Reusing software assets in agile development organizations - a management tool

A case at a medium sized software development organization

Unrestricted version: Master thesis W.J.T. Spoelstra

[This page was intentionally left bank]

Reusing software assets in agile development organizations - a management tool

A case study at a medium sized software development organization

Hengelo, August 2010

Author:

Wouter Spoelstra

Program: Business and Information Technology,
School of Management and Governance

Graduation committee:

Marten van Sinderen

Department: Information Systems (IS)

Maria Iacob

Department Information Systems and Change Management (ISCM)

Jasper Laagland

Department External

Johan te Winkel

Department External

UNIVERSITY OF TWENTE.

University of Twente

Drienerlolaan 5

7522 NB Enschede

Preface

This paper presents the final research project of my master Business Information Technology. After six and a half pleasant years at the University of Twente this research completes my study and provides me the opportunity to discover the world in a new way, to choose a new direction on the road that leads me to the day of tomorrow.

While standing on this road and looking backwards I realize that I would not be here without the great support provided by a lot people, which I hereby want to thank.

First of all, I would like to thank my supervisors from the university, Maria Iacob and Marten van Sinderen, for their patience and ongoing support in all the research directions I went through. Their discussions and sharp feedback allowed me to structure and focus my ideas in such a way as this research report now presents to you.

The research project was completed during an internship at a medium sized software development organization. I would like to thank the management for offering me this opportunity. Moreover, I would like to express my gratitude to Jasper Laagland and Johan te Winkel as my company supervisors for their insights, fruitful discussions and motivation during the whole process. Furthermore, I would like to thank all my fellow colleagues and other interns for the interesting discussions, pleasant lunch breaks and good working atmosphere. Among these people I would like to thank the experts in particular, who helped me with the validation of the research outcomes.

Much appreciation also goes to my parents, my family and my friends for their everlasting support in finishing this research project successfully and supporting me during all the miles that I have travelled on my road already. Special thanks go to Erik van Gorp for helping me review this thesis thoroughly.

When looking backwards at the distance that I have travelled the choices made seem to be so obvious. But when looking forward the choices are not so obvious at all. The road bends in many directions and its crossings represent opportunities which you can take or cannot take.

In the short term future I take the opportunity to submit a paper version of this research to the ACM symposium on Applied Computing. If the paper is accepted a possible trip to Taiwan may follow. If the paper is not accepted this may be my last scientific contribution, because I already started working the 1st of August. From there on my journey continues and many new opportunities and choices are likely to follow.

Now we have arrived at the point that I have nothing more to say than that I hope you will enjoy reading and will be able to maximally benefit from the contents of this thesis. If you have any questions or comments then please do not hesitate to contact me, I will be happy to help you if I can.

Best regards,

Wouter Spoelstra

Hengelo, August 2010

Management summary

Research background

Organization have been building on existing knowledge since the beginning of ages. The reuse of existing knowledge has proven to be a valuable basis for increasing employee productivity and product quality. In the software industry the reuse of knowledge manifests itself in software assets such as code components, functional designs and test cases. The reuse of this kind of knowledge is also referred to as software reuse. Software reuse is well described as a two-fold process. On one side software reuse is about creating reusable assets and on the other side it is about using these reusable assets in building new solutions.

The reuse of existing software assets offers intuitively great benefits, but unfortunately it has never reached its full potential. Within the normal project scope the creation of reusable assets is at its best a secondary concern, as the focus of a project is on creating a project specific solution as fast as possible and with minimal resources. In order to leverage knowledge outside the project scope additional resources are required. These resources are not always available and are more difficult to justify.

One of the organizations reusing existing knowledge in order to create new applications is the case organization. The medium sized software development organization is interestingly enough also working with agile development methodologies, imposing possible additional constraints on the reuse of software assets. Agile development methodologies are characterized by smaller project teams, informal processes and multiple development iterations. The use of these informal processes and the emphasis on communication and collaboration over extensive documentation can impose additional constraints on the leverage of knowledge outside the project scope. The reuse of software assets in combination with agile development methodologies is fairly unknown in literature.

Problem

The problem of software reuse in agile development environments is a complex one. When analyzing the problem in more depth it appeared that there is no holistic approach to address software reuse issues. Without such an approach an organization is not able to address specific software reuse issues and to identify potential improvement areas for achieving even greater benefits. Furthermore organizations are not always aware of the possibilities of reaching higher reuse levels and certainly do not know how to do this.

In order to address this problem a literature survey was performed with the purpose to extract requirements on the management tool for addressing those reuse issues. The requirements are extracted from the analysis of the concepts ‘software reuse’, ‘agile development methodologies’ and ‘software reuse frameworks / models’.

Solution

The proposed solution is a software reuse management tool. The management tool provides the structure and the mechanisms for addressing software reuse issues. Although the tool was intended to be specific for typical agile development environments, the decision was made not to do this, as agile development methodologies can be successfully applied in more complex environments.

The management tool is based on two main components, a reuse maturity model and an assessment method. The reuse maturity model consists of five maturity levels and fifteen reuse factors. The maturity levels are incremental plateaus for addressing software reuse. The reuse factors are factors describing relevant aspects of software reuse. The reuse factors are allocated incrementally to the maturity levels. In order to provide the user of the management tool structure, the reuse factors are allocated

The management tool is based on two main components, a reuse maturity model and an assessment method. The reuse maturity model consists of five maturity levels and fifteen reuse factors. The maturity levels are incremental plateaus for addressing software reuse. The reuse factors are factors described in literature as factors influencing software reuse, quite similar to success factors. In order provide the users mechanisms to structure the reuse factors they are organized in categories according to the BTOPP (business, technology, organization, process and people) model. Within the results of the literature review two reuse factors are emphasized because of their importance for agile development methodologies. Namely the reuse factors 'communication channels and support' and 'communication tool support'. These two reuse factors are, however, expected to be similar important for other development environments. Each combination, of a reuse factor and a maturity level, is represented by a set of reuse practices. These reuse practices are the result of operationalizing a reuse factor and can be measured by the assessment method component. The assessment method component adds a score and a relevance to each reuse factor. The score represents the state of a reuse factor in a maturity level and the relevance variable serves as an indicator for the importance of a reuse factor in a specific context. By using the relevance variable an organization can make the management tool specific for their individual context, which can possible be one where agile development methodologies are utilized. The assessment method component can be used to indicate strong and weak points of an organization.

Recommendations

The management tool has been successfully applied at the case company and proved to be valuable for structuring ideas related to software reuse. Based on the application of the management tool it also became clear that software reuse can occur without having highly standardized reuse processes and physically separated reuse teams as mentioned in traditional literature. Furthermore the relevance variable appeared to be one of the most important aspects of the management tool, making it possible to tailor the tool to the needs of the organization. Organizations applying the management tool are recommended to use a neutral individual for the assessment process and to record the results of the assessment process. Based on this research it appeared that the recorded results can provide a valuable background for explaining the assessment results. The neutral individual can be used to guide the assessment process. Furthermore he or she can be used to gather and process the results. The results can afterwards be presented to the organization in subject.

Table of Contents

1. Introduction

This chapter provides the reader with an insight into the research area and the background of this thesis. First of all the research background will be illustrated by means of an example. Secondly the research setting and the problem statement are elaborated on. Thirdly the research objectives and the research approach are discussed. Lastly the thesis structure is described.

1.1 Background

The reuse of knowledge is considered a major factor for increasing productivity and quality. Although organizations have always used different knowledge practices to produce goods and services, their way of sharing knowledge is often informal and not systematic [1]. A large quantity of literature is dedicated to this subject as the reuse of knowledge is interesting to both practitioners and researchers.

In the software industry the reuse of knowledge manifests itself in the reuse of products and code components. Unfortunately, reusing such knowledge assets is not always without risks. The accident with the Ariane 5 rocket, on 4 June 1996, is arguably one of the most expensive software failures in history [2]. The reuse of a wrong piece of software demonstrated that that system safety and quality does not necessary increase with component reuse, instead it actually took millions of dollars due to a fatal failure [3]. Nuseibeh argues that large scale software development is an engineering principle that requires '*precise specifications and trained engineers*' for the development of '*safer*' systems [3].

Precise specification and trained engineers indeed seem to be crucial for the development of safety critical systems, other more agile practice can be applied for the development of non-safety critical systems. Related to the Ariane 5 rocket example, Nuseibeh recognizes the influence of both fine-grained processes, such as '*programming and design techniques'*, and coarse-grained processes, such as '*project management*', in software development and reuse [3]. These insights perfectly illustrate the large variety of aspects visible throughout this research.

In literature the reuse of existing software assets is referred to as software reuse. Software reuse can be considered as a specific form of knowledge management, where the knowledge assets to be managed are specified down to software assets. Software reuse can be expected in any organization developing software products. The focus of knowledge management is on the support of creating, storing, retrieving, transferring and applying knowledge in organizations [4-5]. Software reuse should similarly support such processes, depending on the nature and the scope of the individual organization.

Throughout this research, software reuse is approached from a business perspective. The technology, such as modular or object oriented languages needed for software reuse, is therefore taken for granted. The focus is instead on more managerial issues, such as the coarse-grained processes mentioned by Nuseibeh.

1.2 Research setting

This section is not available in the restricted version.

1.3 Problem statement

The introduction started with the statement that companies have always used different knowledge practices to produce goods and services. This statement is also true for software development organizations, who have employed software reuse since the beginning of programming languages, to achieve increased employee productivity and higher product quality [6].

Although, intuitively large gains can be obtained with software reuse and a lot of research was dedicated to the subject, it has never reached its full potential. Within the normal project scope the creation of reusable assets is at its best a secondary process, as the focus of such a project is on creating project specific solutions, as fast as possible and with minimal resources [7]. In order to leverage knowledge outside the project scope additional resources are required and several issues have to be addressed [8]. The resources to create and maintain reusable assets are not always available and organization lack concrete insights in the potential of software reuse [9].

The use of agile development methodologies in combination with software reuse raises new questions [10]. Software development, in general, is a knowledge-intensive activity and its success heavily depends on the developers' knowledge and experience [11]. The use of agile methodologies emphasizes the dependency on these individuals even more, as the focus of agile development methodologies is on people and communication, and less on documentation [12]. Tacit knowledge is used and shared, rather than explicit knowledge. Tacit knowledge is knowledge stored in the minds of individuals, where explicit knowledge is stored in artefacts [13]. The extensive use of tacit knowledge seems to impose constraints on the reusability of software assets. Nevertheless software reuse also occurs in these organizations, but is just understood in a limited manner.

The case company is one of the companies successfully reusing software assets utilizing agile development methodologies. Although there are some known issues, their primary interest is optimizing the software reuse program, to achieve higher reuse levels in the future. Higher reuse levels are expected due to maturing solutions and domains that are continuously redfined.

To create a relevant background of known issues a prelimairy research was carried. These known issues with software reuse include that components are being developed over multiple locations, changes in components are not always communicated and that resources are not available to assign full-time roles dedicated to the management of reusable assets. Some of these issues are obviously related to software reuse in general, while others seem to be emphasized by the use of agile methodologies and the related team structures. The team structure is also inherent to the organizational structure, which promotes the flexibility of a smaller organization while still being able to grow large. An overview of the interviewees used for the primary research is presented in Appendix A. Appendix B presents the used interview protocol. The results are not included in the Appendix, but are elaborated on throughout this research report. The main reason for this is that the results, at the point in time they were obtained, lacked a concrete framework for analysis.

The problem to be solved by this thesis is that there is no good framework or tool, against which software reuse issues can be mapped, discussed and evaluated. Because there is no such a framework or tool employees find it difficult to discuss the problems and identify potential solution areas. Also it is unknown what higher levels of software reuse actually are. As such, the problem statement is formulated as following:

P1: Companies have difficulties in addressing software reuse management issues.

As already became clear in this section, the problem statement is multi-facetted. Not only is it difficult to address the issues for managing software reuse, also the demands on software reuse are changing when reaching different reuse levels. Furthermore the management issues are not limited to a single project, but rather exceed project boundaries. Lastly the management of reusable assets is possibly constrained by the use of agile development methodologies, which has to be taken into account. All of these facets are and should be addressed within the problem statement.

1.4 Research objective

Ellis and Levy note that the research objective is to provide an answer to the research problem [14]. This research should provide insights into how companies can address software reuse issues, evolving over time through different levels of reuse. Based on the former, the following research objective is formulated:

O1: To develop a management tool for addressing software reuse issues.

The management tool for software reuse, or simpler, the reuse management tool, needs to be applicable in agile development environments. Based on the previous section some other requirements also became visible. The literature review describe in Chapter 2 is used to gather and formulate all the requirements.

1.5 Research questions

The research objective is operationalized by research questions [14]. Based on the research objective formulated in the previous section, the following main research question is formulated:

R1: How to address software reuse issues through the use of a management tool?

To answer the main research question several sub-questions are defined, which follow the research method elaborated on in section 1.6. The sub-questions formulated are:

Problem analysis

R.1.1: What is software reuse and what are software reuse issues?

R.1.2: What are the incremental stages for addressing software reuse issues?

R1.3: What are the characteristics of agile development methodologies and how does it influence software reuse?

Based on the previous questions:

R.1.4: What are requirements for the software reuse management tool?

Solution alternatives

R.1.5: Which frameworks and models are available to describe software reuse and how suitable are they for a software reuse management tool?

R.1.6: How to define or adapt a suitable software reuse management tool?

Solution validation

R.1.7: Does the software reuse management tool meet the requirements as formulated in R4?

R.1.8: Is the software reuse management tool applicable in an organization working with agile development methodologies (such as at the case company)?

Solution evaluation

R.1.9: What are the recommendations to the case company based on the results of the management tool?

R.1.10: To what extent can the software reuse management tool be applied to other organizations?

R.1.11: Which improvements are possible for future research?

1.6 Research method

The research method is based on the first four phases of the regulative cycle [15]. The regulative cycle is the '*general structure of a rational problem solving process*' [15]. The first phase is the investigation of the current problem. This phase consist consists of identifying relevant problems and relating it to the current state of literature. The second phase consists of formulating solution alternatives, based upon relevant findings in literature. The third phase consists of validating the results within the given research setting. The final phase is the solution evaluation. Here the results are generalized and discussed, including implications for future research. An overview of the research design is presented in Figure 1.

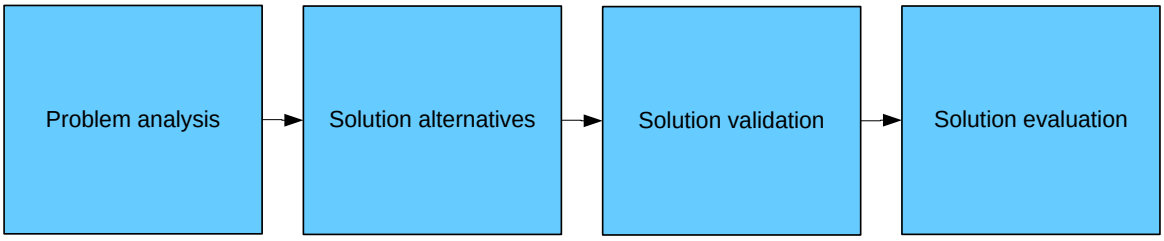


FIGURE 1: RESEARCH METHOD

The problem analysis is based on a literature review, where the results of the initial interviews are used as a relevant background. The purpose of the literature review is to identify requirements for a software reuse management tool and to evaluate available tools, frameworks and model. The evaluation of the existing tools, framework and models is part of the solution alternatives. Based on that, a new or adapted management tool is proposed. The management tool is validated through the use of an expert panel in combination with a case study. The expert panel is asked to apply the management tool to their organization, after this step the results are discussed and evaluated. During the application process and active researcher role is maintained to acquire insights in the reasoning of the experts. The results are used to improve the management tool and to provide recommendations to the case company

1.7 Report structure

The report structure follows the research methodology defined in section 1.6. This chapter provided an introduction to the subject and explained the research set up. The second chapter describes the literature survey and its results by analysing software reuse, agile development methodologies and existing reuse maturity frameworks and model. The second chapter not only elaborates on the problem analysis, but also start with the solution alternatives, as requirements are defined and existing reuse models and frameworks are evaluated. The second chapter provides an answer to research questions R1-R5. In the third chapter the reuse management tool is elaborated on, this chapter provides an answer to research question R6 and is the end of the second phase of the research method. In chapter 3 the requirements are also addressed, but not yet evaluated as a whole. The fourth chapter presents the validation approach, which is the beginning of the third phase of research method. In the fifth chapter the results of the validation are presented providing an answer to research questions R7 and R8. This fifth chapter also finishes the third step of the research method. In chapter 6 the recommendations are presented for both the management tool and the case company. Chapter 6 provides an answer to research questions R9 and R10. In chapter 7 the results are summarized and the solution is further evaluated. This last chapter also provides an answer to research question R11. An overview of the report structure is presented in Figure 2.

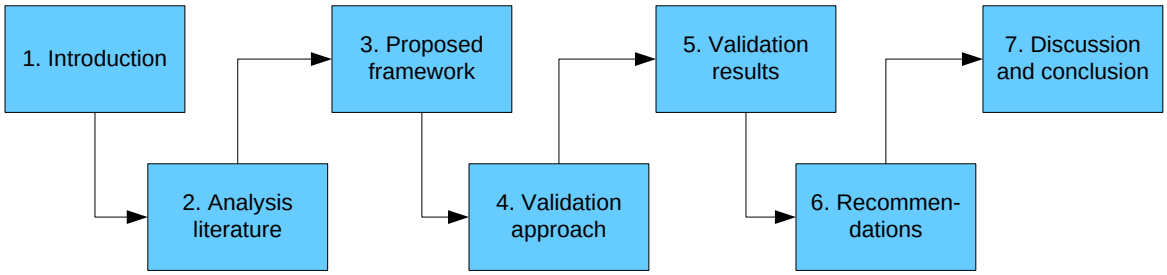


FIGURE 2: REPORT STRUCTURE

2. Literature survey

This chapter discusses the literature survey. The goal of the literature survey is to derive relevant requirements for a software reuse management tool. For this the concepts of software reuse, agile development are analyzed and discussed. After that, existing reuse frameworks and models are evaluated based on the defined requirements. Based upon the evaluation a conclusion is drawn whether to use existing reuse framework and models or to create a new one. The chapter is rounded off by a summary of the requirements discovered in the survey.

2.1 Software reuse

The introduction of this thesis started with the statement that software reuse does not differ much from knowledge management. Interestingly, some researchers use both terms to describe the same concept, e.g. Desouza et al. [16]. When investigating both concepts more closely, it appears that the assets to be managed with software reuse are specified down from general knowledge assets to software related assets. Furthermore, as mentioned by the framework of Holsapple and Joshi, *using* managed assets is the same are *reusing* managed assets [17]. This indicates that the management of assets used is the same as the management of assets to be reused. This section elaborates on software reuse characteristics and reusable assets. At the end of this section requirements are derived for the reuse management tool. Although there are many similarities between knowledge management and software reuse, the concept of knowledge management is not further elaborated on in this research report. The interested reader can use the literature survey of Alavi et al. [4] as a starting point for knowledge management research.

According to Krueger the essence of software reuse is '*the process of creating software systems from existing software rather than building software systems from scratch*' [18]. Mili et al. describe software reuse as a two-fold concept. First of all it is about '*building software that is reusable by design*' and secondly it is about '*building with reusable software*' [19]. This two-fold definition explicitly indicates that software reuse it not pure a technical issue, but also an organizational one.

Systematic software reuse has been regarded as the only appropriate solution for the notion of the software crisis as mentioned in 1968 [18-20] . Systematic software reuse is defined as reuse which is repeatable and excludes ad-hoc reuse events [7]. The software crisis was used to describe the '*ever increasing burden and frustration that software development and maintenance have placed on otherwise happy and productive organizations*' [20]. The alternative for systematic software reuse is automating the development process in such a way that '*a computer system is capable of producing executable code based on informal, incomplete and incoherent user's requirements'* [19]. At the time Mili et al. wrote this statement such an alternative was considered futuristic. Currently, much research is focussed on the development of model driven engineering and model driven architectures. For further reading see the paper of Atikson et al. [21] and the paper of Favre et al. [22]. Despite the fact that software reuse has never reached its full potential it still offers great potential for increasing software productivity and software quality [19-20]. Lanergan and Grasso estimated that 60 percent of all business application designs and code are redundant and thus can be standardized for reuse [23]. Reuse levels reported in literature range from 20% to 85%, but there are some research reports where no significant reuse levels are achieved, making such reuse figures rather meaningless. Reaching high levels of reuse seems to be associated with narrow domain focus and a suitable development strategy [24].

