project maintains a current and approved set of requirements over the life of the project by doing the following:
• Managing changes to requirements
• Maintaining the relationships among the requirements, the project plans, the work products
• Identifying inconsistencies among the requirements, the project plans, and the work products
• Taking corrective action”

(Chrissis et al. 2007 p.489)

Specific Practice 1.4:
“Maintain bidirectional traceability among the requirements and work products.” (Chrissis et al. 2007 p.492)

Table 5.6
The quote from specific goal 1 in REQM in table 5.6 says that if a maintenance project follows the guidelines of the process area it will create a process that focuses on maintaining requirements. This is significant for maintenance projects; one of the main goals in the maintenance phase of a software system is to make sure that the requirements are followed up.

**Process area:** Risk management (RSKM)

**Maturity level:** 3

**Quote**

“Preparation is conducted by establishing and maintaining a strategy for identifying, analyzing and mitigating risks.” (Chrissis et al. 2007 p.500)

Table 5.7
The quote in table 5.7 is describing preventive software maintenance and declaring the importance of having a strategy to identify and analyze risks also in the maintenance phase.

**Process area:** Supplier agreement management (SAM)

**Maturity level:** 2

**Quote**

Introductory notes:
“The Supplier Agreement Management process area involves the following:…

- Establishing and maintaining agreements with suppliers” (Chrissis et al. 2007 s.519)

“To minimize risks to the project, this process area can also address the acquisition of significant products and product components not delivered to the project’s customer but used to develop and maintain the product or service.” (Chrissis et al. 2007 p.520)

Specific practice 1.3:
Establish and maintain formal agreements with the supplier. (Chrissis et al. 2007 p.524)

“The supplier’s responsibilities for ongoing maintenance and support of the acquired products.” (Chrissis et al. 2007 p.525)

Table 5.8
The quote taken from process area of SAM in table 5.8 is addressing maintenance in the way of dealing with parties involved in the software system as suppliers. The supplier can have a part to play during the maintenance phase and SAM is a guide for processes concerning this subject.
**Process area:** Technical Solution (TS)

**Maturity level:** 3

**Quote**

Introductory notes:
“For a maintenance or sustainment project, the requirements in need of maintenance actions or redesign may be driven by user needs or latent defects in the product components. New requirements may arise from changes in the operating environment. Such requirements can be uncovered during verification of the product(s) where actual performance can be compared against the specified performance and unacceptable degradation can be identified. Processes associated with the Technical Solution process area should be used to perform the maintenance or sustainment design efforts.” (Chrissis et al. 2007 p.538)

Specific practice 1.1:
Considerations for alternative solutions and selection criteria includes the following:
- Cost of development, manufacturing, procurement, maintenance, and support etc.” (Chrissis et al. 2007 p.541)

“Product or product component designs must provide the appropriate content not only for implementation, but also for other phases of the product lifecycle such as modification, reprocurement, maintenance, sustainment, and installation.” (Chrissis et al. 2007 p.544)

“Examples of attributes, in addition to expected performance, for which design criteria can be established, include the following:
- Maintainable” (Chrissis et al. 2007 p.547)

Specific practice 2.2:
“Establish and maintain a technical data package” (Chrissis et al. 2007 p.548)
“This technical data package is maintained throughout the life of the product to record essential details of the product design.” (Chrissis et al. 2007 p.549)

Specific practice 2.4:
“Evaluate whether the product components should be developed, purchased, or reused based on established criteria.” (Chrissis et al. 2007 s.552)
“Analyze implications for maintenance when considering purchased or nondevelopmental items.” (Chrissis et al. 2007 p.553)

Specific practice 3.2:
“This specific practice develops and maintains the documentation that will be used to install, operate, and maintain the product.

Typical work products

…

4. Maintenance manual

Subpractices

1. Review the requirements, design, product, and test results to ensure that issues affecting the installation, operation, and maintenance documentation are identified and resolved.
2. Use effective methods to develop the installation, operation, and maintenance documentation.
3. ...  
4. Develop preliminary versions of the installation, operation, and maintenance documentation in early phases of the project lifecycle for review by relevant stakeholders.
5. Conduct peer reviews of the installation, operation, and maintenance documentation.
Revise the installation, operation, and maintenance documentation as necessary” (Chrissis et al. 2007 p.557-558)

Table 5.9

The quote in table 5.9 first states that TS has the purpose of being a process guide to maintenance of design of the system. There are several references to maintenance aspects throughout the process area. When choosing technical solutions maintenance should be a criteria to consider according to the process area TS.

**Process area:** Validation (VAL)

**Maturity level:** 3

**Quote**
“Validation activities can be applied to all aspects of the product in any of its intended environments, such as operation, training, manufacturing, maintenance, and support services.” (Chrissis et al. 2007 p.565

Specific practice 1.1:
“The validation methods address the development, maintenance, support, and training for the product or products components as appropriate” (Chrissis et al. 2007 p.568)

“An example of evaluation of maintenance concepts in the operational environment is a demonstration that maintenance tools are operating with the actual product” (Chrissis et al. 2007 p.569)

Table 5.10

The first part of the quote section in table 5.10 suggests that the process area can be used for all phases of the software systems life time. Next quote says that maintenance projects can benefit from validation methods of the process area VAL.