Software reuse literature, in its early years, was focused on repositories and technical issues such as search and retrieval of reusable assets. Another name for the repository storing reusable assets is the software library. Burton et al. define a software library as a architecture with four subsystems: '*library management*', '*user query*', '*software component retrieval and evaluation score*' and '*software computer aided design*' [25]. The second wave of literature regarding this topic recognized that technology can help software reuse, but that the focus should be on organizational factors for adapting a software reuse program [18]. Organizational factors include top management support, integration of reuse processes, best practices and the addressing human factors [7, 19, 26].

Through the various literature waves different approaches and reusable assets are identified and discussed as well. Krueger defines software reuse approaches ranging from: source code components, software schemes or reusable program patterns, reusable transformation systems, application generators up to very high level languages and reusable processors [18]. Morisio et al. distinguish software reuse by different process maturity levels ranging from ad-hoc to systematic. Beside that they also recognize reusable assets varying from code composition to code generation, and from only reusing code to reusing all artefacts [7]. All of them share the distinction between an approach with '*building blocks*', which is based on reusing software products, and the '*generative or reusable processor*' approach, which is based on reusing the process of previous software development efforts, often embodied in computer tools (processors) which automate parts of the development process [7, 18-19].

An article by Griss also captures the different reuse approaches and reusable assets by presenting an incremental view. The view is in line with the results revealed by the initial interviews held for the problem analysis. With this view it remains important to note that the highest levels of reuse are not necessary the best, this rather depends on other factors such as the scope of the domain and the development strategy. An overview of the incremental levels of Griss is presented in Figure 3.

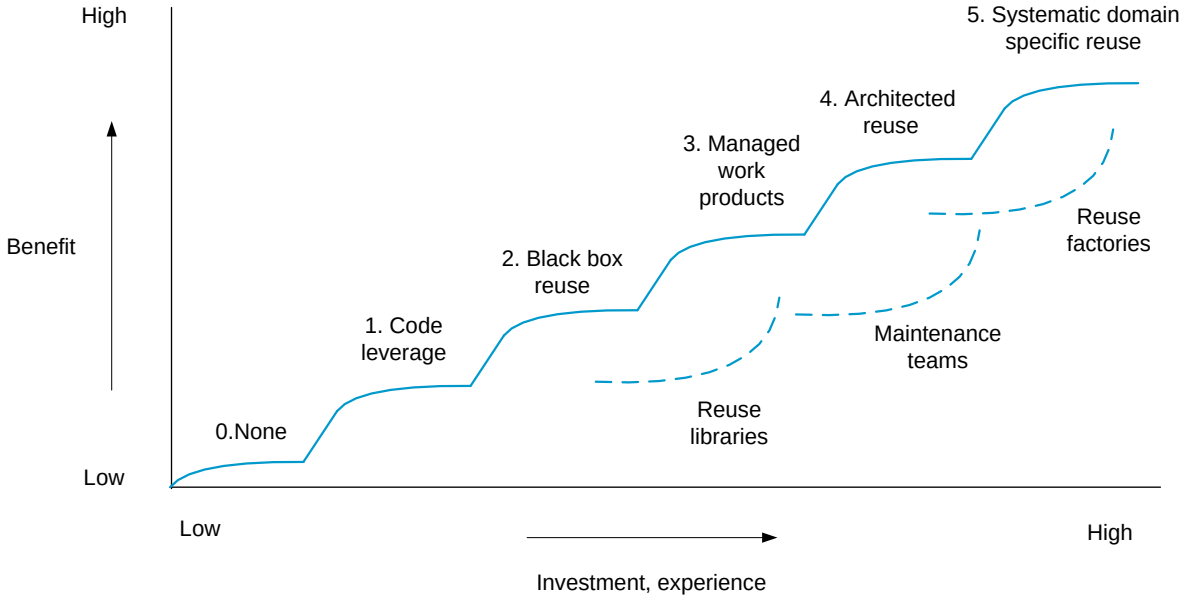


FIGURE 3: INCREMENTAL STAGES OF REUSE ADAPTED FROM [27].

Figure 3 summarizes incremental reuse levels (solid line) and related reuse approaches (dotted lines). The reuse of ad-hoc reuse events with initial benefits is the only level of reuse which can be achieved without investing in software reuse, instead experience of previous projects is used to copy relevant pieces of code. This reuse level is defined as level 0. The first real reuse level presents a form of code-leverage, where pieces of code are made available and can be reused by multiple parties. The pieces of code are made available through the use of a reuse library, providing a central place where the components are stored. The use of these leveraged components is not restricted and can be considered white-box reuse, the code may be adjusted and changed to the specific context in which the component is applied. This strategy works for a while up to then point that multiple copies, each slightly different, have to be managed. The choice can be made to stop using components as white-box and start using them as black-box components instead. Black-box components may no longer be internally adjusted, rather its environment should be adjusted to support the component. Previous research has pointed out that with black-box reuse higher reuse levels can be achieved than with white-box reuse. The reason for this is that there are reduced costs in component maintenance and maintenance across products using these components. However, black-box reuse also has its limitations. The demands on a black-

box component are changing and in order to satisfy a broader range of people it is required that its development is guided into a certain direction. This is what is happening with a managed work product, where a team of users is guiding the development of reusable assets. Beyond this point, it is necessary to architect the components according to the structure of the application that will use them, in order to reach higher reuse levels. The use of architectures and the set of components may be optimized by systematic domain specific reuse [27]. The use of architectures to develop components is in line with the reuse factory approach, also referred to as product line development. A product line is a set of related products set up in a shared manner.

Systematic software reuse is presented in literature as the solution to achieve higher reuse levels. This approach is strengthened by various researchers reporting both productivity and quality gains with more formalized processes [28-29]. Such an approach is often rather static, where the assumption is made that components are predefined and carefully tested before they are populated into a repository. Only a few researchers noted the dynamic aspects of software reuse. Henninger is one of them and he notes that: '*repositories for software reuse are faced with two interrelated problems: acquiring the knowledge to initially construct the repository, and modifying the repository to meet the evolving and dynamic needs of software development organizations*' [30]. Henninger maintains a library approach in the above mentioned citation. When taking a product line approach, additional insights regarding software component evolution are gained. Tomer et al. describe the evolving dynamic aspects along three axes: the development, maintenance and reuse axis [31]. An overview of this model is presented in Figure 4.

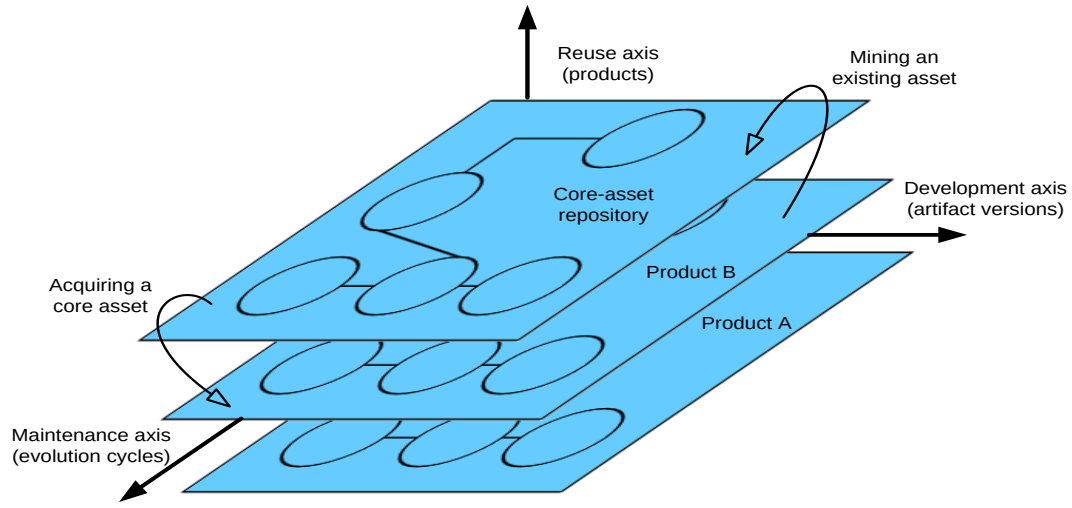


FIGURE 4: THE THREE-DIMENSIONAL EVOLUTION OF A SOFTWARE PRODUCT LINE [31].

The axis of the three dimensional model are explained by an underlying model, also presented in Tomer et al. [31]. The underlying model is presented in Figure 5. For simplicity Tomer et al. transferred the underlying model to a two dimensional model, where development and maintenance are combined into one axis. A key assumption made in this model is that reuse activities cannot be freely transferred between specific products without first storing and cataloguing the assets in a central repository [31].

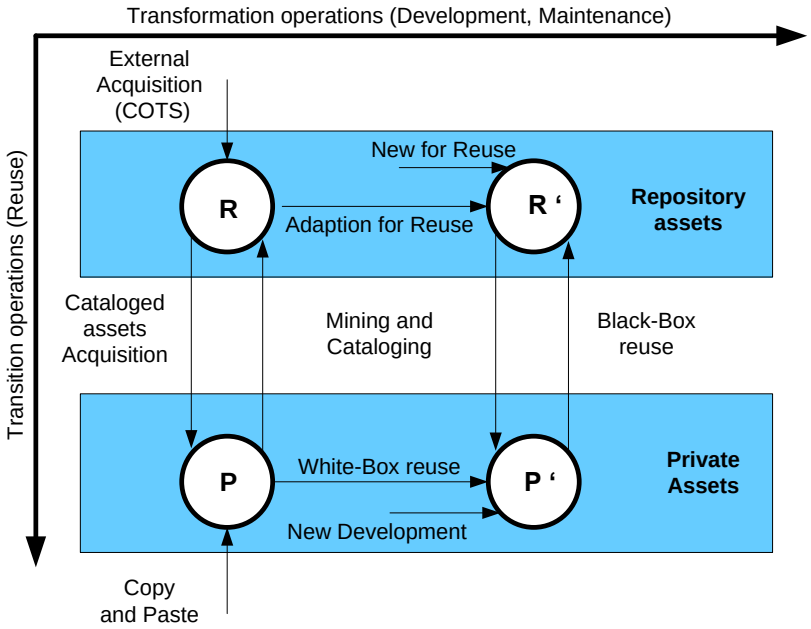


FIGURE 5: ASSETS AND REUSE OPERATIONS [31].

As mentioned earlier, reusable software assets can be used as black-box or white-box components [9]. The model of Tomer et al. [31] explicitly recognizes this distinction. With black-box the component is used 'as-is', which means that it is not adjusted, but instead its environment has to be adjusted to it. With white-box reuse the component itself is adjusted, allowing multiple versions of the same component simultaneously. Higher reuse gains are reported with black-box reuse, which is in line which Krueger's theory of reuse and abstractions. According to this theory reuse can only be successful when abstractions are present hiding the internal logic of components [18].

The key message of software reuse is that the costs related to the reusable assets are lower than when all components are redeveloped from the ground up. This means that with a successful reuse program the average cost of searching for a component, determining its reusability and adjusting or building a new one is lower than the costs of continuously building new components [19]. In practice also indirect costs and gains play a role, such as the possible shorter time to market and the scalability of applications.

During the literature survey it appeared that successful software reuse programs are described by addressing relevant reuse factors. Earlier in this section it already appeared that the factors can be increasingly addressed according to the model of Griss. Beside that the components are often not static, but instead they are dynamic and evolving over time. Based on that premise, the following three requirements for a reuse management tool are defined:

> *Req1: The management tool should address multiple, incremental, levels of reuse in line with the view of Griss [27].*
>
> *Req2: The management tool should address relevant factors describing software reuse issues.*
>
> *Req3: The management tool should address changing reusable assets.*

To conclude this section it is important to note that software reuse itself is not new at all. More than five decades ago the potential of software reuse was recognized as a factor for improving employee productivity and increasing product quality. Despite all efforts, software reuse has never reached its full potential. One of the most plausible explanations is that software reuse is at its best a secondary concern [7]. Furthermore, software reuse does not only include technical factors, but also

organizational factors. The emphasis of recent research is on organizational factors, instead of technical factors. All investments made in software reuse should be justified in such a way that the average cost related to software reuse are lower than the cost associated with building components from scratch. Multiple reusable assets exist, which can be used as static items, but are more likely to be reused as dynamic items, evolving over time.

2.2 Agile development

This section discusses agile development methodologies and related to that the characteristics of agile development environments. A short introduction to the creation of more agile development methodologies is provided. After that agile development methodologies are compared with more traditional methodologies, such as the waterfall model. The comparison is used to provide an insight in the characteristics. Lastly the use of agile development methodologies is evaluated for the adoption of software reuse and relevant requirements are derived.

The development of agile methodologies started two decades ago, back in the 90's, to meet rapidly changing requirements and short product cycles, utilizing: iterative development, prototyping, templates, and minimal documentation requirements [32]. Among these agile approaches are '*Extreme Programming, Crystal methods, Lean Development, Scrum, Adaptive Software development'* and others [33].

All of these methodologies share the common values formulated and promoted by the Agile Manifesto [12]. These are '*individuals and interaction over processes and tools*', '*working software over comprehensive documentation*', '*customer collaboration over contract negotiation*', '*responding to change over following a plan*' [12]. In all stages of the development process, they are characterized to embrace change rather than to fight against it. This is in sharp contrast with more traditional methods, which try to eliminate change in the early stages of the development process, as changes in a later stage would mean significant increases in costs [34]. Such changes can, however, not always be overcome.

In order to create an understanding of what agile development methodologies really are, the example of Scrum is used. An overview is presented in Figure 6 and elaborated on below the figure.

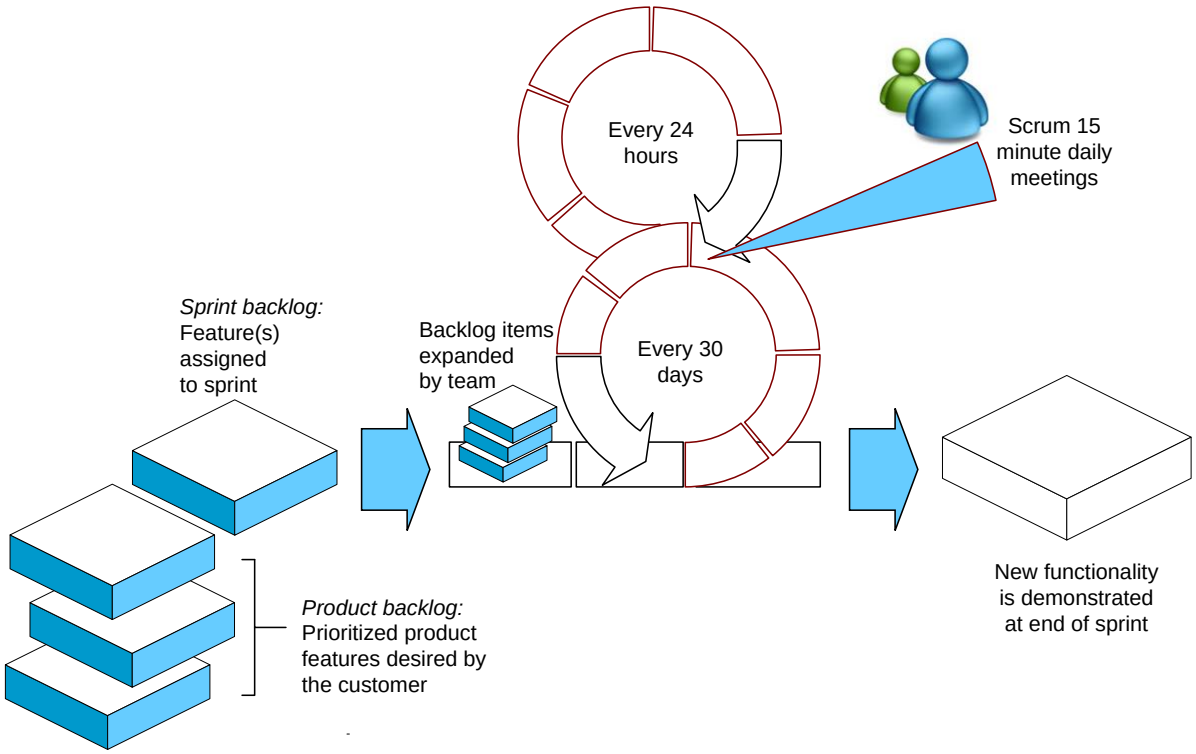


FIGURE 6: SCRUM DEVELOPMENT PROCESS [35]

Scrum is designed to handle rapidly changing business requirements. With this method work is broken down into series of 'sprints' based on a prioritized list of backlog items. The backlog items contain the requirements of features assigned to the sprint. Daily meetings are used to share relevant issues based on the exchange of tacit knowledge. A sprint itself should last one to a maximum of four weeks. After a sprint a working product is delivered and the next sprint can be prepared [32].

The example of Scrum illustrated several characteristics which are common to most agile development methodologies. Iterations are used to develop products, opposing the traditional planned approach. This difference is well illustrated by Robey et al. stating that just as water does not flow uphill, the waterfall model does not allow iterations [36]. By further comparing agile development methodologies with traditional methodologies more insights are gained. Several researchers elaborate on the similarities and the difference between both methodologies, e.g. [37-38]. The differences mentioned by Nerur [38] are presented in Table 1.

Characteristics	Traditional	Agile
Control	Process centric	People centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge management	Explicit	Tacit
Role assignment	Individual , favours specializations	Self-organizing teams, encourages role interchangeability
Communication	Formal	Informal
Customer's role	Important	Critical
Project cycle	Guided by tasks or activities	Guided by product features
Development model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model.
Desired organizational form / structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favours object-oriented technology

TABLE 1: AGILE METHODOLOGIES COMPARED WITH TRADITIONAL METHODOLOGIES [38].

From the perspective of software reuse the first remarkable difference is the use of tacit knowledge over explicit knowledge. Tacit knowledge is knowledge stored in the minds of individuals, where explicit knowledge is stored in artefacts [13]. Agile methodologies make the assumption that most written documentation can be replaced by enhanced informal communications among team members internally, and between the team and the customers, with a stronger emphasis on tacit knowledge instead of on explicit knowledge [39]. By relying on direct face-to-face communication for knowledge sharing purposes, the involved parties benefit from the positive effects of reduced information loss and a decrease in the length of communication chains [40].

Tacit knowledge cannot be captured as it is stored in the minds of individuals. As such, the focus is on explicit knowledge [13]. Explicit knowledge does not seem to have priority with agile development methodologies, nevertheless several artifacts are created. Two reasons exist for creating those artifacts. First of all, artifacts are created to facilitate communication and understanding among project members, and secondly they are created to facilitate communication and understanding between the project members and the customer [38]. The purpose of agile development methodologies is not to ignore documentation, rather it is to prevent over-documentation by having frequent face-to-face meetings. Over-documentation is inherent to long communication chains, in which the writer cannot estimate the prior knowledge of the reader [40]. Furthermore a document can never contain all the elements required for a successful knowledge transfer, so communication remains essential.

The artefacts created during agile development practices do not differ radically from other software development projects. The created artifacts also include requirement specifications (possible in the form of Scrum backlogs), architectural descriptions, source code and test cases. The various artefacts provide different views on the system at different points in time. This means that artifacts do not exist in isolation from each other, but rather are complementary. They all represent the final solution. Likewise, changing requirements in one document possible relate to changes in other documents and have an impact on the final solution or the other way around [9]. The main delivery points for the artifacts are at the beginning or the end of an iteration cycle. At the end of each iteration a release is deployed, representing the state of the solution at that point in time.

Another important aspect of agile development methodologies is the changing roles. Instead of specialization, self-organization teams are used where roles are interchanged. This also enables workers a certain amount of freedom to coordinate their own activities. Social interaction and frequent meetings are stimulated.

Agile development methodologies have proven to be successful in settings where the team size consists of a maximum of 15-20 people [34]. In these setting the focus is on the talents and skills of individuals, where the process is moulded to the needs of individuals and teams, instead of the other way around [41]. Agile methodologies alone are, however, not sufficient in more complex environments such as in large scale projects or distributed environments. This does not mean that they cannot be successful in more complex settings, rather the contrary. Several researchers have identified successful applications of agile development methodologies in such settings, e.g. [42-43].

Ambler defines 8 factors against which agile development methodologies can be scaled to meet the demands of more complex environments [44]. These are:

1. Team size
2. Geographical distribution
3. Regulatory compliance
4. Domain complexity
5. Organizational distribution
6. Technical complexity
7. Organizational complexity
8. Enterprise discipline

The exact way of scaling the factors depends on the individual organization. The organization has to select those factors that are important for them and tailor the methodologies accordingly [44].

Turk elaborates on 6 limitations of agile development methodologies [10], in line with the scaling factors of Ambler. Interesting enough, one of the mentioned limitations is the limited support for building reusable assets [10]. After evaluating the arguments of Turk, it becomes clear that there is a lack of clear understanding of how software reuse can be adapted within agile development methodologies. In fact, Turk confirms the lack of understanding in his paper [10]. The first argument is that agile development methodologies focus on producing code as fast as possible and therefore do

not focus on producing reusable assets, which require substantially larger time investments [10]. While this is true, it is not a problem limited to agile development methodologies, rather for software development in general (see also section 2.1). The second argument is that reusable artefacts require a certain amount of control, as the impact of changes in one artefact affect all applications using that artefact [10]. A part of this is covered by configuration management issues, but the other part has to deal with the control issues of self organizing teams. Communication and control issues are therefore regarded as an important aspect of organizing software reuse in agile development settings. This aspect is also present when reusing in other development environments, where reusable assets also have to be leveraged outside the project scope (see also section 2.1), it is only more reinforced when using agile methodologies.

Based on the comparison of agile development methodologies and more traditional methodologies several characteristics are identified. The characteristics are related to software reuse and based on that the following requirements are extracted:

> *Req4: The focus of the management tool should be on code components*
>
> *Req5: The management tool should not be limited to agile development environments.*
>
> *Req6: The management tool should provide options to make it specific to the individual context of an organization.*
>
> *Req7: The management tool should address relevant communication and control issues.*

The focus should be on code components as agile development methodologies are focussing on producing code as fast as possible. This, however, does not mean that the tool should be limited to it, as additional artefacts such as design documents are still used to support the code components. The management tool should not be limited to agile development environments as recent research has proved that they can be scaled up to more complex environments. Limiting the tool to only typical agile environments would result in a limited applicable tool. The management tool should, however, provide options to make the tool specific for the context of an individual organization. Based on the comparison it already appeared that certain aspects of the management tool are more applicable for a traditional environment than for an agile development environment. Among these are the differences between the focus on process and people aspects. Lastly the management tool should address relevant communication and control issues, as they are indentified as relevant scaling factors.

To conclude this section: agile methodologies promise great benefits over traditional methodologies. By doing so, they focus on characteristics such as people and communication over documentation, imposing constraints on their use over multiple projects. Nevertheless, agile methodologies can successfully be scaled up to meet more complex settings. The literature regarding software reuse lacks a concrete understanding of software reuse issues in agile project settings. While it is not the direct purpose of this thesis, it may require that agile development methodologies have to be scaled up in order to successfully apply software reuse.

2.3 Evaluating existing software reuse frameworks and models

This section discusses relevant existing software reuse frameworks and compares them according to the criteria defined in the previous sections. The goal of this section is to check which models or framework can be used for a software reuse management tool. The articles of Frakes and Terry [45] and Garcia et al. [46] are used as guideline for identifying existing frameworks and models. The various frameworks and models are first presented according to their timeline, after which they are evaluated.

The basis of reuse frameworks and models is traced back to the work of Holibaugh et al. (1989) [47]. The paper describes an initiative set up by the Software Engineering Institute (SEI), to investigate how the benefits of reuse can be obtained. The paper produced a reuse life-cycle approach to initiate these

reuse benefits. In the research approach of Holibaugh et al. a synthesis between an evolutionary and an empirical approach is suggested for investigating software reuse. The evolutionary approach is not elaborated on in the paper of Holibaugh et al., but they do note incremental steps to get from ad-hoc reuse to folklore (systematic reuse).

Koltun and Hudson introduced the Reuse Maturity Model (RMM) in 1991 [48]. The model is introduced to assess organizational processes instrumental to achieving high levels of reuse. The model consists of 5 maturity levels, similar to the levels defined by the CMM of the SEI institute, and 10 dimensions of maturity. The model was set up to enable executive management to view software reuse as an interconnected set of cultural, institutional, financial, technical and legal challenges that must be tackled in a coordinated fashion.

In the same year Prieto-Díaz proposed a model for implementing reuse [49]. The model also defines 5 levels: initiation, expansion, contraction, steady state and expansion. Although the model is rather descriptive it does mention various factors influencing software reuse. These factors are ranging from technical to managerial, economical and legal.

Both models seem to be used as the basis for the STARS (Software Technology for Adaptable Reliable Systems) maturity reuse model, presented at the Fifth Annual Workshop on Institutionalizing Software Reuse in 1992 [50-52]. The reuse maturity model was organized around three parts, an abstract model, self-assessment questions and procedures, and guidelines for strategy formulation. The reuse processes formulated are: reuse planning, reuse creation, asset management and asset utilization. These processes form the basis of the reuse maturity model, where going through these processes can be seen as incremental steps. First a plan is created, after that reusable assets are created, next asset management follows and lastly the assets have to be utilized. Garcia et al. note that the major disadvantage of this approach is that the initial investments are rather high [46].

In 1993, an upgrade of the STARS model was presented by T. Davis [53]. The model is meant for improving an organization's reuse capability and therefore referred to as the Reuse Capability Model (RCM). The RCM consists of two components, an assessment model and an implementation model. The assessment model is used to measure the state of the reuse capability of an organization, based on critical success factors. The papers suggests that strengths (and thus weaknesses) should be identified on which future goals should be set. The implementation model contains four stages: opportunistic, integrated, leveraged and anticipating. Based on mapping characteristics of the results formulated during the reuse assessment the right implementation stage can be chosen as reference point for further developing the reuse capabilities into a reuse-based culture [54].

The work of Wartik and Davis (1999) elaborates on the RCM created in 1993, by proposing a phased reuse adaption model [55]. The insights gained by applying the RCM in about 30 organizations lead to conclusion that risks had to be reduced or avoided during reuse adaption. This resulted in a phased model. The defined phases are 'scoping and assessment', 'definition', 'pilot use and demonstration', 'management and evolution', and 'expansion'. Each phase starts with evaluating reuse goals and a justification should be made for further steps. Although this model is obviously another step forward it remains rather static. Reuse is the result of carefully planned activities.

In 2000, Rine and Nada presented another reuse model called the Reuse Reference Model (RMM) [24]. The goal of the RMM is help organization improve the competitive edge and time-to-market through decreased effort in the software development process and increased product quality. The Reuse Reference Model can then be used to allow organization to compare their practices against the practices of successful companies (best practices). The practices are dived among 5 incremental stages the so called Reuse reference model levels (RRML's). The reuse reference levels are related to a percentage of reuse instead of to maturity levels. Addressing more reuse factors seems to be positively related to the reuse levels. The exact relation between the reuse factors and the reuse levels remains unclear.

In the 2007 paper of Garcia et al. the results of the RiSe (Reuse in Software Engineering Group) project are presented [46]. After Garcia et al. evaluated the vast majority of reuse models mentioned above, they came up with a reuse maturity model. This model consists of 5 maturity levels (ad-hoc, basic, initial, organized and systematic) and 15 reuse factors divided over 4 categories (organizational, business, technology and process). The main goal of this model is to propose a systematic and incremental approach to reuse adoption. At the same time they combine the results of the previous models in the latest known version of a software reuse model.

After evaluating existing software reuse frameworks and models another requirement was added. This is due to the fact that most of the existing framework and models provide little guidance regarding their application and assessment.

Req8: The management tool should support users with guidelines and an assessment method.

The various models and frameworks are compared on the previously defined requirements resulting in Table 2.

Framework	Author	Req1: Supports multiple reuse levels	Req2: Addresses relevant reuse factors	Req3: Addresses changing reusable software assets	Req4: Addresses code components	Req5: Framework is not limited to agile	Req6:Options to make it specific for agile methodologies	Req7:Adresses communication and control issues	Req8: Supports usage guidance
Reuse lifecycle	Holibaugh et al. (1989)	-	+	+/-	-	+	-	-	+/-
RMM	Koltun and Hudson (1991)	+	+	-	-	+	-	-	-
Model for implement-ting software reuse	Prieto-Díaz (1991)	-	+	-	+	+	-	+/-	+
STARS RMM	M. Davis (1992)	+	+/-	-	+/-	+	-	-	+/-
RCM	T. Davis (1993)	+	+	-	+/-	+	-	-	+/-
Phased adaption model	Wartik (1999)	-	+/-	-	+/-	+	-	-	+
RMM(L)	Rine and Nada (2000)	+/-	+	-	-	+	-	-	-
RiSe Maturity Model	Garcia (2007)	+	+	-	+/-	+	-	+/-	-

TABLE 2: EVALUTION OF EXISTING REUSE FRAMEWORKS AND MODELS

Based the evaluation as presented in Table 2, several conclusions can be drawn. First of all, there is no existing framework or model that meets all the requirements of the reuse management tool. Secondly, the RiSe maturity model [7] satisfies most criteria, but also fails at crucial points, namely: in addressing changing reusable assets and providing practical guidance. The assessment method was out of scope at when the Rise Maturity model was presented and the model was never validated. No additional papers were found describing such an assessment method or validation related to the Rise Maturity Model. Changing reusable assets are also barely addressed by the model. Some reuse maturity models present configuration or change management as a way of managing changes, but in the RiSe maturity model this aspect is rather buried in other factors. Some characteristics of changing reusable assets are recognized in factors such as ‘previous development of reusable assets', 'origin of the reused asset' and 'technology support'. Nevertheless this point is rather weakly addressed. The strong point is that it elaborates on the work of previous reuse maturity models. Thirdly, almost all the reuse models and frameworks use the Capability Maturity Model (CMM) of the Software Engineering Institute (SEI), see e.g. [56]. Although many models and frameworks claim to use CMM, it is actually used as guidance instead of a prescription. Ted Davis mentioned that *'maturity as an indicator of capability'* and *'the use of maturity levels'* is not present in his model. The same is true for many other reuse frameworks and models. Fourthly, the reuse factors mentioned by the existing reuse maturity models resemble each other, but at the same time lack a systematic theoretical foundation. The reuse factors can serve as a basis for describing software reuse.

Summarizing this section existing reuse maturity models fail to meet all the requirements. The model presented by Garcia et al. provides a good starting point, although the model itself was never finished. A more systematic literature review to software reuse factors should serve as the basis for the software reuse management tool.

2.4 Summarizing requirements

Based upon the literature survey eight requirements were extracted. These requirements are briefly discussed and summarized in this section.

The first requirements states that the software reuse management tool has to address multiple, incremental levels of reuse in line with the view of Griss [27]. The incremental levels of reuse defined by Griss are: code leverage, black-box components, managed work products, architected reuse and domain optimizing reuse.

The second requirement states that the management tool has to address relevant factors describing software reuse. Although some of these factors have been addressed by the literature review, a more systematic approach is needed to identify them. This is elaborated on in chapter 3.

The third requirement states that the management tool has to support reusable artifacts, which are inherent to changes. The changes are inevitable when working with agile methodologies.

The fourth requirement states that the management tool should focus on code components. The focus of agile methodologies is on producing code as fast as possible, where documentation is only used as support. Similarly the management tool should focus on code components, where documentation is only used as support.

The fifth requirement states that the management tool should not be limited to agile methodologies alone. This seems to be contradicting at first sight, but agile methodologies can be scaled up to more complex environments. To make the management tool applicable for these environments as well, the focus has to be at a more general level.

The sixth requirement states that the management tool should provide options to make it specific for agile development environments. Obviously some characteristics are more important when working with smaller teams then when working with large teams. The tool has to provide a way to make the difference visible.

The seventh requirement states that the management tool has to address relevant control and communication issues. Where face-to-face communication and co-location are sufficient in smaller projects additional attention has to be paid to control and communication issues in larger projects or multiple smaller projects.

The eighth and the final requirement states that the management tool has to provide guidance in its application through an assessment method. Existing reuse framework and models provide little to no guidance in its application. The management tool should include both a reuse maturity model or framework and an assessment method.

The evaluation of existing reuse maturity models and frameworks based on these requirements did not result in satisfying findings. Therefore the next chapter describes the proposed reuse management tool, which has to meet the requirements formulated in this chapter.

3. Reuse management tool

This chapter proposes a reuse management tool suitable for organizations working in agile development environments. Its goal is to provide those organizations a roadmap for managing their reusable assets through various incremental stages of software reuse. The first section of this chapter elaborates on the chosen solution approach. The second section describes the components of the management tool. In the sections after that, the components of the management tool: the maturity levels, the reuse factors and the practices with the assessment method are elaborated on. In the final section guidelines are provided for the application of the management tool.

3.1 Solution approach

Setting up a reuse management tool can be done in several ways. Therefore this section elaborates on the possible solution approaches in paragraph 3.1.1, their advantages and disadvantages are discussed in paragraph 3.1.2 and the chosen approach is discussed in paragraph 3.1.3.The chosen approach is worked out throughout the rest of this chapter.

3.1.1 Possible approaches

The first approach consists of using an existing reuse management tool for this particular research setting. The second approach consists of combining existing reuse management tools. This will enhance the coverage of a newly created tool. The third approach is extending an existing reuse management tool. New elements, specific to this research setting, can be added to the management tool. The fourth approach combines elements of existing reuse management tool together with new elements, into a new management tool. The fifth approach is Greenfield, which defines a completely new reuse management tool, not based on previous created tools or models.

3.1.2 Advantages and disadvantages

In this paragraph the possible approaches are examined in more detail. The approaches are summarized in Table 3 with the advantages and disadvantages stated. The approaches are explained in the text below.

Approach	Description	Advantage(s)	Disadvantage(s)
1	Use of existing reuse management tool.	Use of proven and accepted theories. Easy to validate	Not specific for the research setting.
2	Combining existing reuse management tools.	Use of proven and accepted theories. Relatively easy to validate.	Does not directly match the research setting.
3	Extending existing reuse management tools.	Based on proven and accepted theory. Can be made specific for the research setting.	Limited to the basis of the used reuse management tool.

Approach	Description	Advantage	Disadvantage
4	Combining and extending several existing reuse management tools	Based on proven and accepted theories. Can be made specific to the research setting. Basis literature available.	More difficult to validate. Risk of making it too complicated.
5	Greenfield development	Completely designed for the research setting.	Difficult to validate. Low acceptance level.

TABLE 3: ADVANTAGES AND DISADVANTAGES OF POSSIBLE APPROACHES.

Although existing literature contains, implicitly or explicitly, elements related to reuse management tool suitable for agile development environments, there is no tool focused on guiding the implementation of reuse in such settings. Therefore no existing tools can be used. For the same reason a combination of two existing reuse management tool does also not lead to a solution.

The remaining approaches can all be used to set up a software reuse management tool. The third approach can be used as a basis, but would limit us to the structure provided by that management tool. The fourth approach is more labour intensive and combines reuse management tools with new elements. Although it provides a solid basis it also has the risk to run out of scope or become too difficult to integrate and validate. The fifth approach can also be used to set up a new and specific reuse management tool. This approach should, however, only be chosen when the other approaches do not fit, because of its disadvantages.

3.1.3 Chosen approach

The chosen approach for setting up the reuse management tool, is the fourth approach, which combines elements from several existing reuse management tool and new elements into the proposed reuse management tool. The reason for not choosing the third approach is that the similarities and thus the overlap between existing reuse management tools is great. All of them are, directly or indirectly, related to the SEI CMM levels combined with a relevant, but often different, set of success factors. Combining the sets of success factors can provide a more solid basis for a reuse maturity model.

3.2 Management tool structure

The structure of the management tool is based on two integrated components. The first component is a reuse maturity framework and the second component is the assessment method. The reuse maturity framework provides the structure to describe software reuse over multiple incremental stages, where the assessment method operationalizes the maturity framework by providing a tool to systematically assess elements of the framework. The two components are integrated as the assessment method is based on the reuse maturity framework and the other way around the framework is operationalized by the assessment method. An overview of the management tool structure is provided in Figure 7 and elaborated on below.

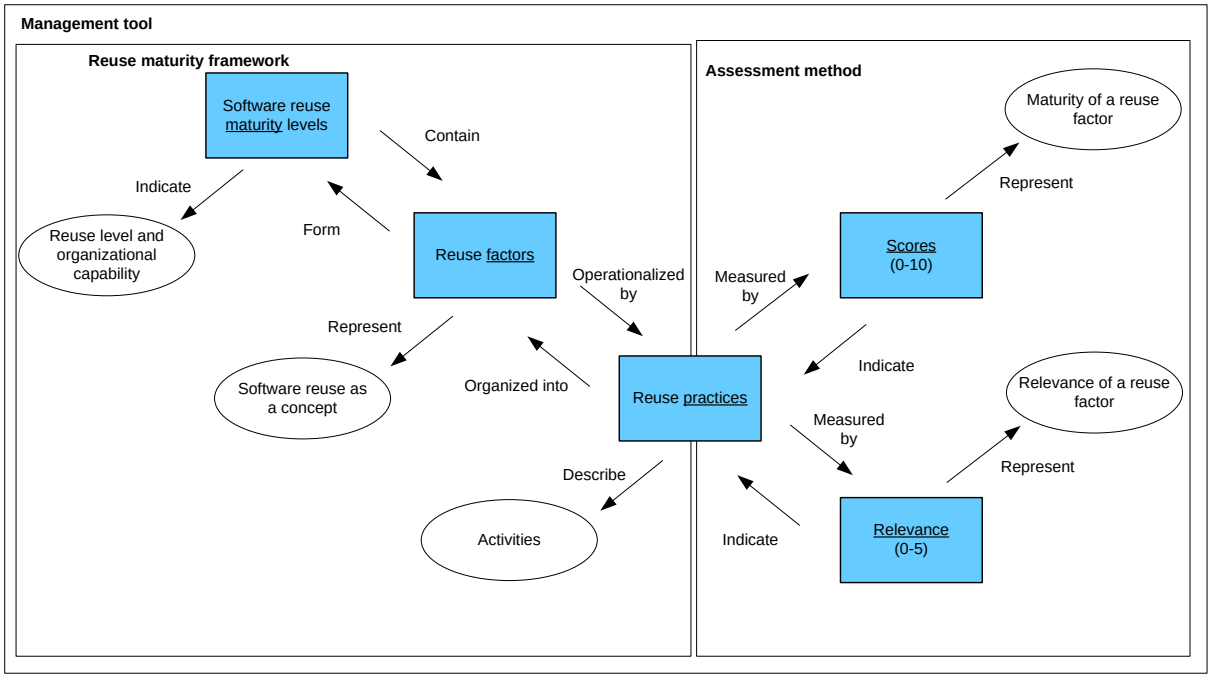


FIGURE 7: MANAGEMENT TOOL STRUCTURE DERIVED FROM [57].

The basic structure is derived from the Software Process Improvement (SPI) framework defined by Niazi et al. [57]. The original framework defines maturity levels, critical success factors and practices for assessment. The process improvement framework is described in Figure 7 as the reuse maturity framework. The maturity levels are make specific to software reuse maturity levels, the critical success factors are renamed to reuse factors and the practices are made specific to reuse practices. The paper of Niazi et al. [57] also describe how the reuse practices can be operationalized through the use of an assessment matrix. This matrix based approach is originally proposed by Daskalantonakis [58]. The matrix based approach adds a fixed score to a combination of reuse practices, which in turn represent a reuse factors at a maturity stage. The matrix based approach is extended with a relevance variable and taken as the assessment component in the structure of the management tool. The relevance variable is added to be able to make the assessment method specific for an individual situation, which is possible a situation where an organization works with agile development methodologies, but not necessary. The structure elements are represented as blue boxes in Figure 7. The relation among the elements is clarified by the arrows. The circles provide a brief description of the structure elements.

The structure of the first component of the management tool shows great similarities with existing reuse maturity models and frameworks. These also describe maturity levels and reuse practices. Implicitly the models and frameworks also define reuse practices, when they define the characteristics of a reuse factor at a certain stage. These insights can be used to fill in the elements of the management tool. The existing reuse maturity models and frameworks, however, fail in putting the results into practice. As such the assessment method for software reuse is a new component, and an addition to existing literature.