**Process area:** Verification (VER)

**Maturity level:** 3

**Quote**
Specific practice 1.1:
“Work products are selected based on their contribution to meeting project objectives and requirements, and to addressing project risks. The work products to be verified may include those associated with maintenance, training, and support services.” (Chrissis et al. 2007 p.581)

Table 5.11

The quote in table 5.11 addresses maintenance, it states that work products produced during the maintenance phase should be verified.

5.3 Discussion/Summary

As the result part above has shown the CMMI explicitly addresses software maintenance in some process areas. Of the 22 process areas 11 clearly mentions software maintenance according to this review. During the review many aspects were found that easily could have been very beneficial for maintenance work but were it is not clearly stated.
The way maintenance is explicitly addressed is shallow in many cases, for example, in the Verification and Integrated Project Management. In others, like Technical solution, there are several references to maintenance and maintenance is more incorporated. The aim of the CMMI is to be integrated and take on a broad approach. Therefore I think many aspects can be used in maintenance projects as well as in development projects. The broad, overall process improvement strategy of the CMMI might be enough to improve maintenance processes to a certain extent. An explicit guide for just maintenance projects might be to weaken some of the values of the CMMI. Software maintenance is an engineering discipline and is served by solid processes and practices that are monitored and controlled by empirical tools.

_Casual analysis and resolution_ is one of the process areas not mentioned above because the explicit relation to maintenance is not present. The process area can still be useful for maintenance work. Analyzing causes and defect are one of the main tasks of maintenance activities. A major goal of this process area is to prevent defects and problems from occurring. This should be of interest for a maintenance team. Their job is essentially to keep a system running smoothly under all circumstances and to repair and control defects. The corresponding practices to this process area provide methods to detect trends and patterns to be able to prevent problems from occurring.

_Decision analysis and resolution_ has also no explicit support for maintenance, but this process area could have great potentiality to improve maintenance work. When facing a problem or task in the context of an existing system it is important to choose the solution that best fits and is reliable in the future. Impact analysis is important and DAR can be a needed tool. DAR helps to keep knowledge of the decision process and mapping it to established criteria’s. The DAR also has guidelines to involve and handle stakeholders in the decision process. This is always an issue in maintenance work because the stakeholders are the owners and the users of the system and the basic goal for maintenance is to make the system working according to their requirements. This process area includes many general practices. Apart from evaluating alternative solutions to a specific problem there is also training of people and a lot of activities to institutionalize the process.

_The Configuration management_ process area address maintenance directly in two cases. Even though it may not be that obvious from the quotes above _Configuration management_ can be very useful for maintenance work. To have up-to-date documentation on a system is essential for the maintenance work. _The configuration management_ processes help to keep close track on all work products within a project. Maintenance means continually changing of a system. It is very important to follow up with information about what has been done. This process area provides a systematic approach to dealing with large amounts of information, work products. The maintenance work should be easier with a structure like this because of the importance of keeping track on what has been and what is to come. The maintenance work is about understanding what have been, where the system is and the right way to go.
Integrated product management process area gives some support for the unique software maintenance activity of transferring the software system from developers to maintainers. The instruction is short and not detailed but points in the directions of highlighting the transition activity. The unique activities of modification requests activities of software maintenance are not accounted for in the CMMI in the needed level of detail. The requirements development and management process area gives support to the handling of emerging new requirements not for prioritizing and maybe reroute the requests they receive. The requests may not be in form of new requirements, these are two different things. Service level agreement is supported by the process area of Supplier agreement management. This is the unique activity that is given most support in the CMMI documentation. The handling of agreements with suppliers at maintenance level is clearly described in the process area.

From the review of the CMMI explicitly supporting software maintenance conducted in the frames of this thesis, the results show that 11 of 22 process areas explicitly address software maintenance. Three of the 11, Requirements development, Requirements management and Technical Solution has more extended guidelines towards software maintenance. In Requirements development the support for maintenance projects are mentioned already in the introductory notes to the process area. There are also several more detailed instructions to maintenance work. This is in contrast to, for example, Organizational innovation and deployment that just mention maintenance quite short.

Overall the result of the review is that there is a support to find, mainly in three of the 22 process areas. A maintenance project could count on support and improvement possibilities in Requirements development and management, and Technical solution. Technical solution gives guidelines in how to perform design efforts in maintenance projects and consider maintenance aspects during development phase. The other eight addressing maintenance can also give some direction and foundation for process improvement in maintenance projects.

Many of the process areas that not explicitly address software maintenance has mostly to do with organizational, administrative and management aspects and training of people. The connection to technology intense work is not so clear in these process areas. To take an example Organizational process definition is a process area entirely concerned with producing a library of work-products concerning the processes, which are descriptions and guidelines of processes. This can be summarized as administrative work concerning only the processes. All this work with process of course takes time and resources, which maybe could be spent on actually doing maintenance or development tasks. A conclusion is that the CMMI does not only concern the performance of technical work tasks. The CMMI affects many aspects of an organization and demands to integrate how to perform technical work in the business goals and all policies of the organization. As the name of the CMMI implies, it is an integration of models in the organization. If that is a positive or negative thing is for discussion.
6 Case study

There are several different ways of doing science research, in software engineering, surveys, case studies and experiments are often used. Case studies are an often used research method when answering the question “why” or “how” in a contemporary context of which the researcher has little or no control (Yin 1994 p.1). Case studies are explorative or descriptive; the research aims to investigate a piece of reality. The case study results are an example from reality. It shows how things work in a specific place, in a specific time with some specific people. The focus of the case study is how the people in the case function, often according to a special phenomenon. When conducting a case study there is no specific hypothesis that is confirmed or falsified. Often case studies are thought of a pre phase to such a study. The case study method was used in this thesis because little has been done in the specific area of CMMI and maintenance before. When researching processes it is difficult to set up an experiment. It takes a long period of time to establish processes and involves a lot of people and resources. The processes exist in a context of an organization and can not be moved to a laboratory.

Therefore it was suitable with an explorative approach to get a picture of what people in a software company have experienced with maintenance tasks and CMMI. The case study focuses on software developers experiences. Future work can be to examine several CMMI accredited companies and see if CMMI can improve specific concepts important for successful maintenance.

6.1 Description of the case

This case study was conducted at a large international software development company. The company produces systems, products and services in the defense, security and civil markets nationally and internationally. The company provides large scale systems integration, project management and software engineering, development and maintenance. At the time for this study the company was accredited CMMI level 2 and going for level 3. Because of confidentiality the thesis writer did not have authority to access specific information about the company or projects the company were currently working on which are therefore described by the supervisor Amela Karahasanovic (Karahasanovic & Vinsen 2008).

6.1.1 The projects

Project A (by Karahasanovic & Vinsen 2008)

The project developed an upgrade of a safety-critical system including software, hardware with embedded software, and support applications as part of a larger complex capability development program. The developed software is a component of a highly complex and interrelated platform system and several challenges are presented in its development; including:

• Migration of existing functionality into the new environment;
• Identification and baseline of new functionality required as a result of the capability upgrade;
• Integration with legacy and newly developed subsystems with a variety of proprietary and open standard interfaces;
• Coordination with end-users and other key stakeholders to ensure functionality is validated and with suppliers located domestically and abroad to ensure tight integration and system robustness.
• Assurance that high availability, responsiveness, redundancy, and safety attributes are built within the system for mission and safety critical operations.

The system was developed in Java and C/C++ and runs on a Sun Solaris platform. Approximately fifteen persons are assigned to this project.

**Project B (by Karahasanovic & Vinsen 2008)**

The project developed a complex software and communications system. The project has been developed primarily in a classic ‘waterfall’ development model. This was enhanced to provide a large number of working groups with end users. These working groups provided the opportunity for end users and developers to discuss everything from the functionality, user work flow, external system interfaces and operational scenarios the system might be used for.

The resulting system has been used throughout its development to assist with system integration and to allow operational testing. This in turn has provided useful feedback for subsequent interim deliveries. The system is about to start Through-Life-Support and is expected to be developed further in line with operational changes and user feedback.

**Project C (by Karahasanovic & Vinsen 2008)**

This project developed a system to assist in the monitoring, control and execution of special operations in response to crisis events. This system was developed closely with end users to jointly define and implement a solution tailored to their unique needs.

The company designed and developed a ‘proof of concept’ to demonstrate the type of technologies and functionality that could be employed. The proof of concept was further developed into a fully functional System. Using the proof of concept as a baseline, the company and the end-user community jointly performed a series of Joint Application Design workshops to elicit end user and system functional requirements. These requirements were then prototyped for confirmation of requirement intent for further refinement and baseline. The system has undergone continuous enhancement and development and evolved in response to user requirements, emerging technologies, and lessons learned from exercises and operations.

The evolution of the system has resulted in its expansion from supporting a single department to an enterprise system used across the entire organisation in support of operations and exercises locally and abroad.

**6.2 Method**

The data of this case study was collected by presenting a questionnaire on the company’s intranet. It was completely optional to participate. The analysis was done as follows. The
23 answers to the questionnaires were processed several times. The first time only looking and registering the answers about demography, the next time going back and controlling what was registered the first time. Then the most relevant questions for this thesis were registered, the questions about CMMI and what would make maintenance work easier/more efficient. The results were controlled and processes several times. The same procedure was also followed with the rest of the answers.

When going through the answers of the open questions there were sat up categories, like initial design. Every answer that said something about the initial design got a mark under that category. For every new answer that not fitted a category the answer was noted and a new category established.

The answers in Likert-scale were analyzed by calculating the median. The Likert scale is a measurement technique for measuring qualitative data. The respondents mark their answer to each item according to a simple response scale numbered in this case from 1-5 (Punch 1998).

Qualitative data like the answers of the questionnaire in the frame of this thesis are subjective expressions and must be handled accordingly. The categorization or coding described above is also a subjective evaluation of the researcher. In this case the answers were concerning engineering work and quite clear in the sense that the respondents described practical issues about their work. For example, the question about what would help to perform better maintenance had answers where 16 respondents mentioned the word documentation; then it is quite clear that they all are referring to just documentation. Of course, they can have different perception of what documentation is. The nature of the case study and qualitative data is a piece of experienced reality and not general facts. The level of detail and nuance is hard to control. As a researcher one can just try the best to handle the data as it was expressed and not read in other things apart from what is clearly stated. There can be valuable experience to gain from qualitative data, because of the value of investigating in a reality context.