Lastly, it should be noted that existing maturity frameworks and models claim to be based on the CMM stages, but actually the defined structure is slightly different. CMM(I) uses Key Process Areas (KPA's), instead of reuse factors or critical success factors [59]. The details are further elaborated on in section 3.3, which presents the maturity stages. For this section it is important to note that the foundation, created by reuse factors, does not necessary lead to a weaker theoretical basis. Niazi et al. [57] even claim the contrary. Arguments for using critical success factors (here renamed to reuse factors), instead of KPA's include:

- Different studies have confirmed the value of CSF's in the field of information technology.
- Implementation programs require real life experiences where on learns from mistakes and continuously improves the implementation process. CSF's are based on relevant real life experience and can provide practical guidance.
- The CSF approach is regarded as a richer view when considered in the context of their importance for each stage of the implementation process.

For this research the distinction between CSF's and KPA's is relevant, but does not lead to the decision whether one is better and should be used over the other. The choice is made to use existing insights, based on a large body of relevant research as the basis for the management tool. These existing insights are based on reuse factors instead of on process areas, so it is a logical choice to base the model on the reuse factors as well. Furthermore one of the requirements set up for the proposed management tool is to address relevant reuse factors, instead of KPA's. The comparison with the CMMI model is regularly made throughout this thesis, to clarify the similarities and the differences. Beside that, CMMI is considered a useful guideline, as it is considered a proven and accepted way to fill in the maturity stages, as also explained in section 3.3.

As already became clear from several references in this section, section 3.3 elaborates on the reuse maturity levels. The reuse factors are explained in section 3.4. The reuse practices can also be derived from section 3.4, but their characteristics are summarized in section 3.5. The assessment method, with the use of a fixed score and a relevance variable, is discussed in section 3.6. The interpretation of the results is discussed in section 3.7.

3.3 Software reuse maturity levels

Software reuse maturity levels are the first element of the management tool structure. Software reuse maturity levels present an evolutionary plateau for incrementally adapting software reuse and managing related issues. The correct level of adapting software reuse depends on the context of an individual organization, which means that not all organization can reach the highest reuse levels. This section first elaborates on the incremental software reuse adaption as suggested by the interviewees. The results are presented in section 3.3.1. After that the incremental adaption of existing reuse maturity models and frameworks is discussed in section 3.3.2. Lastly the used maturity levels for this thesis are elaborated on in section 3.3.3.

3.3.1 Incremental software reuse adaptation according to the interviewees

The interviews held at the start of this research provided useful insights for the incremental adaption of software reuse.

The interviewees expect that higher reuse levels can be achieved in the future. The reason for this is two-fold. First of all they note that the solutions for a domain are becoming increasingly mature. More and more customers are requesting similar solutions resulting in a set of components which can be used out of the box. Secondly they note that the focus on a market niche is continuously being redefined. The focus can be set on a new market niche, but can possible also lead to a refocus on an existing market niche. When the choice is made to refocus on an existing market niche, two business units divide the market niche in two logical parts and set their focus points. By doing so the similarities among customers is also likely to increase, possibly leading to higher levels of reuse.

On top of that some of the interviewees noted that they are currently working with white-box components and tend to go towards black-box components. The black-box components can no longer be adjusted, but some variables can still be set, because they are set up in a generic way. The use of black-box components seems to be possible because the components are becoming more mature. Not only is there a shift in use from white-box to black-box components, there is also a shift noted in functionality. Where functionality is first based on more general functionality, used in almost all applications, the functionality is shifting towards becoming domain specific.

The views of the interviewees regarding higher reuse levels are in line with existing literature. Rothenberger noted that organization are likely to invest more in reuse activities, when they are in a narrow domain [60]. The reason for this is that number of times that individual components are reused is higher than in wider domains. Beside that he noted that the experience of developers in a narrow domain is likely to lead to a better quality of the set of components as a whole [60]. These statements explicitly indicate that a narrower and a more mature domain lead to higher reuse levels. The shift from white-box to black-box components is recognized in the article of Griss [27]. Griss notes that black-box components are not an end by itself, as several parties will push the component in different directions. The result is that the black-box component will become work-products, which have to be managed actively. The model of Griss is explained in section 2.1 and will be used as the basis for defining higher reuse levels in section 3.3.3.

3.3.2 Incremental software reuse adaptation according to existing maturity models and frameworks.

Existing reuse maturity models and frameworks use the Capability Maturity Model (CMM) to describe incremental steps of software reuse. The reason for this is that existing literature assumes that higher reuse levels can only be achieved with a more systematic software reuse process. The CMM is a maturity framework which describes incremental steps according to more mature processes. Therefore it is a logical choice to use the CMM to describe incremental steps of software reuse.

The latest version of CMM is the Capability Model Integration for Development (CMMI-DEV) [59]. Similar to the original CMM model it defines maturity stages ranging from ad-hoc, to defined, managed, measured and ends with optimizing. The stages are incremental and can be viewed both continuous as staged. Incremental means that the steps are building on each other, so the next stage has got all the characteristics of the previous stage, plus more. The characteristics of a stage are defined by Key Process Areas (KPA's). Each KPA with the CMMI model is assigned to fixed maturity stages. With the staged approach all KPA's are present at a certain stage. One cannot move on to the next stage unless all conditions are met. With the continuous approach the requirement to satisfy all conditions of a certain stage before moving to the next, does not have to be met.

The structure of CMMI is based on maturity levels, KPA's, specific and generic goals and related practices. The KPA's are allocated to maturity levels. So if a certain KPA is met it automatically leads to a maturity level. Each KPA has a set of generic goals and specific goals. The generic goals are goals which are set for all KPA's and the specific goals are, as the name already suggests, specific for that KPA. The goals can be measured by related practices. The practices can be based on related sub-processes work products and other characteristics. A fundamental difference with the reuse maturity stages is that they use reuse factors instead of KPA's. As such CMM or CMMI is used as a guideline for reuse maturity models and frameworks, instead of a prescription. This is recognized as a limitation of the Rise Maturity Model [7].

Beside the previously explained structure, CMM(I) also maintains four representation to support continuous integration, which are Process management, Project Management, Engineering and Support [61]. The representations can be used to identify specific improvement areas. This concept of using multiple representations is also used in this thesis, as will be explained in section 3.4.2.

Summarizing the evaluation of incremental adaption based on existing reuse maturity levels and frameworks the conclusion can be drawn that the CMM(I) model is a logical choice, but also that it has its weaknesses. Incremental reuse adaption has been associated with a more systematic reuse process, but not all the reuse factors necessary touch on a process area or are process related.

3.3.3 Used maturity stages

The used maturity stages in this thesis are a combination of the incremental levels defined by Griss and the maturity stages defined by the CMMI model. This is not a radical choice, but instead a logical one. Both approaches require that additional investments are made to attain higher levels. At these higher levels also higher benefits are expected. Not all organizations are able to reach the higher reuse

levels and it should therefore not be a goal as such. Only some organizations have succeeded in reaching high maturity levels in the past. The vast majority of organizations have a maturity of level 2 or level 3 in software development processes, so it is also not expected that the vast majority will reach the highest software reuse levels. The choice to combine the incremental levels of Griss with the CMMI model is in line with the requirements set up in the previous chapter, stating that incremental levels of software reuse have to be addressed.

The combination of the incremental stages of Griss and the CMMI model is indirectly supported by existing literature. Morisio et al. note that a library approach can be good with 30% reuse, but also that a product line approach can be bad with 70% reuse [7]. This statement emphasizes that the right level of reuse depends on the chosen reuse approach. In other words the investments made in a certain reuse approach have to be gained back by the reuse activities. Furthermore it indicates that a more systematic approach, such as with a product line, is associated with higher reuse levels.

An overview of the mapping between the incremental levels defined by Griss and the maturity stages defined by the CMMI model is presented in Table 4. The used maturity stages are briefly elaborated on below the table.

Used maturity stages	**Based on incremental stages of software reuse defined by Griss**	**Compliant with SEI CMMI Maturity stages (staged)**	**Compliant with SEI CMMI Capability levels (continuous)**
Level 0: **No reuse**	-	-	*Level 0:* Incomplete
Level 1: **Ad-hoc reuse**	L*evel 0:* None	*Level 1:* Initial	*Level 1*: Initial
Level 2: **Managed reuse of leveraged code.**	*Level 1:* Code leverage; *and* *Level 2:* Black-box reuse.	*Level 2:* Managed	*Level 2:* Managed
Level 3: **Defined reuse of work products.**	*Level 3:* Managed work products	*Level 3:* Defined	*Level 3:* Defined
Level 4: **Quantitatively managed architected reuse.**	*Level 4:* Architected reuse	*Level 4:* Quantitatively Managed	*Level 4:* Quantitatively Managed
Level 5: **Domain optimizing reuse.**	*Level 5:* Systematic domain specific reuse.	*Level 5:* Optimized	*Level 5:* Optimized

TABLE 4: OVERVIEW MATURITY STAGES.

As can be derived from Table 4, *level 0* indicates that no reuse is taking place at all. The organizations and its individual members have not become aware of reuse events yet. In practice this is likely to almost never be the case, as employees have some previous experience or find existing knowledge on the internet, which is used in new projects.

At *level 1* ad-hoc reuse events can be found. Ad-hoc reuse events are reuse events effected by individuals. The reuse events are not coordinated and not monitored. No reuse processes are present. The individual is driven by previous experience, where code is often scavenged. Scavenged code is code that is copy-pasted from previous projects. Every organization is expected to have a form of ad-hoc reuse.

At *level 2* the reuse process is characterised as managed. The basic infrastructure is installed to let reuse events take place. Its processes are more structured and can be controlled and monitored. Pieces of leveraged code, or rather code components, are used at this level.

At *level 3* the processes are standardized and thus defined. The code components are managed and controlled by a group, guiding it in the right direction. Additional elements of a systematic reuse process are implemented.

At *level 4* software reuse is defined as quantitatively managed architected reuse. Architected indicates that an architecture is used to define and fit the code components, which is in line with the software product line approach. Quantitatively managed reuse means that the reuse processes are controlled using statistical and other quantitative techniques. The reuse processes are managed throughout its entire lifecycle.

In the final maturity stage, *level 5*, the reuse events are optimized for a specific domain. Not only do the reusable assets have to fit within the architecture, but the architecture is also guiding the development of reusable assets. The reusable assets have to fit within the architecture and empty areas of the architecture can be filled with components. Each new product is composed of or created by reusable assets. The development of reusable assets is exploited by optimizing one or more reuse factors.

The combination of the incremental reuse levels defined by Griss [27] and the maturity stages of the CMMI model [59] lead to a logical composition resulting in the used maturity stages. Both approaches are used as a guideline, as none of them explicitly define the maturity related to reuse factors. The allocation of the reuse factors to a maturity stage will become clear in the next section.

3.4 Reuse factors

The second structure element of the reuse management tool is the reuse factors. The reuse factors are success factors or factors of influence for software reuse. Various studies have identified such factors and use them to describe software reuse. Before elaborating on the reuse factors themselves, the organization of the reuse factors in the maturity model components of the management tool is discussed. After that the methodology used to identify the reuse factors and a brief overview of the factors themselves is presented. Lastly the identified reuse factors are elaborated on in more detail, the results follow the organizational structure provided by the BTOPP model. When elaborating on the reuse factors they are also mapped against the maturity stages presented in the previous section. The identification of reuse factors is in line with one of the requirements formulated in the previous chapter, stating that relevant software reuse factors should be addressed in the management tool.

3.4.1 Reuse factor organization

The organization of reuse factors provides the reader and the user of the framework practical guidance for assessing relevant areas of the framework. To provide such a structure the concept of software reuse is visualized in pillars and principles, as presented in Figure 8. .The principles provide the foundation of the software reuse concept, which are organized around different pillars. These pillars provide the structure to describe the principles. The principles are used as a metaphor for the reuse factors and the pillars are used as a metaphor for the organization of the reuse factors.

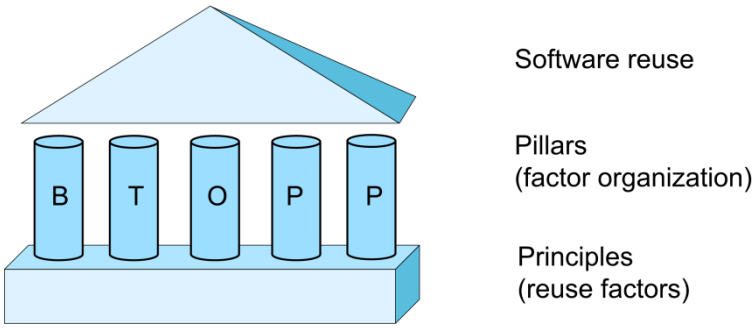


FIGURE 8: SOFTWARE REUSE PILLARS AND PRINCIPLES

Previous reuse maturity models and frameworks do not always organize their reuse factors into categories, making it difficult to address relevant areas. When looking at the model of Garcia et al. they do distinguish different perspectives, another term for describing the organization of reuse factors. The different perspectives identified by this model are: '*organizational*', '*business*', '*technology*' and '*process*' [46]. The work done by Lucrédio et al. [62] implicitly recognize the same four pillars, when structuring their research results. Nerur, who focuses on agile methodologies instead of software reuse, uses the pillars '*management & organizational*', '*people*', '*process*' and '*technology*', suggesting a slightly different mix of the pillars [38]. Returning to the work of Morton [63] a synthesis between the various defined pillars is found. An overview of the model of Morton is presented in Figure 9.

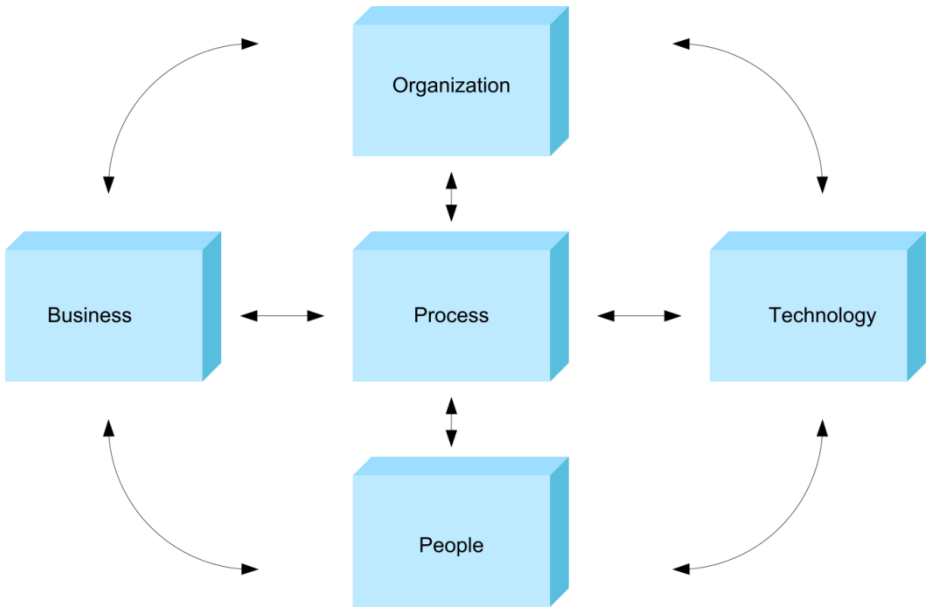


FIGURE 9: BTOPP MODEL OF MORTON [63].

The BTOPP model stands for the business, technology, organization, process and people model. Its original purpose is to blend IT solutions with the '*business system*', its purpose for this thesis is to define the pillars as a common language for describing a '*software reuse system*'. The software reuse system is nothing more than a synonym for software reuse or a software reuse program. The BTOPP model does not only provide the structure to define the pillars, but also takes the dependencies among the pillars into account. While the pillars provide the structure for organizing the concept of software reuse, the pillars should not be considered stand-alone, but rather as an integrated whole. The BTOPP model is used in the remaining of this thesis, to provide structure for describing the reuse factors.

3.4.2 Identification of reuse factors

As mentioned several times throughout this thesis, existing reuse maturity models and frameworks can be used as the basis for identifying relevant elements, such as the reuse factors. When comparing existing reuse maturity frameworks and models it becomes evident that they overlap for a great part, but also that they have their own differences and interpretations. Based on this observation the decision is made to identify the reuse factors through a systematic literature review. This sub-section presents the method used for the identification of the reuse factors and finishes with a quick overview of the results. The results themselves are further elaborated on in the sub-sections following this one.

To make sure the quality of the theoretical basis is high and the scope of the literature review is narrowed down, the top 25 information system journals are taken into the literature review. An overview of the top 25 journals and the used databases is presented in Appendix C. The method used for the systematic literature review and the identification process is presented in Table 5. This table also presents the results of the various steps of the methodology. The results are elaborated on below the table and an overview of the identified reuse factors is presented in the final paragraph.

Step	Stage	Description	Result
1.	Searching	Searching the top 25 journals with relevant queries.	222 papers
2.	Searching	Removing false positives.	58 papers
3.	Searching	Selecting the empirical papers.	13 papers
4.	Searching	Forward and backward searching.	24 papers
5.	Extracting	Summarizing factors per paper.	List of factors per paper.
6.	Extracting	First round of merging and grouping of reuse factors. In this phase synonyms and obvious similar requirements are merged. The results are grouped around the BTOPP model.	25 factors
7.	Extracting	Second round of merging and grouping of reuse factors. In this stage also the requirements of the previous chapter are checked.	15 factors

TABLE 5: IDENTIFICATION METHODOLOGY AND RESULTS

The first searching step requires some preparation, because the search queries have to be defined before any searching process can take place it all. Furthermore the results of the search queries have to be limited to the top 25 journals. The databases, which cover the top 25 journals and where the search queries are performed on, are Scopus, Web of Knowledge and Communications of the AIS. The limitation to the top 25 journals is necessary as the databases provide more results than needed to cover the top 25 journals.

The search query ‘software’ and ‘reuse’ resulted in more than 9000 hits on Scopus alone. This was a problem as Scopus is only able to export the first 2000 hits, making it a difficult task to handle the remaining 7000 hits. Web of Knowledge can export 500 hits per search query, but allows the selection of the export range. Communications of the AIS has no export support and has to be searched manually. After evaluating the databases with related export options, a combination of search queries was tried to evaluate the best possible results.

The evaluated search queries are all a combination of software reuse with relevant topics of this thesis including: ‘framework’, ‘agile’, ‘model’, ‘staged’, ‘maturity’, incremental’, ‘reuse of software’. Also the terms ‘reuse of software’, ‘reusable software’ and ‘component reuse’ where evaluated. Based on the evaluation the choice was made to limit the search to the queries ‘software reuse’ or ‘component reuse’. The first query term ‘software reuse’ assumes that the two words are always placed together, which often seems to be the case. Since the first paper about the software crisis in 1968 [6], the words ‘software’ and ‘reuse’ are indeed placed together often. Beside that the separation of words leads to the large amount of false positives as noted earlier in the example where a query returned 9000 hits on Scopus. The second query ‘component reuse’ was added, as component reuse is considered the most important instantiation of software reuse. Adding a third term based on another instantiation of software reuse, such as ‘product line’, did not result in significant additional results. Lastly, different verb forms of reuse are ignored, as infinitive seems to be used most.

Searching the database with the queries ‘software reuse’ or ‘component reuse’ resulted in a total 2740 hits. These hits also contain results not belonging to the top 25 journals. Removing these hits resulted in a total of 304 hits. Unfortunately, there is some overlap in the coverage of the used database, so duplicates are found. Removing the duplicates leads to a result of 222 high quality papers in the first step of the searching process.

The second step of searching consists of removing the false positives. False positives are results that were found with the selected queries, but did not relate to software reuse as defined in this thesis. The false positives are identified by scanning titles, reading abstracts and most often also by readings its contents. The result of this step of searching is 58 papers. Especially the results of the AIS were less relevant, as its search engine apparently scanned the full text of an article, where the other databases only looked in titles and abstracts for relevant key words.

The third step of searching consists of selecting only the empirical papers. The empirical papers are considered the only source for a solid theoretical basis. By reading the contents of these articles, for as far as that had not been done yet, only those articles were extracted which contain case studies, practical surveys, controlled experiments or experiences reports. The result of this step of searching is 13 relevant empirical papers.