This case study was designed by Karahasanovic and Vinsen (2008). Their preliminary analysis was available to the thesis writer; the results were compared and showed to be approximately 80% alike. This reduces validity problems, because, the results matches quite well. Two analyses have ended up with roughly the same results. Taking into account that the results from Karahasanovic and Vinsen were only preliminary the results can be even more equivalent. The differences were mostly concerned with the number of answers in a specific category, in the case of two questions. The answers to the open questions all had the same categories, like documentation, as the most frequently answered.

6.3 Data presentation and analysis
For complete questionnaire see Appendix A.
6.3.1 Demography
There were 23 respondents of this study, mostly men. The age of the group was between 22 and 52 with median 31 and mean value of 32.3. The education of the respondents varied from having high school/some collage till master degree. There were nine of the respondents with honours degree and nine with bachelor degree. The area of education was for 19 of the 23 computer science. As current primary job 19 of 23 said to be system developer/programmer that is 60%, and 30% team leaders, merest were project managers and business analysts. Years the respondents have been working in the software industry varied from 1 to 33 with a median of 8.6 and mean value 8. Of the 23, 19 were at the moment maintaining an application.

6.3.2 CMMI
As the company is CMMI level 2 accredited and going for level 3 this means processes are undertaking changes.

Has CMMI changes of processes meant any changes to the maintenance work positive or negative?
Five of the respondents were positive. More stable processes were said to be a good thing for maintenance. That the documentation is more detailed and gets more attention with CMMI is another positive aspect. And that the configuration management is better with CMMI. Metrics was easier obtained. One answered that the requirements engineering was improved with CMMI. There were four respondents with negative experiences, concerning more administrative work with the process that did not seem to gain the maintenance work. The extra work to follow process meant more time consumption and not more resources to deal with it. One thought there would be more difficult to change tools in the frame of CMMI process.

The organization is going for CMMI level 3 accreditation, which changes will improve the daily work in quality and/or efficiency?
Five of the respondents were positive. There were some that thought the processes and tools had become more coherent. The process activities are done electronic rather than on paper. Metrics are easier to obtain due to process changes. The changes due to CMMI level three also was said to make it easier to move staff in the organization and that good practice more easily could be spread. The documentation, testing and code review were other things said to be improved. The process became easier to understand and follow.

The organization is going for CMMI level 3 accreditation, which changes will negatively affect your work in quality and/or efficiency?
Nine of the respondents had negative things to say. The process changes meant more administrative work around the process that they could not see improves the maintenance work. Time spent on development was decreasing. Higher staff turnover and more complex code. More work with documentation and process but not more resources. Harder to change procedures and constantly changing requirements was also mentioned as negative changes.
6.3.3 Difficulties during software maintenance work

The answers showed that comprehension of the original design and implementation is one of the things that are difficult and complex during software maintenance, 13 mentioned this issue. Tight coupling of functionality make it difficult to isolate changes without affect other areas, this comes down to poor initial design. The initial design has to be solid and build for the possibility of changes. There were 13 answers claiming poor initial design is a problem for maintenance work. Lack of documentation and up to date documentation is also a problem said 11 participants. Performing impact analysis can be difficult thought five participants. Poor testing and preventing regression problems is complex, said seven answers. Not proper configuration management is a problem said 2 participants.

6.3.4 The most difficult type of changes and why

There are different types of maintenance tasks, ten of the respondents think it is most difficult to do corrective changes to a system; that is repairing code, design and requirement errors, seven think adaptation to new technology is most difficult and five thinks it is perfective changes i.e. changes of functionality that is most difficult.

One reason why corrective changes are difficult is that it is difficult to find the right place to do the changes. It can be complex to understand the code base, seven participants agreed to this. If the initial design is poor a small change might demand architectural changes that take a lot of time. If there has been done a workaround of one problem then the next are even harder to solve. Seven thought this is a problem, i.e. poor initial design. Design errors are hard to fix because it takes a lot of time and understanding of the whole code base and are connected to risks, answered three. Poor documentation makes the corrective changes hard, said five. The possibility of discovering even more errors once one has been fixed makes it difficult because of the extension of the problem. If a change has been done it is complicated to make sure that it does not introduce new errors to other parts of the code. This can be summarized as impact analysis and eight said this was difficult.

Adaptive changes are often difficult due to lack of information about what the software is being adapted to, answered five. The different details of the new technology to adapt to are difficult to grasp and get a total picture of, five participants. Also impact analysis and finding interfaces to the new things are troublesome, four participants. Problems due to unstable initial design are an issue concerning adaptive maintenance for two participants.

6.3.5 What would help to better perform maintenance tasks?

The thing most of the participants said would help the maintenance work is better and up-to-date documentation, 16 participants. More and automated testing would be a great support said 15 participants. Better initial design of systems would be beneficial said 12 participants. That is systems designed with clear division and maintenance in mind, so that changes are easily made. Solid management process with control and well founded requirement changes makes the maintenance work easier, it is helpful to have a process to follow, said nine participants.
6.3.6 Maintenance answers in Likert scale
(5= strongly agree, 1=strongly disagree)
Many of the respondents thought the complexity of the code they are maintaining is high, median four. Many also thought that the domain of the application is difficult, median four. Many of the respondents did not agree that the implemented code is easy to understand, especially if you are new to it, median two. It is difficult to understand the program logic many agreed, median four.

6.3.7 General maintenance answers in Likert scale
Successful maintenance means satisfied customers, that quality of documentation and code is preserved and that required change requests has been processes on time, median four on Likert scale. It is important for successful maintenance that developers and maintainers have good knowledge of the application to maintain, the programming language and tools that are used, median five.

Many answered that they think it is important to have a method and process to follow when handling change management, to do high quality maintenance work, median five. Configuration and version management tools are very helpful for maintenance, median 5. It helps maintenance work if all changes are registered/document, median four. To have enough time and people, resources are crucial for successful maintenance thought many, median five.

6.3.8 What is successful maintenance and which are the most common errors?
Of the respondents 12 thought preserving the systems quality was most important for successful maintenance and nine that the customers are satisfied.

The most common errors during software maintenance are to not do a proper impact analysis or to fail doing one, 14 participants. Not doing enough testing say nine is a problem. Insufficient documentation of performed changes is a problem thinks five respondents. To do a quick fix, not refactor or find the best solution is problems think six respondents. Hard to follow process is an issue for three respondents.

6.4 Results
The respondents have respectful experiences, with a mean value of eight years, and nearly all of them were working with software maintenance at the time of the survey. The majority of them also had training and degree in computer science. The case study showed that to comprehend an existing system in the context of maintenance work is difficult. It also showed that if initial design suffers with problems this has great impact on maintenance work in a negative way. Not up-to-date documentation over the system to be maintained is also a problem according to the respondents.

Positive changes due to CMMI were accounted for by a few of the respondents. There were also a few that had negative things to say. Positive things were said to be more controlled maintenance process, more documentation, more efficient configuration
management. Negative things with CMMI changes can be more process administrative work that does not improve the actual problem solving maintenance work.

Things that were mentioned as positive due to changes in process moving to CMMI level 3 were more coherent and easier process to follow. Also that the changes would lead to easier to move staff around in the organization and less paperwork and more electronic tools around the process. More sufficient documentation, testing and code review were other positive changes.

Negative changes caused by the CMMI adoption were that much more work with the administration around the processes was required. Developers did not see how it would gain the actual maintenance work. All this extra work with process took a lot of resources that was not accounted for. As the process changes it became harder to change procedures or other things inside it.

The answers concerning CMMI had only nine answers of 23 possible, at the most, and the picture was kind of dual. Some positive things were lifted but also negative associations were expressed.

The most demanding type of maintenance tasks are corrective and adaptive maintenance. In corrective maintenance this is mostly due to that it is complicated to do sufficient impact analysis. Also comprehending code base is one of the greatest problems associated with corrective maintenance.

The difficulties with adaptive maintenance are to get knowledge about the new technologies to adapt to and to control all technical details to make the adaptation run smoothly.

Aspects that would make maintenance work easier are up-to-date documentation that is easily available and searchable, as well as commented code. Another aspect is testing, that if used, would improve maintenance. To work with a system that has been built to be maintained is a great benefit.

The case study overall showed that there are many difficult things about maintenance, and that it is possible to make changes to improve and facilitate the maintenance work. The lacking documentation, testing and poor initial design, and difficulties in making impact analysis all could be helped off. There are with other words possibilities to improve maintenance work. Based on the answers from the case study on CMMI there are not many obvious conclusions, there were too few answers. The answers received pointed at some different directions, there were positive reactions and some example of that people thought it was helpful with CMMI processes and that it meant improvement for maintenance. The negative reactions can be traced to lack of proper resources to cope with increasing work pressure due to more process work.
7 Conclusions

7.1 Part I the Review

The results have shown that in half of the process areas there are explicit support for software maintenance. The support is of changing character. In especially three of the process areas, Requirements development, Requirements management and Technical Solution there are broad support and in the rest less, the others may still be of importance to maintenance improvements.

The process areas that explicitly address software maintenance found in the review were described at maturity levels 2-5. This indicates that increase in maturity level also means more mature maintenance processes. But the question is if the CMMI process areas describe maintenance processes? The CMMI can support and improve elements important to software maintenance, showed in the found process areas. This does not automatically mean that the processes consisting of the essential maintenance activities become more mature.

For example, impact analysis which is an important activity in software maintenance is not mentioned in the CMMI documentation. However, CMMI is a process improvement framework at a different level. The CMMI describes processes at a higher level of abstraction. The CMMI demands of the organization to choose for them self which areas of their work that is to be improved. This general approach gives a lot of space to decide which areas and processes to focus the improvement effort around. The CMMI is supposed to help by establishing machinery around the core tasks to make them more stable and controllable.

The objective of this thesis was to look at software maintenance because it is important in the society today and because it demands a lot of resources. This can be set in contrast to software development. To say that CMMI gives greater support for software development is not that obvious. As just said, CMMI does not get too involved in certain technological factors or unique activities. The processes and procedures are the means to the goal and have main focus. So as a conclusion higher CMMI maturity levels give more mature processes, in a general way, and there is the possibility to focus on maintenance processes and there is explicit support for this in half the process areas. The rest of the process areas may also benefit the maintenance work, in terms of a overall thought through approach to every aspect of the work of an organization and to a lot of helping functions around the specific maintenance tasks, like Decisions analysis resolution or Organizational training.