The final step of searching consists of forward and backward searching. Forward and backward searching is used to extend the data set with other quality papers outside the top 25 IS journals. On top of that the results of the literature survey to existing software reuse maturity models and frameworks were also included. The forward and backward searching resulted in 11 additional empirical papers. So the total amount of empirical papers is 24. An overview of these papers can be found in Appendix D.

The first step of the data extraction stage gathers the reuse factors per paper. This was done to systematically analyse the factors. For each identified factor its relation with software reuse was also recorded. A relation can be positive or negative. A positive relation implies that a reuse factor is confirmed as a factor influencing software reuse. A negative relation implies that a reuse factor does not or not necessary influence software reuse. No factors were identified that negatively influence software reuse in the sense that they reduce the effects of software reuse.

The second step of the data extraction stage combines the results of the individual papers into a large table. Similar reuse factors and synonyms are merged together into a single reuse factor. This yielded in 25 reuse factors. An overview of the results is presented in appendix E. The results of this table are also mapped against the BTOPP model, the result is presented in Appendix F. The BTOPP model is explained in the previous section.

In the third and the final step the results of the previous data extraction step are analysed more thoroughly. In this step the relations between the identified reuse factors are analyzed in more depth. At this point the factors which are considered as variables moving on the background and are

influenced by other factors are removed or merged with another reuse factor. The factors are regrouped and sometimes rephrased with the purpose of creating independent and operationalizable reuse factors. The grouping of the reuse factors is done through the use of the BTOP model. The requirements formulated in the previous chapter are also taken into account. These state that communication and control issues should be addressed. The communication and control issues are taken in two reuse factors. The reuse factors are not new, but emphasize characteristics of existing reuse factors. At this stage 15 reuse factors are identified. These are presented in Table 6 with a brief description.

Nr.	BTOPP element	Reuse factor	Brief description and related factors
1.	Business	Domain focus	The domain focus is used as an indication for the level of similarities among products in a certain domain. The domain is the environment in which a software development organization is operating and producing solutions for. The domain focus is also referred to as the product family approach and it includes implicitly the origin of reusable assets.
2.	Organization	Top management support and instrumental mechanisms	Top management support is taken as an indicator for the level of top management support. Top management support can be both passive and active. Top management can utilize instrumental mechanisms to promote or enforce desired behaviour. The instrumental mechanisms are a combination of other reuse factors, including the use of reuse champions, sample solutions, rewards and incentives, reuse education and training.
3.	Organization	Organizational structure and reuse roles	The organizational structure for software reuse elaborates on the separation of producers and consumers of reusable assets. The producers or reusable assets create the components, where the consumers use the reusable assets in building new applications. Beside the consumers and producers also several other reuse roles are identified.
4.	Organization	Communication channels and organizational support	The identification of communications channels and organizational support is also identified as a reuse factor. This factor is embodied in a systematic software reuse process and includes the communication of change requests. The factor is emphasized by agile methodologies, as stated by one of the requirements set up in the previous chapter.
5.	Process	Reuse planning	Reuse planning is an indicator of a systematic software reuse process. It also includes the reuse factor which describes origin of a reusable asset. Domain analysis may be deployed to perform systematic reuse planning. Domain analysis is a tool which evaluates the domain for potential reuse opportunities.
6.	Process	Reuse measurement and cost models	The measurement of reuse activities is also an indicator of a systematic software reuse process. When the reuse activities are measured it is possible to link them to cost models. The result is that systematic cost benefit analysis of individual components can take place.

7.	Process	Requirements management	Requirements management is also an indicator of a systematic software reuse process. Managing the requirements over multiple projects can provide a good basis for identifying their similarities.
8.	Process	Quality management (internal)	Quality management is once again an indicator of a systematic reuse process, but this indicator was also explicitly mentioned as a reuse factor in several papers. The use of quality models and ranking systems can provide the basis for quality management.
9.	Process	Supplier management (external)	Supplier management is also related to a systematic reuse process, but then with a focus towards the outside world. Supplier management deals with the acquisition of external assets, e.g. from black-box components markets. Related reuse factors include legal issues and the use of certifications.
10.	Process	Configuration and change management	The last aspect of a systematic reuse process is embodied in configuration and change management. This reuse factor is not only an aspect of a systematic reuse process, but both configuration and change management are recognized as separate reuse factors in literature. In this thesis they are combined, as they show great similarities when they are operationalized.
11.	People	Producer skills and experience	Producer skills and experience relates to all the knowledge and practices which have to present at the producer side. This includes development and programming experience. A certain amount of domain knowledge is also required to be able to estimate what is project specific for a component and what can be made generic.
12.	People	Consumer skills and experience	Consumer skills and experience relates to all the knowledge and practices deployed at the consumer side. This includes knowledge of the reusable assets and best practices for integrating them into new applications.
13.	Technology	Repository support	Repository supports refers to the use of a repository for software reuse. It also includes the search and retrieval factors and the use configuration management tools.
14.	Technology	CASE tool support	Computer Aided Software Engineering (CASE) tool support refers to the use of additional programming tools. Effective software reuse provides tools where the repository is integrated in programming tools for daily use, reducing barriers to software reuse. By further integrating CASE tools with the repository the idea of an application generator can be realized.

15.	Technology	Communication tool support	Communication tool support addresses the need to support communication through the use of tools. These tools actively facilitate communication about reusable assets. E.g. the changes in a component or the latest developments. Also, the use of discussion forums or mailing lists can be considered communication tool support. This tool support differs from the other tools because they are passive. A user can for example populate a change on repository, but the other users are not aware of it until they check out a component and note a difference. Communication tool support is a reuse factor added based on the requirements formulated earlier, the literature review did not directly indicate this as a reuse factor.

TABLE 6: IDENTIFIED REUSE FACTORS

This section presented the methodology of identifying the reuse factors. After the reuse factors were identified they were grouped around the BTOPP model and checked with the requirements formulated in the previous chapter. Due to the requirements two reuse factors are emphasized, namely 'communication channels and organizational support' and 'communication tool support'. In the next section the mapping of the requirements with the maturity levels is discussed, after which the individual reuse factors are presented in more detail.

3.4.3 Mapping reuse factors to maturity stages

Mapping the reuse factors against the defined maturity stages requires a systematic approach. It remains a difficult task, however, to determine when, where and how each reuse factor pops in. This problem is also recognized by Garcia et al. in their paper of the RiSe Maturity Model [46]. The mapping of the reuse factors to the maturity stages is based on three steps.

As the used maturity stages are compliant with the CMMI-DEV [59] model it is a logical first step to map the reuse factors against the CMMI-DEV model. The reason for this is that the CMMI-DEV model is considered a proven and generally accepted model, resulting in high quality mapping. The CMMI model has allocated Key Process Areas (KPA's) to the maturity stages. Mapping the maturity stages to the KPA's results in the overview presented in Appendix G. After evaluating the results of this mapping the conclusion is drawn that not all reuse factors can be mapped against maturity stages and that not all KPA's are relevant for software reuse. The reuse factors that can be mapped with this step are the process factors. This is logical because the CMMI-DEV model is process oriented.

The factors that could not be mapped based on the CMMI model are evaluated further. For the second step the incremental model of Griss [27] is used. This model does not directly provide a structure for the mapping, but the insights can be used to guide the mapping process. In most cases the insights of the model of Griss are complemented by additionally insights obtained from the literature survey.

Some factors could not be directly related to the model of Griss. These factors are filled in only based on the insights gained by literature. This is considered the third step.

An overview of the mapping methodology and mapping results is presented in Table 7. The mapping approach does have several limitations, which cannot be completely overcome. The quality of the mapping decreases with each step, as the mapping starts with proven methods and ends with additionally gained insights. All of the steps are, however, based on relevant literature and experts. Beside that it does not describe how a reuse factor should pop in.

Step	Description	Quality	Allocated factors
1	CMMI-DEV Key Process Areas [61].	Excellent	7 (Mostly process areas)
2	Incremental model of Griss [27].	Good	6
3	Insights gained by other relevant literature.	Average	2
		Total factors:	15

TABLE 7: MAPPING METHODOLOGY AND RESULTS.

The result of mapping the individual reuse factors to the maturity stages is presented in Appendix F. Based on this mapping the conclusion is drawn that almost all reuse factors start to play a significant role at level 2. In the CMMI-DEV level below that (level 1) no KPA's are assigned to a maturity stage [59]. It is, however, unlikely that no characteristics of software reuse factors can be found at this level. Therefore the decision is made to start all the reuse factors at level 1 and spread them out incrementally to the other maturity levels. The insights gained from the mapping methodology are used to spread the reuse factors out over the maturity levels. Although this result was not intended when setting up the mapping methodology, it is in line with existing reuse maturity models and frameworks. When looking back at existing reuse maturity models and frameworks the conclusion is drawn that they did not drawn this conclusion explicitly.

Relating the results to the CMMI-DEV model the conclusion is drawn that the staged approach is implemented differently. The CMMI-DEV model allocates KPA's to maturity stages, suggesting that several KPA's have to be addressed first before attention should be paid to others. This management tool addresses all the reuse factors at once, only the way they are addressed differs per maturity stage.

The management tool is designed in such a way that a relation is expected between the reuse factors and the maturity stages. The exact combination of reuse factors and maturity stages remains unknown, as there is a scarce amount of empirical evidence for when a reuse factor exactly pops in. The patterns for addressing reuse factors and incrementally implementing them have to be analyzed in further research. This is also one of the points of improvement mentioned in Chapter 7. For now the assumption is made that all the reuse factors are indeed addressed at all maturity levels. The way they are addressed, however, is incremental. Based on this assumption it is possible to identify strong and weak reuse factors, by comparing them to the other reuse factors.

The incremental addressing of reuse factors over the maturity stages is elaborated on by presenting the reuse practices itself, which is done in the sub-sections following this one. The insights gained through the mapping methodology discussed in this sub-section are used to address them incrementally. When discussing the reuse factors per maturity stage, the reuse practices also become clear. An overview of the characteristics of the reuse practices is presented in section 3.5. The characteristics of a reuse practice for a maturity level actually represent how they can be incrementally addressed.

3.4.4 Business factors

The business element of the BTOPP model refers to the strategy of a company [64]. For software reuse this also means that a software reuse program or system should be in line with the business strategy. The literature review for relevant software reuse factors resulted in one element related to the business strategy: the domain focus. More business factors are available, but they can hardly be influenced and are therefore excluded.

Domain focus

The domain focus is an indicator for the level of commonalities among products. The development of applications for a narrow and focussed domain will likely result in high levels of similarities among

these applications. The development of applications for a broad and unfocussed domain will likely result in low levels of similarities in these applications. Software reuse levels are directly influenced by the strategic choice of organization to focus on a certain domain.

Garcia et al. describe three ways of having a domain focus [46]: a company creates products which evolve over time, or creates an application customized for different customers, or a combination of both. Interesting enough, they also note that a domain focus does not indicate the presence of a domain engineering technique or a product line approach. While this is true for the lower levels of reuse, high levels of reuse do require that they are architected according to the model of Griss [27]. So domain engineering techniques and a product line approach can be expected at higher reuse levels, which are only possible in a narrow and focussed domain. It is also only in such a domain that investments in a product line can be justified. Furthermore, Griss mentioned that there has to be a clear need to commit to software reuse, as some employees may behave according to the Not Invented Here (NIH) syndrome [65]. This syndrome assumes that developers prefer to build their own assets instead of reusing the assets created by someone else. The domain focus define indirectly to what extend people should commit to software reuse, so the commitment to software reuse is assumed to be embodied in the domain focus.

The vast majority of literature identified the domain focus as a relevant reuse factor. An overview of the domain focus is presented in Table 8. The allocating of domain focus to the maturity levels is based on the incremental stages defined by Griss [27] and additional literature. The allocating is in line with the RiSe Maturity model of Garcia et al [46].

	Stage	*Factor description*
1. Domain focus	1	Domain focus is unknown. Either because no reuse focus has been defined, or because the organization recently entered a new domain.
	2	Multiple parties recognize reuse opportunities through sharing common functionality and code patterns. A reuse focus is initialized, but not limited to a specific domain.
	3	Reuse becomes domain specific. A moderate level of similarities among applications in the same domain allows the reuse through more specific domain focussed components and patterns.
	4	The focus is on product families with a high level of similarities among applications in the same domain.
	5	The focus is on product families with a high level of similarities among applications in the same domain. The domain focus is the same as in stage 4, only the domain is maturing more.

TABLE 8: DOMAIN FOCUS PER STAGE.

Excluded business factors

The evaluation of business factors is based on the overview presented in Appendix F. In this Appendix two other business factors have been identified, but were excluded as a reuse factor in the management tool. The first is the kind of applications which are developed. Some researchers note a difference between the development of management information systems, embedded systems and other systems. But no significant difference were found with the study of Lucredio et al. [62]. This factor was excluded as there was no strong support found for its importance. The second factor which was excluded is the application domain. Lucredio et al. [62] identified that the drug store, food & drinks and the telecommunication sectors have a slightly higher degree of reuse success. A weak link between the application domain and reuse success has been identified, this factor can however barely be influenced. The factor seems is assumed to run on the background of identifying a reuse focus.

3.4.5 Organizational factors

The organization element of the BTOPP model does not only refer to the psychical separation of organizational structure, such as business unit, but also to the division of tasks, roles and responsibilities [64]. The organization element also includes the factors top management support and the use of instrumental mechanisms.

Top management support and instrumental mechanisms

The influence of top management support is considered a crucial factor for setting up any organizational wide reuse program. The vast majority of literature identified top management support explicitly as a crucial factor for software reuse. Instrumental mechanisms can be used by top management support to guide software reuse programs. Top management support is first discussed after which relevant instrumental mechanisms are elaborated on.

The importance of top management support is illustrated by various experience reports. Joos describes software reuse at Motorola, where they became aware that top management support was required after the first phase of setting up a software reuse program failed [66]. Without top management support the task force, dedicated to software reuse, changed frequently of members and lacked the planning and foresight to become successful. Also no resources were available to adequately support the consumers of reusable software assets, as software reuse was nothing more than an additional activity. At Motorola they noted that software reuse activities require up-front investments, which cannot made without top management support [66].

Instrumental mechanisms are mechanisms which can be applied by top management to guide the software reuse program. The literature review identified several instrumental mechanisms which are: rewards and incentives, a reuse champion, examples, education and training. Although contradictive findings were discovered for the individual mechanisms, a combination of instrumental mechanisms is likely to be used for any reuse program. Instrumental mechanisms can help to take away, or at least reduce, the following issues summarized by Griss: '*people do not know how to reuse effectively, or are biased against it through a lack of trust, or 'suffer' from the Not Invented Here (NIH) syndrome, or fear a loss of creativity and independence, or a combination*, [20].

Rewards and incentives vary from intrinsic rewards like recognition, to extrinsic rewards like monetary rewards. The survey of Frakes and Fox evaluated recognition rewards and discovered no evidence that it influences software reuse [67]. The effects of monetary rewards were not evaluated in that survey. Interesting enough the open source community is successfully reusing, without having any extrinsic rewards, suggesting that intrinsic rewards do play a role [68]. Furthermore Haefliger argues that software reuse occurs naturally in open source projects, because developers are heavily constrained by available resources, such as time [68]. Morisio et al. places another side note at the use of rewards and incentives, as a complex reward policy can actually lead to the failure of a software reuse program [7]. According to Morisio et al. the use of rewards and incentives alone is not enough, but human factors should be addressed [7].

A reuse champion and the examples set by them or the management can also be used as instrumental mechanisms to stimulate software reuse. A reuse champion is often an experienced and respected employee, who is able to motivate other employees to actively participate in a software reuse program. The examples serve the same purpose as the reuse champion, they are both meant to stimulate employees to actively participate in a software reuse program.

Education and training are also mentioned in literature as a success factor. Top management support needs to be in place before any form of education and training can take place, as resources will have to be made available first. The level of education contributes to the people factor in the form of relevant experience, either in the form of domain knowledge or in the form of reuse knowledge. Both Joos and Banker note education as an important factor for teaching employees best practices regarding software reuse [66, 69]. Joos also uses it to create understanding and commitment among the employees. Fafchamps notes education for teaching employees knowledge about the functionality provided in a repository and how to find it [70]. The provision of resources for education and training are considered as a part of this reuse factor. The result of providing these resources are embodied in the two people factors, elaborated on in 3.4.7.

The influence of top management varies over software reuse levels. While there is no support required for ad-hoc reuse stages, a high level of support is required for software reuse factors or product line

approaches. The allocation of top management support issues is based on the model of Griss [27] and insights gained by additional literature discussed in this section.

An overview of the suggested levels of top management support and related instrumental mechanisms is presented in Table 9. The model of Garcia et al. [46] does not include top management support, the model of Koltun and Hudson [48] does, but in an indirect way. One of the reuse factors in that model defines the reuse motivation and culture, which describes how top management influences reuse. The term 'indoctrination of reuse' was populated from that model into the fourth maturity stage. In previous reuse maturity models and framework the instrumental mechanisms are not discussed as such, rather they are considered as individual reuse factors e.g. in [46, 48]. The literature survey to the reuse factors provided conflicting results for these individual reuse factors alone, the combination of them with top management support seems to provide better results. The combination is, however, not explicitly confirmed by literature.

2. Top management support and instrumental mechanisms	*Stage*	*Factor description*
	1	Top management is not aware of reuse opportunities or does not support it.
	2	Top management recognizes and supports reuse activities. Reuse champions and examples are used to encourage reuse. Rewards and incentivizes are changed to stimulate reuse activities.
	3	Top management assigns dedicated resources to software reuse. Resources are allocated to and focussed on areas where reuse potential is perceived as high.
	4	Top management is enforcing reuse activities. Not reusing is regarded as poor employee performance.
	5	Total management commitment to software reuse. Management is striving for organizational excellence through reuse activities.

TABLE 9: TOP MANAGEMENT SUPPORT AND INSTRUMENTAL MECHANISMS PER STAGE.

Organizational reuse structure and identification of reuse roles

The organizational structure of software reuse is at its most abstract level a division between a supplier of software reusable assets on one side and a producer of software reusable assets on the other side [70]. This division can be made both within organizations as among organizations. The organization of software reuse among organization manifests itself through so called component markets elaborated on in e.g. [71-72]. One organization is offering components, where another one is buying it. Although some examples of black-box component markets are present these days, they are still in their infancy [73]. Several limitations need to be overcome before the full benefits of component markets can be obtained, including setting up quality standards and accepted certificates. The reuse factor called supplier management take the possible use of components markets into account.

The internal organization of software reuse is often embodied in a separate team, dedicated to creating and maintaining reusable software assets. The vast majority of literature recognizes the need to set up a separate team as a way of dedicating and allocating resources to software reuse. Allocating dedicated resources to software reuse is required, as software reuse won't occur by itself. The main reason for this is that individual projects are normally focussed and judged by the outcomes of the project itself and not by the reusable assets created.

A good team structure is one that minimizes the costs associated with software reuse and maximises the benefits. Unfortunately, only a few efforts have investigated the most appropriate team structure for a given situation. The paper of Fafchamps provides guidance regarding the team structure dedicated to software reuse [70]. Fafchamps appears to use the library approach regarding software reuse. The paper of Bosch describes similar team structures, but then from the perspective of a software product line [71]. Morisio et al. conclude that different team structures can be used as long as they are adapted to its environment [7], which actually provides no guidance at all.

Fafchamps did a case study at Hewlett-Packard (HP), one of the earliest starters of reuse programs, where 10 HP sites were studied for their reuse programs [70]. Based on her research she abstracted four models of reuse organization: *'lone producer'*, '*nested producer', 'pool producer'* and '*team producer'*. With a lone-producer model a single individual handles several product teams' reuse needs. With a nested-producer model each product teams has a team member dedicated to providing reuse services and expertise. The nested-producers have to communicate their efforts among each other. With a pool-producer two or more teams collaborate to both produce and share reusable components. Finally with the team-producer model a separate team is assigned to provide reusable software components. This team is on the same organizational level as the software developing teams. Fafchamps found that the team producer is working best at HP. Lone and pool producers seem to have potential, while nested producers doesn't work well at HP due to the multiple responsibilities and communication lines.

In the article of Poulin the results of Fafchamps are compared with a case study of IBM done by Brauer [74]. Equal to the team-producer structure, Brauer presents a similar successful organizational structure defined as the '*reusable parts centre*' [74]. The paper of Poulin, however, fails to remark that both cases are done in rather large companies with highly division structures and official reporting channels [70]. In such organizations a centralized reuse program indeed seems to be the most fit.