Research has been done showing that CMMI can be a successful process improvement framework for organizations (Herbsleb 1994, 1997; Jung & Goldenson 2003; Krishnan & Kellner 1999 etc.) April et al. have conducted a comparison of CMMI and what they referred to as unique software maintenance activities and concluded that CMMI does not support those aspects in a satisfying way (April et al. 2005). The results from this thesis confirm the study of April et al. because they show that there is some support for maintenance in the CMMI but it is limited. The difference in approach is that April et al.
started with their unique activities and search for support for these. This thesis was searching for maintenance support in a broader sense. That could also explain why the review in this essay found more support than April et al. Similarities in results is because CMMI is shown to be quite general and only to a certain extent oriented towards the software engineering area.

That Jung & Goldenson and others have shown that positive results in maintenance and development projects due to CMMI can maybe be derived from the general improvements in process. The overall management control, planning, documentation, configuration management etc are improved. Those results agree with the ones emerging from this thesis. General process improvement has potential to improve processes that have elements like configuration control etc that is helpful to maintenance work.

The results from this thesis are important because of the need to get understanding and knowledge about maintenance and how to improve it. This is because of the increasing need of maintenance of software in society and the large expenses that is enclosed. The review and results emerged from this thesis does thus add some knowledge to the whole picture of understanding the role of the CMMI in improving software maintenance.

7.2 Part II the Case study

The results from the case study of this thesis merely show that as many others have done, that maintenance must be laid ground for in development phase. Also that documentation is very important to keep track of a software systems lifetime and its architecture and functionality. Comprehension and maintenance are also as shown in this thesis and in research before tightly connected. If the maintenance personnel do not comprehend the system it is impossible to make the right changes to solve a problem.

The new results emerging from this thesis are the role of the CMMI in maintenance projects. Only some of the 23 participants in the case study had something to say about CMMI. At the most there were nine answers to a question baut CMMI; this was when asked about negative changes due to CMMI accreditation. There were also positive experiences expressed. To conclude the results from the case study concerning CMMI and maintenance there is to say that there are things to benefit from in the maintenance work when adopting CMMI but to a certain price.

What is to remember as an organization considering adoption of CMMI is weather one is able to invest in all the process work to gain the improvements. This is a result from the case study; the participants thought they were given more work with the same resources, which is a negative experience when time is taken from what you consider your core tasks, to maintain a software system. The process work has a lot of aspects not directly coupled to the core tasks which can seem unnecessary. The negative experiences were mostly concerned with this extra work load that they did not see to pay off.

The positive experiences were more articulated and could point at more detailed aspects of improvements like better configuration management, better documentation, more
stable processes, easier obtained metrics, easier to move staff and better spreading of
good practice.

This is connected to the previous studies (Herbsleb 1994; Jung & Goldenson 2003 etc.)
that say that organizations have to gain from CMMI adoption in different areas, like to
follow time frames, product quality etc. But also to studies showing that there is a lack of
support towards the core tasks of software maintenance (Kuilboer & Ashrafi; Drew 1992;
Niessink & van Vilen 2000).

This thesis focuses on maintenance processes after implementation of a system. The
maintenance work after implementation is however strongly influenced by how well the
system is created to be easily maintainable. The CMMI also have potential to improve
processes during development which could result in a better documented and well
planned system that is easier to maintain.

Limitations of the study of this thesis are due to limited time and resources. Given the
data being qualitative, general conclusion can not be drawn from the results. The part of
the questionnaire concerning CMMI and maintenance had only a few participants which
also clearly limits what can be concluded. Nevertheless the study can be made of use to
point out the direction for future work of more experimental nature.

7.3 Future work

Future work can be to examine organizations with pure maintenance project and with
high maturity levels to test hypothesis about if CMMI makes the initial design,
documentation, testing, comprehension more efficient.
8 References

8.1 Literature


### 8.2 Electronic information

Software Engineering Institute, Canegie Mellon, Press center. 


Appendix A

Study on Maintenance

The goal of this study is to understand and advance software maintenance.

You will be asked to complete a questionnaire about your background and software maintenance. It takes about 30 minutes to answer all the questions. Please take your time and answer these questions from your own point of view.

The questionnaire consists of 9 pages. Questions marked with * are mandatory. When you are asked to choose an option, please choose ONLY ONE option by marking it with ‘x’. When you are asked to give your opinion, please feel free to give as many details as you want. There are no correct or wrong answers.

Information collected during this survey is confidential and will not be disclosed to your colleagues, superiors or third parties without your written permission. The information is for research purposes only and will not be published in such a way that any participant can be identified. Your name will not be recorded and cannot be connected with your answers. The data will not be used at all to assess your job performance.

Thank you very much for your collaboration!

Part I: Background

1.1)* Please indicate your gender:

- Female
- Male

1.2)* How old are you?