The models defined by Bosch describe several organizational structures for a software product line [71]. The organizational structures defined by Bosch are: '*development department*', '*business units*', '*domain engineering unit*' and '*hierarchical domain engineering units*' [71]. With the development organization, all resources for developing reusable assets are located in a single department. With the organization of business units, each business unit is responsible for a subset of reusable assets. With domain engineering units, an organizational distinction is made between producers of reusable assets and consumers. The domain engineering is responsible for producing reusable assets, where the application engineering units are responsible for consuming these reusable assets. The hierarchical domain engineering is a hierarchical division of the domain engineering unit, where domain engineering units may be controlled by other domain engineering units. Bosch notes that the domain engineering unit is the one most often discussed in literature, however, that the suitability of the various models depends on the given situation [71]. The factors that influence the organizational model for a particular situation include geographical distribution, project management maturity, organizational culture and all the type of systems [71].

The models of Fafchamps and Bosch cannot be directly compared. First of all there is the difference between a library approach and a factory approach. And secondly Fafchamps tends to focus on the roles of members and their relations within and among teams, while Bosch is focusing on purely organizational divisions [70-71]. Despite these differences there are also complementary results. The advantages and disadvantages of each of these models are presented in Appendix E.

Apart from the basic reuse roles of producer, consumer and top management, several other roles have been identified in literature. Prieto-Diaz mentions a reusability committee, maintenance groups and a qualification group [75]. Morisio et al. add the reuse program manager, asset owner and library manager to this [76]. Lastly, Isoda mentions the reusability committee [77]. The various reuse roles will be briefly evaluated. For the framework it does not matter whether dedicates roles are part-time or full-time, the thing that matters is that they have been addressed.

- A *producer* is responsible for creating reusable assets. In most cases it also includes the maintenance of the reusable assets, but not necessary.
- A *code owner* is responsible for managing reusable assets and guiding its development in the right direction. The code owner is often an experienced developer who also created the original component. So, the component owner is familiar with details regarding the piece of code.
- The *maintenance team* is a group dedicated to the maintenance of components. The maintenance team separates the roles of producing and maintaining reusable assets.

- The *qualification group* is proposed as a separate group responsible for the quality of reusable assets in a repository. Furthermore they collect, produce and certify new additions to the repository. They function as a board that determines which producer items are populated into a database and which are not. The *library managers* and the *reusability committee* describe a similar task.
- The role of the *reuse manager* is defined as the head of the reuse group. The reuse manager is responsible for guiding the reuse programs in the right direction.

Rosenbaum describes a more practical approach by defining 'owners' and 'partners'[78]. Each component is managed by a code owner, assisted by partners who are users of the component in their product. The responsibilities of the owners and partners are the same, but they report to different individuals. Partners report to their project or product groups (consumer teams), where the code owner reports to the producer team.

The organizational structure and reuse roles are likely to change when shifting through various levels of software reuse. The suitability of related organizational models depends both on the maturity stage as on the given context. An overview of the team structure and related reuse roles per stage is presented in Table 10. The division of the team structure and the reuse roles are guided by the incremental staged of Griss [27] and insights gained by literature as elaborated on in this section.

The team structure is also taken as an element in the model of Garcia et al. and the model of Koltun and Hudson [46, 48]. Both models describe reuse teams and roles evolving from a shared initiative to a corporate group. Interestingly, it is unlikely that high levels of reuse can be achieved through the use of a separated dedicated group, as this group will face problems regarding relevant domain knowledge [79]. An overview of the proposed team structure and reuse roles per maturity stage is presented in Table 10.

	Stage	*Factor description*
3. Team structure and responsibilities	1	No reuse structure or roles have been identified.
	2	Component owners are assigned to guide the evolution of general components. Components owners meet each other and discuss potential reuse opportunities. The role of component owners may be given to different individuals over time. Mostly expert developers are used as component owners as they are trusted for their decisions.
	3	A maintenance group of component owners is set up together with a group of partners. Partners are representatives of cross-linked projects working with domain specific components.
	4	The maintenance group is not only guarding the components, but also the architecture in which the components have to fit.
	5	Apart from the maintenance group guarding the basic architecture and the related set of components, additional roles may be defined for the development of extension to the basic product.

TABLE 10: TEAM STRUCTURE PER STAGE.

Communication channels and organizational support

For software reuse in general and agile development methodologies in particular, communication channels and related organizational support have to be addressed. The support has been made explicit to organizational support, as communication tool support has also been identified as a software reuse factor. The scaling of agile development methodologies is embedded in both reuse factors. Software reuse literature itself also identified the addressing of communication channels as an important factor. Both viewpoints are elaborated on below.

Agile methodologies in more complex settings emphasize the importance of communication channels and support. Where direct face-to-face communication is effective in normal settings, additional communication channels and support is required in more complex setting. The addressing of

communication channels and related control issues is one of the scaling factors for agile methodologies mentioned in Chapter 2. It is not specified how the scaling factors should be addressed, rather that they should be adjusted to the context of the individual organization.

In software reuse literature the experience reports of Isoda [77] and Fafchamps [70] recognize the need to address communication issues. These issues include the '*reporting structure*' and facilitation of '*feedback to change requests*'. Frakes and Fox mention the communication problem as an important factor for the failing of software reuse programs [45]. Haefliger describes how mailing list are used in the open source community, to obtain knowledge regarding specific components [68].

The addressing of communication channels is mainly based on insights created upon existing literature. The filling is compliant with the CMMI maturity stages [59]. The model of Griss [27] could not be used for this reuse factor. An overview of communication channels and support is presented in Table 11.

In previous reuse maturity models the communication channels and organization support was not taken as a reuse factor explicitly. With the potential use of agile methodologies the decision was made to include it. This also meets the requirement formulated earlier that the model should address relevant communication and control issues.

	Stage	*Factor description*
4. Communication channels and organizational support	1	Communication management issues are not addressed.
	2	The communication channels are defined, but not used systematically. Informal and irregular meetings provide the basis for reuse development.
	3	Communication channels are supported and used systematically. Regular meetings are held and its results are spread out through the organization.
	4	The use of communication channels is enforced by top management. Reuse issues are visible for all relevant parties.
	5	The use of communication channels is optimized and tailored to the needs of the organization.

TABLE 11: COMMUNICATION CHANNELS AND ORGANIZATIONAL SUPPORT PER FACTOR

Excluded organizational factors

The comparison with the identified reuse factors, as presented in Appendix F, leads to the conclusion that no organizational factors have been excluded. The three reuse factors describing the organizational aspects of software reuse cover the results of the literature survey. The merging of top management support with the instrumental mechanisms provides a logical extension of existing literature.

3.4.6 Process factors

The process element of the BTOPP model contains several relevant process factors. The literature review identified the systematic software reuse process as a key element for software reuse. The composition of such a systematic software reuse process is first discussed, after that the individual process factors are elaborated on. Systematic software reuse is defined as a software reuse process which abandons ad-hoc reuse and at the same time exceeds the project level [7]. An overview of the software reuse process is first provided, after which the relevant process factors are discussed.

Software reuse process

A systematic software reuse process is defined as a software reuse process which abandons ad-hoc reuse and at the same time exceeds the individual project level [7]. The software reuse process can be described according to the model of Caldiera and Basili [79], which is presented in Figure 14. The model describes the interaction between the consumers of reusable assets on the left side and the

producers of reusable assets on the right side. Most organization do not recognize the reuse processes this explicitly, as reuse is considered as common as everyday life. This is also why reuse is often put into practice through more informal and unsystematic reuse processes [1, 79]. By making the distinction explicit, a common basis for understanding software reuse processes is provided.

The model of Caldiera and Basili defines three process areas consisting of the '*project organization*', the synchronous process of the '*experience factory*' and the asynchronous process of the experience factory referred to as '*component factory*' [79]. The purpose of the project organization is to deliver applications to the customers [80]. By doing so they are interacting with the experience factory for reusable components in order to develop those applications. The experience factory is responsible for monitoring and analyzing project developments, developing and packaging experience for reuse in the form of knowledge, processes, tools and products, and supplying the packaged experience to the project organization upon request [80]. So, the experience factory is not considered as a passive entity only delivering services on request, but rather an active entity, looking around for opportunities to deliver reuse services in order to meet potential future request [79].

With this model the process of creating reusable assets is assigned to the experience factory. The experience factory has to know what the projects are working on and needs to continuously gather relevant knowledge. A key factor is that the domain knowledge of the experience factory and the component factory are up to date and in line with the latest development within the various project organizations. A separation of the experience factory and the component factory are claimed to be necessary, as reuse during a normal development process is at its best a secondary concern [79].

Morisio et al. emphasize that the adjustment of existing development processes and the introduction of reuse processes is crucial for successful software reuse programs [7]. Morisio et al. also noted that successful reuse programs incrementally implement the process adjustments by minimizing change each stage. The same is noted by Mili et al. by stating that building with reusable components does not necessary differ radically from building without reusable components [19].

For the reader interested in the human activities related to the processes we refer to the article of Maiden and Sutcliffe [26]. The human activities can be mapped against the model of Caldiera and Basili [79].

The literature review for software reuse factors resulted in several process activities arranged in six process factors. The process factors are mapped against and numbered in Figure 14. The process factors are elaborated on below the figure according to their assigned number.

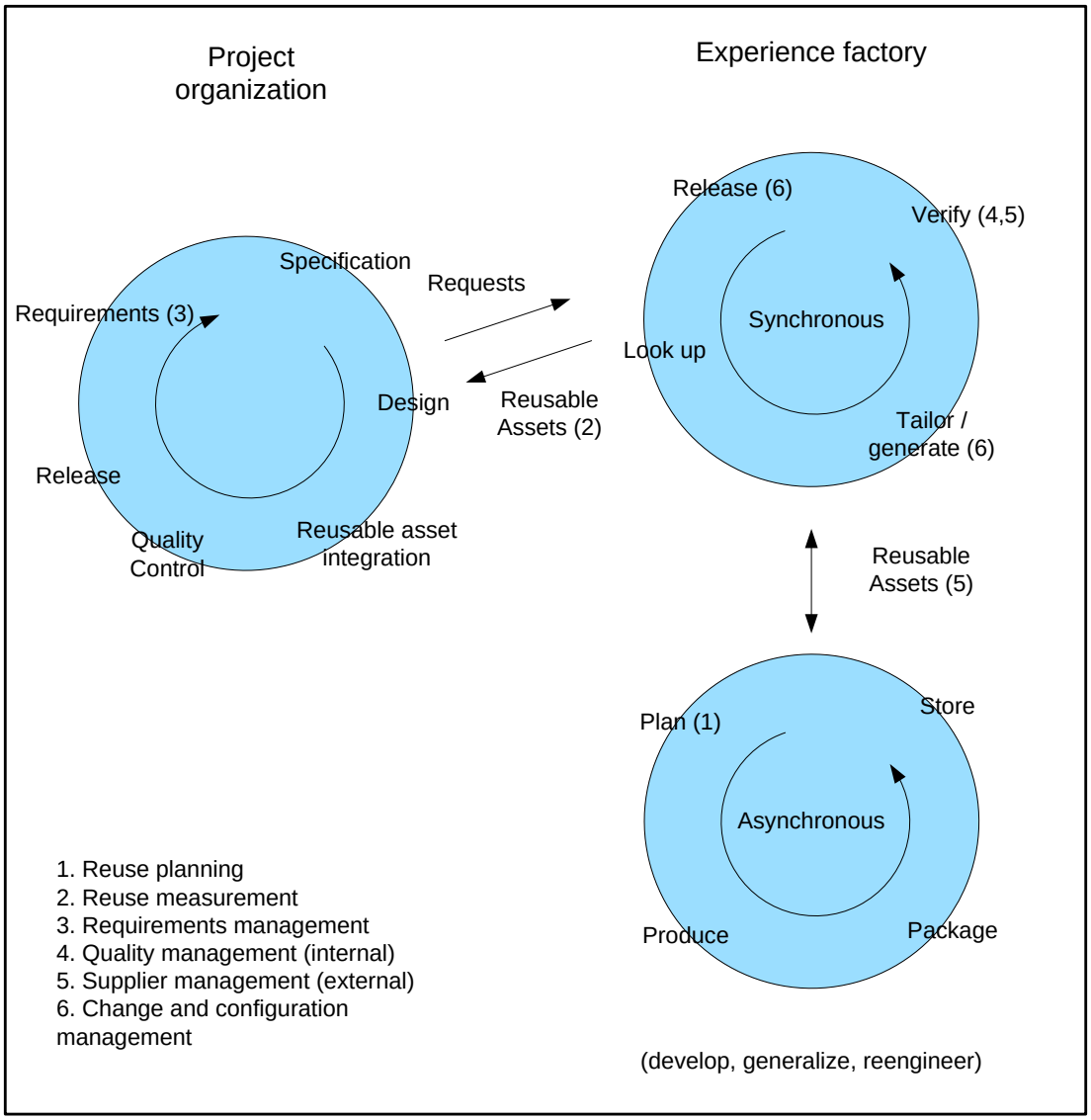


FIGURE 10: THE REUSE PROCESS MODEL ADAPTED FROM CALDIERA AND BASILI [79].

Planning for reuse (1)

Planning is the process of setting goals, developing strategies and outlining tasks and schedules to accomplish these goals. Before the goals can be set an indication of reuse opportunities is required. A systematic reuse approach makes use of domain analysis, which scans the relevant domain for possible reuse opportunities.

Isoda presents results where reuse remained at a steady rate of 20%, stressing the need for domain analysis to achieve higher levels of reuse [77]. Morisio et al. agree with the statement that domain analysis should occur, but, they also found successful projects where no domain analysis was executed. The data set of Morisio et al. was over based on a set of organizations just starting with software reuse, indicating that domain analysis may become more important later on. Rine describes domain analysis as a method for reducing risks [81]. By carefully analyzing the domain and selecting reuse opportunities risks can be reduced and the existing stock of preferred parts can be leveraged. Executing domain analysis requires a more planned approach than is possible with ad-hoc reuse [77].

Beside the use of domain analysis techniques for reuse planning, also the scope and the use of the plan are factors of influence. A plan can be based on long term goals, short term goals or medium long term

goals. The reuse plan can be used to quantitatively manage reuse activities by comparing planned results versus actual results.

All the aspects are combined in the overview presented in Table 12. The CMMI model is used as the main guideline for allocating the reuse factor to the maturity stages. From stage 3 and up domain analysis is included. Stage 4 is based on quantitatively managed activities and uses the reuse plan to measure the actual outcomes versus the expected outcomes. The final stage optimizes the use of reuse planning.

The results are in line with the article of Poulin [82] and indirectly with the article of Morisio et al. [7].

5. Reuse planning	*Stage*	*Factor description*
	1	No reuse plan is present.
	2	Reuse events are planned, but based on the perceptions of relevant experienced developers.
	3	Reuse events are planned and supported through the use of domain analysis.
	4	Reuse events are planned, supported through the use of domain analysis and used to quantify the actual outcomes versus predicted outcomes.
	5	Reuse plans are frequently created and optimized to maximise reuse results for a given domain.

TABLE 12: REUSE PLANNING PER STAGE

Reuse measurement and cost justification (2)

Reuse measurement includes all the activities related to the measurement of reuse events. When reuse events are measured they can be presented by reuse metrics and combined with costs models to provide the basis for cost benefits analysis of software reuse.

A rough estimation is that setting up a reusable component requires 200% of the normal efforts. Beside the additional costs of creating a reusable component, there are costs such as component maintenance costs and the costs of integrating the reusable component into new applications. Mili et al. define a cost model for software reuse taking these factors and others into account [8]. Obviously a component has to be reused more than two times before it is profitable to make a component of it in terms of direct costs and benefits. On the long term other benefits can be gained such as shorter time to market. The literature review to reuse measurement and related cost justification resulted in some contradictive findings, which are evaluated in the paragraphs below.

Morisio et al. did not identify the use of reuse metrics as necessary for successful reuse projects [7]. Rather the organizations evaluated in their research relied on the intuitive perceived benefits of software reuse. The organizations in the research of Morisio et al. were possible in the initial stages of software reuse. Morisio et al. noted for example a shift in reusable assets from code based reuse to reusable components, similar to stages of this thesis [7]. Beside that Morisio et al. argue that software reuse is likely a factor of influence [7]. The survey of Frakes and Fox did not identify reuse metrics as crucial for software reuse, but at the same time they did place a side note that reuse measurement was probably just not used by the surveyed organizations [67]. The experience reports of Isoda [77] and Joos [66] did confirm the use of reuse metrics as important. Rine justifies the importance of reuse measurement by stating that management demands insights in the payoffs of a reuse program [81]. This means that the management demands that the investments in software reuse have to be justified by the benefits gained by software reuse. Reuse measurement and costs models have to be installed to provide the numbers needed for cost and benefit analysis. However, one has to bear in mind that setting up a reuse measurements metrics and installing it, requires additional investments. Therefore it is unlikely that initial reuse levels have any form reuse measurement, but at higher and more systematic levels of reuse, reuse measurement can be expected.

The allocation of reuse measurement to the maturity stages is based on these findings and the guidelines provided by the CMMI-DEV model. An overview is presented in Table 13.

6. Reuse measurement and cost justification	*Stage*	*Factor description*
	1	No reuse measurement is present. Benefits are justified for the individual through reduction in personal (re)work and time savings.
	2	Reuse initiatives can be tracked manually when needed. No reuse metrics are installed and reuse benefits are often based on intuitive perceptions.
	3	Reuse metrics are installed and reuse events are transparent. The actual costs of reuse activities still have to be analyzed manually and are often estimated.
	4	Reuse metrics are installed and cost models are present to justify investments. In this stage all reuse activities are measured and quantified.
	5	Reuse investments can be estimated and related investments are optimized.

TABLE 13: REUSE MEASUREMENT AND COST JUSTIFICAITON PER STAGE

Requirements management (3)

Requirements management consists of a wide range of activities related to identifying, obtaining, controlling and maintaining requirements [59]. The CMMI-DEV model distinguishes requirements at three levels, namely at customer level, product level and component level. All requirements levels can be used as a basis for software reuse, but the latter one is of course the most interesting.

Requirements management was identified in the literature review as a software reuse factor embedded in a systematic software reuse process. The model of Caldiera and Basilli also has requirements management as one of the software reuse activities [79]. Furthermore the Krueger refers to requirements management issues for optimizing software reuse [18]. In Krueger's view, the ideal situation is that an application generator is capable of generating a working solution based on an incomplete set of requirements. While not much more was found through the literature review it is clear that requirements management can provide a valuable basis for software reuse. Therefore this factor is confirmed as an important factor for software reuse.

Existing reuse literature does not address requirements management for software reuse explicitly. The allocating of software reuse to the maturity stages is based on the insights gained by the CMMI-DEV model. Also Important also the three levels on which requirements management can take place. The result of allocating requirements management to the maturity stages is presented in Table 14.

7. Requirements management	*Stage*	*Factor description*
	1	Requirements are managed at customer level. There is no requirements management for individual reusable assets.
	2	Requirements are documented and managed at product level. It however remains a difficult task to relate them to components.
	3	Requirements are documented and managed at product and component level and the results are visible organization wide.
	4	Requirements are documented and managed at product and component level. Frequently appearing requirements are analyzed for possible reuse opportunities.
	5	Requirements are exploited based on similarities. This can provide the basis for an application generator.

TABLE 14: REQUIREMENTS MANAGEMENT PER STAGE

Quality management (4)

Quality management addresses the quality of reusable assets. When the quality of reusable assets is low it is not expected that they are reused at all. Quality management includes the use of quality models, certificates and ranking systems. Quality management can be both about internally created reusable assets as about external created reusable assets. In addition to the quality management of external reusable assets, supplier management issues should be addressed. Supplier management is elaborated on after this reuse factor.

The application of quality models and ranking systems differ in where they pop in at the reuse process. The quality model can be used as criteria *before* reusable assets are populated into a repository, where a ranking system measures the popularity of reusable assets *after* they are populated into a repository. When reusable assets are popular and frequently used, the assets are likely to be of good quality. The other way around, when a component is not popular and thus not frequently used, it is not necessarily of bad quality.