- (years)

1.3)* Please indicate your highest level of education:

- High school/some college
- Bachelor degree (3 years)
- Honours degree (4 years)
- Master (MSc)
- Doctorate (PhD)

1.4)* Please indicate the area of your highest level of education:

- Computer Science/ Software Engineering/ Informatics/ Information Science
- Other engineering fields (Electrical, Mechanical, Civil, etc)
- Natural science (Physics, Biology, Chemistry, etc)
- Other
1.5)* Please indicate your current primary job (only one option):

<table>
<thead>
<tr>
<th>Job Title</th>
<th>Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Analyst</td>
<td></td>
</tr>
<tr>
<td>Database Administrator</td>
<td></td>
</tr>
<tr>
<td>Database Developer</td>
<td></td>
</tr>
<tr>
<td>Network Engineer</td>
<td></td>
</tr>
<tr>
<td>Requirement/System Analyst</td>
<td></td>
</tr>
<tr>
<td>System Developer/Programmer</td>
<td></td>
</tr>
<tr>
<td>Maintainer</td>
<td></td>
</tr>
<tr>
<td>QA/ Testing responsible</td>
<td></td>
</tr>
<tr>
<td>Change Control Officer</td>
<td></td>
</tr>
<tr>
<td>Project Manager/Leader</td>
<td></td>
</tr>
<tr>
<td>Team Leader</td>
<td></td>
</tr>
<tr>
<td>Senior Manager</td>
<td></td>
</tr>
<tr>
<td>Other, specify</td>
<td></td>
</tr>
</tbody>
</table>

1.6)* Which project/projects are you currently working on?

1.7)* How many years have you been working in software industry?

__ years

In the questions given below you will be asked to indicate your expertise. The following four levels are used to describe the expertise:

- **Basic knowledge:** an individual has a basic knowledge of the subject matter and shows an awareness of how this competency relates to their job.
- **Working knowledge:** an individual has a working knowledge of the subject matter such that they are able to effectively apply that in their job.
- **Proficient:** through the experience of applying their knowledge on the job, or other related activities, an individual has developed a thorough understanding of the subject matter AND is highly proficient in being able to apply that knowledge in their work environment.
- **Subject matter expert:** through years of experience, or advanced study, an individual has developed a comprehensive understanding of the subject matter and its interactions with other disciplines/competencies. The individual has advanced their set of skills to be able to apply their expertise to a multitude of projects and situations. The individual utilizes their in-depth knowledge to communicate and collaborate with peers within their normal work environment and outside to other professional business or technical communities.

1.8)* Please indicate your general expertise as a software professional:

<table>
<thead>
<tr>
<th>Level</th>
<th>Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic knowledge</td>
<td></td>
</tr>
<tr>
<td>Working knowledge</td>
<td></td>
</tr>
<tr>
<td>Proficient</td>
<td></td>
</tr>
<tr>
<td>Subject matter expert</td>
<td></td>
</tr>
</tbody>
</table>

Please indicate your expertise in the following areas:

1.9)* Design:
1.10)* Development:

Basic knowledge
Working knowledge
Proficient
Subject matter expert

1.11)* Verification and testing:

Basic knowledge
Working knowledge
Proficient
Subject matter expert

1.12)* Maintenance:

Basic knowledge
Working knowledge
Proficient
Subject matter expert

1.13)* Application system (functionality, code, documentation) you are currently working on:

Basic knowledge
Working knowledge
Proficient
Subject matter expert

1.14)* Tools and the integrated programming environments you are using in your current project/projects:

Basic knowledge
Working knowledge
Proficient
Subject matter expert

1.15)* Programming language/languages you are using in your current project/projects:

Basic knowledge
Working knowledge
Proficient
Subject matter expert

1.16)* Application domain of your current project/projects:

Basic knowledge
Working knowledge
Proficient
Subject matter expert
1.17)* Do you maintain an application now?

Yes ☐
No ☐

If you answered YES to Question 1.17, please answer Questions 1.18 to 1.20

1.18) How long have you been working in the maintenance domain (total number of years, not only on this project; you can write decimals if you want)?

___ (years)

1.19) How long have you been working on maintenance with the current application?

___ (years)

1.20) Have you been involved in the development of this application?

Yes ☐
No ☐
Part II – Maintenance

2.1)* What do you think is complex, i.e., difficult to understand and perform during software maintenance? Please describe briefly at least the three most difficult issues.

2.2)* Please indicate what kind of changes you find most difficult?

<table>
<thead>
<tr>
<th>Change Type</th>
<th>Not Applicable</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrective changes - repairing coding errors,</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>design errors and requirement errors</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adaptive changes - changes due to changes of</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>the system’s environment (hardware,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>operating system or other support software</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perfective changes - adding or modifying system</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>’s functionality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preventive changes - changes conducted to</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>prevent problems before they occur</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.3)* What do you think is difficult about corrective changes and why?

2.4)* What do you think is difficult about adaptive changes and why?

2.5)* What do you think is difficult about perfective changes and why?

2.6)* What do you think is difficult about preventive changes and why?

2.7)* What would help you to better perform maintenance tasks? Please suggest at least three areas for improvement.

In the questions given below, please consider the software maintenance of the system/systems you are currently working on. Please read each statement and indicate how strongly you agree or disagree with the statement by choosing a number on the scale (5 = strongly agree, 1 = strongly disagree)

<table>
<thead>
<tr>
<th>(5 = strongly agree, 1 = strongly disagree)</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Not applicable</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.8)* Other team members and experts are</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>easily available when needed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.9)* The team I am working with is</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>generally well motivated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.10)* I often receive feedback on the</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>quality of conducted changes from other</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>team members.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(5 = strongly agree, 1 = strongly disagree)</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Not applicable</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.11)* The provided on-the-job training</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>helps me in conducting maintenance tasks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>at the current project.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(5 = strongly agree, 1 = strongly disagree)</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Not applicable</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.12)* There is enough resources for</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>conducting maintenance tasks (time and people)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>at this project.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.13)* The change management process</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>is strictly followed at this project.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.14)* All changes (change requests and</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>conducted changes) are registered/document-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ed at this project.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.15)* Change management tools and</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>configuration/version management tools used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>at this project provide very good support</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for conducting maintenance tasks.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.16)* The complexity of the code is</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>very high.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.17)* The application domain is difficult</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>(complex).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.18)* The quality of the design/implemented</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>code was very good during the early phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>of the project.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In the questions given below, please consider the **software maintenance of the system/systems you are currently working on**. Please read each statement and indicate how strongly you agree or disagree with the statement by choosing a number on the scale (5 = strongly agree, 1 = strongly disagree)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.19)*</td>
<td>The quality of the design/implemented code at present is very good.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.20)*</td>
<td>The implemented code (at present) is easy to understand even if you are a new team member or have not been working with the application for a while.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.21)*</td>
<td>When undertaking changes on the current system it is difficult to understand the program logic.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.22)*</td>
<td>When undertaking changes on the current system it is difficult to understand and reuse classes developed earlier.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.23)*</td>
<td>When undertaking changes on the current system it is difficult to understand overall program structure.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.24)*</td>
<td>When undertaking changes on the current system it is difficult to find all classes/files affected by a change.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.25)*</td>
<td>When undertaking changes on the current system it is difficult to find all the places within a class/file affected by a change.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.26)*</td>
<td>Links between a program and documentation were reliable during the early phases of the project.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.27)*</td>
<td>Links between a program and documentation are reliable at present.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.28)*</td>
<td>The documentation is not updated after emergency change requests.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2.29)*</td>
<td>The provided documentation is very useful for the maintenance.</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

2.30)* Please indicate your three most common errors during software maintenance.
Part III – General maintenance

In the questions given below, please consider **software maintenance in general** (not any particular project). Please read each statement and indicate how strongly you agree or disagree with the statement by choosing a number on the scale (5 = strongly agree, 1 = strongly disagree).

<table>
<thead>
<tr>
<th>Statement</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1)* Successful maintenance means that customer is satisfied.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2)* Successful maintenance means that the required change request has been processed on time.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3)* Successful maintenance means that the quality of the code and documentation do not deteriorate.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4)* Successful maintenance means that the defined process is strictly followed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.5)* Please order the following criteria according to their importance for successful maintenance by assigning numbers from 1 to 4 (1 = the most important).

<table>
<thead>
<tr>
<th>Criteria</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer satisfaction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On-time processing of change requests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preserving the system quality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Following the process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the questions given below, please consider **software maintenance in general** (not any particular project). Please read each statement and indicate how strongly you agree or disagree with the statement by choosing a number on the scale (5 = strongly agree, 1 = strongly disagree).

<table>
<thead>
<tr>
<th>Statement</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6)* It is important for successful maintenance that the maintenance team is involved in the development.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.7)* It is important for successful maintenance that developers/maintainers have good knowledge of the application.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.8)* It is important for successful maintenance that developers/maintainers have good knowledge of development/maintenance techniques and tools.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.9)* It is important for successful maintenance that developers/maintainers have good knowledge of programming languages.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.10)* It is important for successful maintenance that the maintenance team is stable.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.11)* It is important for successful maintenance that other team members and experts are easily available when needed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.12)* It is important for successful maintenance that developers/maintainers are well motivated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statement</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.13)* It is important for successful maintenance that developers/maintainers receive feedback on the quality of their work.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.14)* It is important for successful maintenance that good training (general, project specific) of team members is provided.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.15)* It is important for successful maintenance that enough resources (time and people) is available.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.16)* It is important for successful maintenance to strictly follow a well defined process/methodology for change management.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.17)* It is important for successful maintenance that all changes (requests and conducted changes) are registered/documentated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.18)* It is important for successful maintenance to use good configuration and version management tools.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.19)*</td>
<td>It is important for successful maintenance that a separate budget is dedicated to maintenance.</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\(5 = \text{strongly agree, } 1 = \text{strongly disagree}\)
Part IV – Capability Maturity Model Integration (CMMI)

Your organisation is a Capability Maturity Model Integration level 2 accredited company and it goes for level 3 accreditation. You may have noticed that some processes have been changed due to the CMMI accreditation.

4.1)* Do you think that any of these changes make maintenance easier/more difficult? Please describe briefly these changes and explain how they affect maintenance.

4.2)* What other changes have happened as the organisation is moving to Level 3? Please describe briefly three changes that you think improve your work in terms of quality and/or efficiency.

4.3) Please describe briefly three changes that you think negatively affect your work in terms of quality and/or efficiency.

Please check if you have answered all questions and save the file!
If you have any other comments and suggestions, please write them here.