The importance of measuring the quality of components before populating them into a repository is emphasized by both Rosenbaum and Rine. Rosenbaum describes quality criteria as mandatory for a successful reuse program [78]. The view of Rosenbaum on reusable assets is rather static, without good quality components there won't be used any reuse at all. Rosenbaum judges the quality of components based on a 10 point scale assigned to attributes of reusable assets. Rine emphasizes the importance of only populating quality components into a repository by stating that the word for repository would be better replaced with '*junkyard*' as it would result in failure of the reuse program [81].

Haefliger notes that open source projects use popularity, measured by downloads of reusable components, as a benchmark for related quality [68]. Haefliger argues that is the same as the use of quality ratings and certificates [68]. Haefliger did not identify or at least does not mention limitations of determining the quality after the reusable assets are produced. In the open source community it seems that the consumers of reusable assets are well capable of determining the quality and the usefulness of reusable assets themselves. In additional the opinions of others are use though, through the use of mailing lists. It can, however, be expected that the use of a quality model becomes more important when the development of reusable assets is based on a fixed architecture.

Frakes and Fox did not identify the quality of components as direct inhibitor of reuse activities [67]. Reuse success can also be achieved without the use of quality models and ranking systems. Frakes and fox do place a side note at the findings of their survey, stating that the use of certificates for contracting may play a more dominant role in the future, especially with external contracting.

The allocation of quality management is guided by the CMMI-DEV model. In contrast with the other reuse factors, quality management is assigned to the third stage of the CMMI-DEV model and not the second [59]. The allocation of more formal quality management is therefore also assigned to the third stage. An overview of the allocation of quality management to the maturity levels is presented in Table 15.

	Stage	*Factor description*
8. Internal quality management	1	No quality management models or ranking systems are present. The perceived quality is based on previous experience.
	2	Quality management is not formally addressed, at its best there are ranking systems present which measure the quality afterwards.
	3	Quality management issues are discussed through regular meetings and frequent interaction. Quality models are likely not present yet.
	4	Quality models are present which determine if a components is of high quality and if its fits within the architecture.
	5	Quality models are present which determine if a components is of high quality and if its fits within the architecture. Quality models and components are frequently evaluated and optimized.

TABLE 15: INTERNAL QUALITY MANAGEMENT PER STAGE

Supplier management (5)

Supplier management has the purpose to manage the acquisition of external assets from a supplier [59]. Supplier management is likely to become important when reusable assets are bought from external black-box components markets. With white-box components the internal logic of the

component can be viewed and adjusted, a consumer is this case not completely dependent on the supplier.

The CMMI-DEV model defines a similar process area named ‘supplier agreement management’[59]. Similar to that process area supplier management issues include contract management, guarding of quality of delivered products (e.g. use of certificates), contractual agreements and the transition of acquired assets to the projects [59].

The literature review did identify the use of external assets as a potential further possibility. For example in the survey of Frakes and Fox [67]. However the identification of a reuse factor in the form of supplier management was not mentioned in any of the papers in the literature review. The model of Caldiera and Basilli indirectly recognizes supplier management, by drawing the arrow between the component factory and the experience factory [79]. With this model there is again little known about supplier management. This can likely be explained due to the infancy of existing black-box components markets in general [72].

In order to make sure that the management tool addresses all aspects of software reuse, supplier management was also taken as a reuse factor. Although supplier management is likely to be not very relevant at the moment, it can play a more dominant role in the future. An overview of the allocation of supplier management issues per maturity stage is presented in Table 15. After stage 3 it is unknown how supplier management will behave and it is therefore not included in the proposed management tool. The reason for this is that no empirical evidence nor logical lines of reasoning can be provided for these two last stages. Additional research is required to fill in these two stages.

	Stage	*Factor description*
9. Suppler agreement	1	Supplier management issues are addressed based on the experience of individuals.
	2	Supplier management is not present, but multiple external suppliers may be used. E.g. through the acquisition of white-box components from open source projects or public repositories.
	3	Supplier management issues play a role when making use of black-box component markets.
	4	Not applicable yet.
	5	Not applicable yet.

TABLE 16: INTERNAL QUALITY MANAGEMENT PER STAGE

Configuration and change management (6)

Configuration and change management consists of a set of practices for establishing and maintaining the integrity of work products [59]. The CMMI-DEV model mentions the use configuration identification, configuration control, configuration status accounting and configuration audits [59]. Although the practices defined by the CMMI-DEV model use the term configuration, they also address the change aspects. Hence the name configuration and change management.

Configuration and change management can be divided into three areas. These are change management, configuration management and release management. All three deal with changes at different levels. Change management deals with the management of change requests prior to change itself. After the change is accepted and executed a change request such a change request is closed. The configuration management level deals with the physical change requests, it keeps track of previous states, i.e. the state before a change requests is executed and current states, i.e. the state after the change requests. Release management is about releasing a set of changes into a stable version. Release management is strongly recognized by the library approach as defined by Griss [27].

Configuration and change management issues are limited addressed in literature. The vast majority of papers assume components to be carefully tested before they are populated into a repository. The change management issues are often addressed through assigning dedicated roles to reuse maintenance (e.g. [75]), or by installing a version revision management system (e.g. [7]).

Change management is closely related to communication channels and support. Fafchamps explicitly notes the need for cross-linked negotiation as a necessity, before any change-requests can be executed [70]. Isoda reports situations where individuals are uncooperative and responding slow to maintenance needs [77]. Fafchamps describes various recommendations when describing her four organizational models, including: '*specify processes for change requests*' and '*ensure rapid feedback to change request*' [70].

The allocation of change and configuration management to the reuse maturity levels is based on the CMMI-DEV model and insights gained by literature. In the initial stages it is likely that the configuration management issues are based on project level, where in latter stages they are based on product or component level, similar to what was noted at requirements management. An overview of the allocation of configuration and change management to the maturity levels is presented in Table 17.

10. Configuration and change management	*Stage*	*Factor description*
	1	Configuration and change management issues are addressed at project level. At its best an individual use tools to keep track of change and configuration issues at component level.
	2	An existing configuration management tool is used to support configuration management issues. Change management issues are also addressed, but informally.
	3	Change management is organized more systematically and becomes addressed pro-actively. A configuration management tool is used in combination with a change management tool. Changes are registered and can be manually related to revisions.
	4	Change and configuration management is enforced at product and component level. Changes can always be related to revision.
	5	Change and configuration management support is integrated and optimized.

TABLE 17: CONFIGURATION MANAGEMENT PER STAGE

Excluded process factors

When looking back at the identified reuse factors presented in Appendix F, no process factors are excluded, but two process factors are placed in the background. The first factor placed in the background is the kind of reusable asset used. Agile methodologies focus on producing code as fast as possible, so the focus is likely on producing code components and not necessary on producing additional documentation. Documentation is, however, not meant to be disregarded. A requirements document or a functional design is still valuable and probably necessary to achieve higher reuse levels. It was however not mentioned so explicitly. The other identified factor that is placed in the background is the origin of reusable assets. This factor is strongly embedded in the reuse planning factor. When a reuse event is carefully planned its origin is likely to be new. When reuse events are unplanned it almost has to be based on existing pieces of code, requiring possibly substantial amounts of rework to be able to reuse them. Furthermore requirements management and supplier management are extracted from a systematic reuse process and added as reuse factors. Both provide possibilities for optimizing reuse opportunities.

3.4.7 People factors

The people element of the BTOPP model describes the factors relevant for individuals when participating in software reuse. The literature review resulted in two factors, namely 'producer experience and skills' and 'consumer experience and skills'. Education, as an instrumental mechanism, can be used to influence both factors, but is not taken as a people factor itself.

Producer experiences and skills

The producer is responsible for the production of reusable assets. Producer experiences and related skills are often not mentioned explicitly as a reuse factor in software maturity models or frameworks. Still the vast majority of evaluated articles refer to the experience and skills of a reusable asset producer as a success factor for software reuse.

Fafchamps noted that a trusted and experienced developer with effective communication skills would be best as a producer of reusable assets [70]. A developer is considered experienced when he or she has at least 5 years of relevant work experience. Assigning an experienced and skilled developer contributes in two ways. First of all, the experience and skills of the developer likely lead to good decision making. The quality of components is therefore expected to be higher than when someone with little or no experience is assigned to the production of the same component. Secondly the decisions made by an experienced developer are likely to be accepted more readily. An experienced developer is trusted to make the right decision. Based on those two arguments Fafchamps argues that a code owner of reusable assets should be an experienced developer [70].

Rothenberger noted that the experiences of developers in a narrow domain can lead to increased reliability of the components itself [60]. Furthermore developers with more experience are likely to recognize reuse patterns sooner [83]. As such more reuse opportunities may be seized by experienced developers.

Morisio et al. discovered that those projects that were successful are indeed assigned to experienced developers. The study of Morisio et al. did, however, barely contain projects where producer roles were assigned to non-experienced developers [7]. As such no direct conclusion can be based on that research.

The allocation of producer experience and skills to the maturity stages cannot be done solely based on the CMMI-DEV model, the model of Griss [27] or the insights gained by literature. To operationalize this reuse factor a different variant of the CMMI-DEV model is used, namely the people capability maturity model (P-CMM) [84]. The people capability maturity model addresses the skill and experience development of individuals from an organizational perspective. The result is presented in Table 18.

	Stage	*Factor description*
11. Producer experience	1	Producer uses present skills and knowledge. Components are produced in an ad-hoc manner, relying heavy on previous experience.
	2	The producer gets a recognized producer role. Based on this roles repeatable processes and practices are developed.
	3	Core competencies for developing reusable assets are identified leading to best practices.
	4	Producers are mentored to develop their skills. Weak skills are identified and improved, where good skills are reinforced.
	5	Continuous personal competency development.

TABLE 18: PRODUCER EXPERIENCE PER STAGE

Consumer experience and skills

Consumers are those who are using existing reusable assets in the process of building new applications. Consumers experience is also referred to as reuse experience in literature. Reuse experience is relevant experience of the domain in which reuse events are taking place. Reuse experience also includes knowledge of available components [85]. The skills embody the knowledge and techniques to needed to use existing components in new applications.

The experience and skills of consumers are subject of debate since the beginning of software reuse. Several papers mention that consumer experience and skills are success factors for software reuse, while others note that they are actually inhibiting software reuse events. The arguments of the various papers are presented below.

According to Banker et al. 85% of the reuse events occur through internal reuse and only 15% of the reuse events occur through external reuse [85]. This research identifies relevant domain knowledge and experience as a strong indicator for potential reuse events. Rothenberger also noted that

developers should have a good understanding of the reuse model, the contents of the repository and the capabilities of the reusable components [83].

Lee did a survey within the ADA community and found contradictive findings [86]. ADA is a modular language strongly promoting reuse events. In the survey by Lee the reuse experience negatively influenced software development. An experienced programmer may prefer to develop components from scratch as the perceived quality of existing components is too low. In that case the perceived benefits for an experienced programmer are considered lower than the perceived benefits for a novice programmer. This statement provides arguments for noting that novice users may be more eager to participate in reuse events than experienced developers.

The survey of Frakes and Fox also indicated that software reuse does not increase with more reuse experience [67]. Based on the research of Morisio et al. a similar conclusion can be drawn, as all the successful reuse projects started with no or limited reuse experience [7].

The findings presented by Banker et al. [69] can be used to explain the two perspectives. In their case study project managers noted that it took inexperienced programmers only two months to become a productive consumer, and that after about six months the learning curve flattened out [69]. Based on that the conclusion is drawn that consumer experience is a relevant reuse factor, but also that it does not take many efforts to become an experienced consumer. Too much consumer experience and relatively low quality of components are expected to inhibit reuse events. In this case the consumer should become a producer and improve the component.

The allocation of consumer experience and skills to the maturity stages cannot be done solely based on the CMMI-DEV model, the model of Griss [27] or the insights gained by literature. Similar to the producer skills and experience this reuse factor is operationalized through the use of the people capability maturity model (P-CMM) [84]. The result is presented in Table 19.

12. Consumer experience	*Stage*	*Factor description*
	1	Consumer uses present skills and knowledge. Components are reused in an ad-hoc manner, relying heavily on previous experience.
	2	Consumer is aware of the basis of software reuse and has a decent understanding of the available reusable assets.
	3	Core competencies for integrating reusable assets are identified leading to the use of best practices.
	4	Consumers are mentored to develop their skills. Weak skills are identified and improved, where good skills are reinforced.
	5	Continuously improving individual competencies and innovating best practices.

TABLE 19: CONSUMER EXPERIENCE PER STAGE

3.4.8 Technology factors

The technology element of the BTOPP model is used to describe how technology can support a reuse program. The literature review indicates the use of repositories and CASE (Computer Aided Software Engineering) tools. Elaborating on the need to manage changes and providing support for communication, a third factor is identified and named communication tool support.

Repository support

The repository is the database system, with possible extended functionality, used to store the components and share them with other members of the organization. The repository is also referred to as the knowledge library, the reuse library and the component library [20]. The use of a repository in the form of a knowledge library is not identified as a critical factor for software reuse (e.g. [7, 62, 67]), its presence does, however, significantly influence software reuse (e.g. [52, 75]).

Morisio et al. identified that companies usually do not make large investments for setting up a reuse library, but instead prefer to use their own configuration management tool [7]. The study also proved

that decent levels of reuse can be achieved with such a configuration management tool. The repository itself does provide options for further optimizing its use, ultimately through the use of an application generator [18]. In the latter case a repository is transformed from a passive entity, only capable of storing components, to an active entity capable of generating applications based on the stored components. Such an application generator requires substantial investments and is expected to take away a part of the flexibility of developing applications. Only a few organizations operating in a narrow domain can benefit from such a system

Together with repository support a lot of research is dedicated to search and retrieval techniques for software reuse [85]. Where the earliest papers invested much effort in optimizing the possibilities, the latter papers try to keep it as simple as possible [7]. The search and retrieval techniques are considered as a possible focus point of repository support, but are not further elaborated on. In line with this relevant reuse experience and domain knowledge is considered more important than search and retrieval techniques. This is indirectly confirmed by Banker et al. [85].

The allocation of repository support is based on the CMMI-DEV model, the insights gained by the model of Griss [27] and the literature discussed in this section. As Morisio et al. mentioned often an existing configuration management tool is used for the repository, therefore a close relation can be found between 'configuration and change management' and 'repository support'. Similar to configuration management a distinction is made between to the levels of deployment. This can be within a project, but also outside a project. The results are presented in Table 20.

13. Repository support	*Stage*	*Factor description*
	1	The components have to be derived from the repository of existing projects. At its best individuals have separated repositories and personalized tools.
	2	A central repository is available with basic functionality and limited support. Morisio et al. noted that often existing configuration management tools are used [7].
	3	A knowledge library is set up and the configuration management tool is placed to the background. The knowledge library contains more stable versions of components and has more advanced functionality.
	4	The reuse repository is extended to meet the demands imposed by a common architecture. Categorization and possible other additional functionally is provided.
	5	Optimized repository support, integrated with other tools.

TABLE 20: REPOSTITORY SUPPORT PER STAGE

CASE tool support

Computer Aided Software Engineering (CASE) tool support is a way of enhancing software engineering through the use of dedicated tools. In literature the use of CASE tool support is approached from the perspective of the consumer, with the goal to make the task of integrating reusable components into a new application easier.

Without CASE tool support a programmer has to exit the programming environment, use a limited set of tools to search for possible reusable components and then has to re-enter the design environment [82]. The additional effort required to search and integrate components is mentioned by Poulin as a factor hindering software reuse [82]. The integration of CASE tool support with the repository of components can help reducing such barriers.

Furthermore CASE tool support can facilitate the integration of a library centred approach [75]. The same is recognized by Banker et al., who note that it is important to ingrate tool support within the development life cycle [69]. CASE tool support has the potential to speed up the implementation of new systems and leverage modules across multiple projects and business areas within the organization [87].

Although CASE tool support is recognized by researchers and practitioners as a way of leveraging software reuse, it is not identified as a critical factor for software reuse. The survey of Frakes and Fox

found no positive results regarding the use of CASE tools [67]. Also the work of Haefliger regarding the open source community did not report the use of CASE tools as a critical factor for software reuse. The open source community did use other ways to leverage reuse across multiple projects, elaborated on in the reuse factor 'communication tool support'.

The allocation of the factor across the maturity stages is based on the CMMI-DEV model, the insights gained by the model of Griss [27] and the literature discussed in this section. For this reuse factor the various levels of implementation are also addressed, varying from project specific CASE tools to software reuse CASE tools. Also its integration with the repository is spread out over the maturity stages. The integration of the CASE tool with the repository becomes important when a common architecture is defined. This occurs in stage 4 and onward. The results are presented in Table 21

14. CASE tool support	*Stage*	*Factor description*
	1	Project specific CASE tools are used. Individuals may have their personalized extensions which integrate the components into the tool.
	2	CASE tool support is provided for software reuse and integrated with high potential areas for software reuse.
	3	CASE tool supported is provided for all reusable assets. All developers can select the components from their CASE tool.
	4	Integrated CASE tools provide the building blocks for reusable assets. This may results in the deployment of a flexible architecture through the use of CASE tools.
	5	The usage of CASE tools is optimized. This may lead to the application generators as purposed by Griss [65].

TABLE 21: CASE TOOL SUPPORT PER STAGE

Communication tool support

Communication tool support actively facilitates the communication among producers and consumers. The use communication tool support is identified as a potential way to further operationalize 'communication channels and organization support'. The factor is explicitly included as this it is also identified as a possible way to scale agile methodologies for more complex environments. Previously mentioned tool support facilitates communication in a passive way. E.g. when a new component is placed on the repository it is up to the user to check the repository to find out that it is placed. Communication tool support can be used to inform users of such issues. Communication tool support can also be used by consumers to ask for relevant information.

Haefliger mentions the use of mailing lists to facilitate communication in open-source projects [68]. In the investigated research setting a user asked a question through the use of mailing lists and get forwarded to the potential person who has more information. Beside that the users get informed of ongoing events by the mailing. A similar use of mailing lists is informing users of changes. Instead of just populating a change in a repository it can actively be communicated through mailing lists or other communication tools. A bug tracking system is another example of such a communication tool. The bug tracking system contains information about change requests possible related to components. Through the use of a bug tracking system a user can be notified when a change to a component is requested.

The allocation of communication tool support is based on the characteristics of the CMMI-DEV stages. The allocation is further operationalized through the use of the discussed literature. An overview of the results is presented Table 22. The use of communication tool support is not addressed in previous reuse maturity models or frameworks.

15. Communication tool support	*Stage*	*Factor description*
	1	No communication tool support is provided for software reuse. Individuals communication software reuse issues face-to-face or through other informal ways.
	2	Initial communication support is provided, but not used systematically. This includes the use of regular email, mailing lists and discussion forums.
	3	Communication tool support is provided and used systematically. This can include the use of change management systems and other tools. Reuse communication is still separated from the normal communication chains.
	4	Communication is supported through the use of dedicated tools, which allows both communication within projects and among projects. The tools are enforced by top management and the use of it is evaluated and measured. Consumers and producers can mine the communication issues top obtain relevant knowledge.
	5	Communication support optimized and integrated in development tools.

TABLE 22: COMMUNICATION TOOL SUPPORT PER STAGE

Excluded technology factors

Evaluating the results found by the literature and presented in Appendix F the conclusion is drawn that no technology factors are excluded. The search and retrieval techniques are placed to the background and integrated with the repository support, which is in line with the findings of Morisio et al. [7], stating that the simplest approach is often the best. The reader may find that there are other technology factors which have been excluded, namely those that are related to the component itself. As stated at the beginning of this thesis the focus is not on the technology needed to create components, but rather on the management of these components. The technology factors discussed in this thesis are the technology factor supporting the management of the components.

3.5 Reuse practices

The implementation of a reuse factor on a certain maturity stage is put into practice through the use of reuse practice. A reuse practice is therefore a combination of a reuse factor and a maturity stage. The reuse practices form the basis for the assessment of the management tool. The attentive reader has already discovered several reuse practices in the previous sections. The purpose of this section is not to discuss the individual reuse practices, rather its purpose is to present the characteristics of the reuse practices of the various reuse factors and maturity stages.

The combination of the CMMI-DEV model [59], the incremental levels presented by Griss [27] and the insights gained by literature led to a common set of characteristics. Again, the exact mapping has already been discussed in the previous section. The common set of characteristics is presented per maturity level. An overview is provided in Table 23 and elaborated on below the table. The result is quite similar to the discussion of the maturity stages as presented in section 3.3, although from a different perspective and with a different purpose.

Characteristics of maturity stages					
	Stage 1	*Stage 2*	*Stage 3*	*Stage 4*	*Stage 5*
CMMI-DEV model [59]	Ad-hoc, undefined	Initial, reactive	Pro-active	Measured, quantified	Optimizing
Incremental levels presented by Griss [27]	Not available	Leveraged code components	Work products managed by teams	Architected.	Domain specific optimizing
Insights gained by literature.	No general support, individualized, code scavenging	Some support is provided, no systematic reuse	More systematic	Enforced	Optimizing, total commitment, organization wide, exploiting

TABLE 23: OVERVIEW OF CHARACTERISTICS PER MATURITY STAGE

The practices at the *first maturity stage* are characterized by the terms 'ad-hoc', 'individual', ‘code scavenging’ and 'no general support'. In this stage reuse events are happening ad-hoc, which often means that developers are reusing knowledge of previous projects to save personal time and work load. The benefits of software reuse are perceived and utilized by the individual. Code scavenging means that code is copy-pasted and adjusted to the needs of the individual.

Characteristics of the practices at the *second maturity stage* are 'initial', ‘reactive’, ‘leveraged code components’, 'some support is provided’ and ‘no systematic reuse’. The characteristics present a maturity stage where reuse events are happing and the organization has become aware of it. The basics are present to manage software reuse. The management of software reuse is, however, rather reactive. Management issues are evaluated after they have happened.

In the *third maturity stage* the following characteristics of software reuse practices are present ‘pro-active’, ‘work products managed by teams’ and ‘more systematic’. The major difference with the second stage is that reuse issues are managed more pro-actively. Reuse opportunities are evaluated and the processes are more systematic.

The practices at the *fourth maturity stage* are characterized by terms as ‘measured’, ‘quantified’, ‘architected’ and ‘enforced’. In some case it makes sense to indeed quantify and measure the outcomes of implementing a reuse factor, while other reuse factors have to be adjusted to the demands imposed by a common architecture. In some cases the implementation of a reuse factors also has to be enforced by top management in order to reach higher reuse levels.

In the *fifth maturity stage* the characteristics of reuse practice are ‘optimizing’, ‘domain specific’, ‘total commitment’, ‘organization wide’ and ‘exploiting’. In order to optimize the reuse factors a narrow and specified domain is required. The organization has to show total commitment to software reuse and software reuse is also implemented organization wide. Several or all reuse factors are exploited.

An overview of the reuse practices can be found in the assessment tables presented in Appendix M. In the assessment table the reuse practices are divided over three categories ‘approach’, ‘deployment’ and ‘results’. This is further explained in the next section. .

3.6 Reuse factor scoring

This section provides the reader insights into the basis of the assessment method component. The focus of this section is on the scoring of reuse factors. In the next section an additional assessment method variable, the relevance variable, is discussed. Before elaborating on the scoring of reuse factors two assessment methods are compared and evaluated. After that the used methodology for scoring the reuse factors is presented.

3.6.1 Evaluated assessment methodologies

The two assessment methodologies which are evaluated are the SCAMPI method [88] and the matrix based assessment used in the article of Niazi et al [57]. The SCAMPI method is evaluated because it is used for the assessment of CMMI-DEV[88]. The matrix based approach is evaluated as it was suggested for the SPI framework, used for defining the structure of the proposed management tool.

SCAMPI

The Standard CMMI Appraisal Method for Process Improvement (SCAMPI) is the official SEI assessment method for providing benchmarks [88]. The SCAMPI method is also used for the assessment of the CMMI-DEV model, which is used as the basis for the proposed management tool. The SCAMPI method is used for identifying strengths and weaknesses of current processes, reveal development/acquisition risks, and determine the capability and maturity level ratings.

The SCAMPI method recognizes three phases of assessment. In the first phase a plan is set up and the assessment is prepared. In the second phase the plan is carried out and in third phase the results are reported. The SCAMPI method is documented extensively [89], but the documentation is described in natural languages and for its real applicability additional information is required. For the application of the assessment method additional training or the use of a SCAMPI appraisal expert is required.

Furthermore the SCAMPI method can be applied according to three classes of appraisal. These are the classes A,B and C. Each of the classes meets the Appraisal Requirement for CMMI (ARC), but they differ in their degree of tailoring and the depth of investigation. An overview of the characteristics of the ARC classes is presented in Table 24.

Characteristics	SCAMPI Classes		
	Class A	*Class B*	*Class C*
Amount of objective evidence gathered	High	Medium	Low
Rating generated	Yes	No	No
Resource needs	High	Medium	Low
Team size	Large (4 or more)	Medium (2 or more)	Small (1 or more)
Organizational unit coverage	Required	Not required	Not required
Data sources (instruments, interviews, and documents)	Requires all three data sources	Requires only two data sources (one must be interviews)	Requires only one data source
Appraisal team leader requirement	Authorized Lead Appraiser	Authorized Lead Appraiser or person trained and experienced	Person trained and experienced

TABLE 24: CHARACTERISTICS OF ARC CLASSES DERIVED FROM [90].

Matrix based approach

The paper describing the SPI framework structure of Niazi et al [57] refers to work of Daskalantonakis [58]. Daskalantonakis proposes a matrix based approach for assessing CMM levels. The matrix based approach defines three assessment dimensions, which are: '*approach*', '*deployment*' and '*results*'. The assessment dimensions are used to evaluate the state of a factor according to the maturity levels. The three dimensions have to be evaluated simultaneously leading to a single outcome. So the three

dimensions do not have to be scored individually, but just provide different perspectives on the maturity state of a factor.

3.6.2 Advantages and disadvantages

In this sub-section the two assessment methods are compared based on advantages and disadvantages. An overview is presented in Table 25 and elaborated on below the table.

Approach	Description	Advantage(s)	Disadvantage(s)
1	SCAMPI	Widely used by the CMM community. Can be tailored to the individual organization. Vast quantities of available documentation.	Requires relatively large amounts of resources required. Results are compared against the CMMI model, organizations are not compared directly. Difficult to use, experts or training is required. Documentation does not provide all the insights.
2	Matrix based approach	Repeatable for multiple organizations due to fixed scoring and ranking. Requires few resources and is easy to use.	Cannot be tailored to the individual organization. Depth of investigation is relatively low.

TABLE 25: COMPARISON SCAMPI AND MATRIX BASED APPROACH

The SCAMPI method has the major advantage that is standardized and that there are vast quantities of documentation available. The documentation is also a disadvantage, as it does not provide enough information to really apply the method. Experts or additional training is required to apply the method. The SCAMPI method can be tailored to the organizations and different depths of investigation can be maintained.

The matrix based approach does not have different levels of investigation and cannot be tailored to the individual organization. It is however easy to apply and requires relatively few resources. The results are repeatable for multiple organizations.

3.6.3 Used assessment approach

The used assessment approach, for scoring the reuse factors according to their maturity, is the matrix based approach. The main reason for this is the thought that every investment made in software reuse has to be gained back through these reuse activities. A quick assessment can provide valuable insights in potential improvement areas through the identification of strengths and weaknesses. Investing large amounts of resources for assessing the reuse factors cannot be expected to be paid back. These investments are better made in developing real improvements.

The scoring of the reuse factors is thus doing according to the three dimensions ‘approach’, ‘deployment’ and ‘results’. The combination of these dimensions lead to a fixed score. The scores are mapped against the maturity levels, where each score is equal to the maturity level multiplied with a factor of 2. An overview of the mapping of the score to the maturity levels is presented in Table 26.

Score mapping						
Maturity level	0	1	2	3	4	5
Score	0	2	4	6	8	10

TABLE 26: MAPPING SCORES TO THE MATURITY LEVELS

Daskalantonakis [58] uses the even scores to indicate a maturity level. The odd scores are used to indicate that certain characterises of reuse practices are met, while some are not met yet. The use of odd scores can be applied when a given situation cannot be placed in an assessment dimension combination.

An overview of the matrix based approach with the fixed scores can be found in Appendix M. The result is a set of assessment tables, which can be scored according to the maturity based approach.

3.7 Reuse factor relevance

This section explains the relevance variable and its applications. The relevance variable is added to the assessment method to overcome the limitations of not being able to tailor the assessment method to the individual organization. Furthermore the relevance variable is used as an indicator of the need to scale a certain reuse factor.

To keep it simple the relevance variable ranges from 1-5 and is described as a combination of three possible scores. The scores are referred to as weights in this section, to avoid confusion with the previous section. An overview of the relevance variable and its weights is presented in Table 27.

Weight	Importance	Description
1	Low	The reuse factor is of little or no influence and is not worth improving it.
3	Normal	The reuse factor is important for this organization, but not a critical focus area.
5	Critical	The reuse factor is very important for this organization and should be improved if possible.

TABLE 27: MAPPING RELEVANCE TO IMPORTANCE

The description of the relevance variable is related both the importance of a reuse factor for the organization under assessment and the need to scale the individual reuse factor.

The use of a relevance variable obviously favours the continuous approach of the CMMI-DEV model [59]. By adding a relevance variable to the reuse factor the organization determines which reuse factor is more important for them to scale. Using the relevance variable in multiple case studies in similar research settings it is expected that a pattern can be discovered. This pattern can then be used to identify the importance of the reuse factors at a certain maturity stage. The filling in of fixed relevance variables is out of scope for this research, but does provide opportunities for using the staged approach of the CMMI-DEV model.

The relevance variable is also necessary to meet the requirements set up in advance. The requirement referred to states that the management tool should provide options to make it specific for agile situation. The use of the relevance variable provides this option as it gives the individual organization a way to indicate the importance of a reuse factor. This individual organization can be an agile organization, but also a normal organization. The relevance variable does not limit the use of the management tool to agile organizations only, meeting another requirement. This requirement states that the management tool should not be limited to agile organizations alone, as they have proven to be capable of being scaled up to more complex environments.

The assessment method consists of the matrix based approach for the scoring of reuse factors and use of the relevance variable. This relevance variable can also be found in the assessment tables presented in Appendix M. The next section puts all pieces of the management tool together by explaining how the management tool should be applied.

3.8 Using the management tool

This section puts all the elements of the management tool together by describing how the management tool should be used and how the results should be interpreted. The first sub-section describes the application procedure of the management tool. The next section elaborates on the identification of the strong and weak points resulting in potential improvement areas. And the final section elaborates on the interpretation of the results.

3.8.1 Application procedure

The application procedure of the management tool can be divided in three steps. The three steps are the same three steps as recognized by the application of the SCAMPI method [88], discussed in section 3.6.1. The recognized are preparation, application and reporting results.

Preparation

In the preparation phase the application of the management tool is planned. The relevant stakeholders are determined and gathered. The group of relevant stakeholders differs based on the amount of involved and available individuals. It is considered a good practice to use multiple representatives of several roles and business units in the organization. In this phase it is also recommended to select a neutral person to guide the application of the assessment and gathering its results. It is important that the neutral person has the knowledge to explain the assessment tables as presented in Appendix M. The neutral person can be an external consultant or someone in the organization, who is not directly influenced by reuse opportunities.

Application

The neutral person explain the relevant stakeholders the goal of the management tool and the application procedure. After that they receive the tables as presented in Appendix M. The relevant stakeholders are asked to add a score and a relevance to the reuse factors. The neutral person can be consulted in case of questions or additional comments. At the end of the application the neutral person discusses the results with the relevant stakeholder. To prevent a chaotic whole it is recommended that the neutral person performs the application one on one with the stakeholder. The stakeholder is then not directly influenced by the other stakeholders.

Presenting results

The neutral person gathers the results and combines these. The results are presented to the stakeholders as a group, leading to potential further discussion. The results can be used to identify a set of improvement points, which can then be implemented into the organization.

3.8.2 Identifying strong and weak points

The assessment method component of the management tool adds a score and a relevance to the reuse factors. Both have to be used to identify strong and weak points, leading to potential improvement areas for an organization. The identification depends on the context of the individual organization, but a set of common rules is provided to help these organizations. The following characteristics of assessment scores and assessment relevance can be used for the identification process:

- A strong factor is a reuse factor with a high score
- A weak factor is a reuse factor with a low score
- An improvement factor is a relatively weak factor with a relatively high relevance.
- A factor with a relatively low score and a relatively low relevance is likely no point of improvement at the time being, this may change in the future.

- The factors that have a relatively average score and an average relevance are factor in the ‘grey area’ and do not require immediate attention. Possible they are already at the desired level of the organization.

Summarizing the characteristics it is important to note that the identification of strong and weak points is based on a combination of a relevance and a score. The two variables can, however, not be multiplied. A strong point with a low relevance is not the same as a weak point with a high relevance leading to the same multiplied score. Furthermore the exact identification of strong and weak points depends on the individual context, which is the reason why the characteristics are defined using a ‘relatively’ low and ‘relatively’ high scores and relevance.

The help the identification of the strong and weak points the results can be visualized in two ways. The first is by presenting the assessment results in a graph and the second is by presenting them in a Kiviat diagram. Figure 11 presents example results presented in the form of a graph and Figure 12 presents example results in the form of a Kivit diagram. Additional information is provided below the figures.

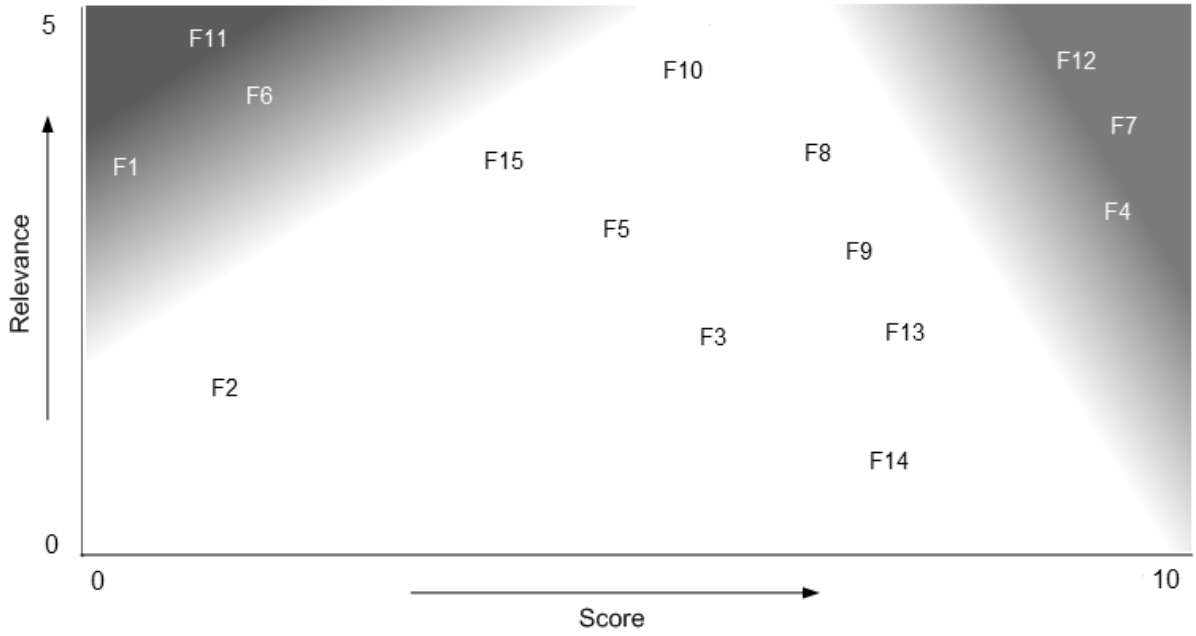


FIGURE 11: IDENTIFCATION OF STRONG AND WEAK POINTS

Figure 11 presents a graph with example results. On the horizontal axis the factor score is presented and on the vertical axis the factor relevance is presented. The fifteen reuse factors are presented by the numbers F1-F15. Using the characteristics presented earlier in this section the factors F12, F7 and F4 are identified as strong factors. They have a relative high score. Weak factors in this figure are the factors with a relative low score. These are F1,F11, F2 and F6. F2 is, however, not a point for improvement, as it also has a low relevance score. The factors which can be identified as improvement factors are: F1, F11 and F6. The improvement factors and the strong points are presented in the greyed out areas, respectively on the left side and the right side of the figure. The darkness of the colour can be used as an indicator for the strength or the weakness of the factor.

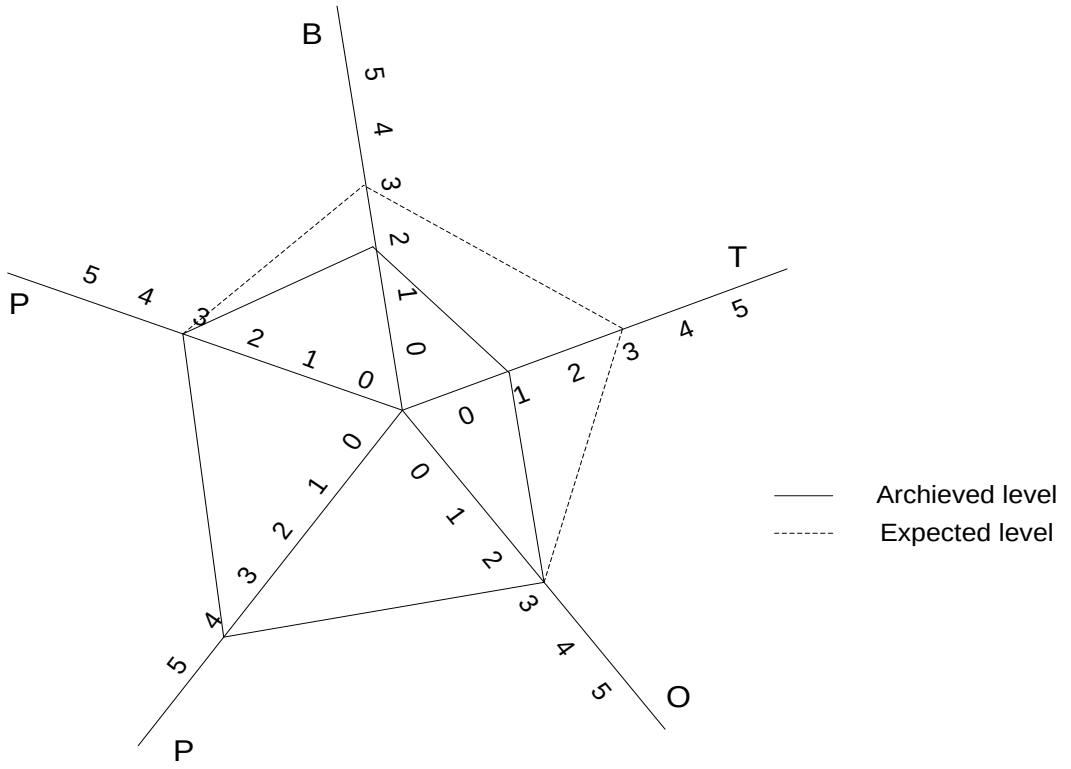


FIGURE 12: EXAMPLE KIVIAT DIAGRAM

Figure 12 presents an example Kiviat diagram based on the work of Daskalantonakis [58]. The diagram presented in Figure 12 makes use of the organization and categorization provided by the BTOPP model. The BTOPP elements are presented on the axis, indicating potential improvement areas. Instead of the BTOPP elements also the individual reuse factors themselves can be presented. The way the Kiviat diagram is presented in Figure 12 favours the staged approach of the CMMI-DEV model [59]. The relevance of a reuse factor can be added to figure itself though, through the use of additional numbers or the use of colours. A darker colour can for example indicator a more relevant reuse factor than a brighter colour. The identification of strong and weak points again depends on the context of the individual organization.

3.8.3 Interpretation of results

Inspired by the CMMI-DEV model [59] there are two interpretations possible. The first is the staged representation and the second is the continuous representation. Both are discussed in this section.

Staged interpretation

The staged representation of the CMMI-DEV model assumes that all reuse factors have to be scaled to a certain maturity level [59]. The proposed management tool works slightly different at this point. In sub-section 3.4.3 the reuse factors are mapped against the staged representation, after which the conclusion was drawn that it is better to spread the reuse factors across all the maturity stages. Similar to the CMMI-DEV model it can be expected that there is a relation between the reuse factors in the way and depth they are addressed. For a starting organization it can be expected that the business and organizational factors are first addressed before processes and tools are installed. The relevance variable can be used to gather information about potential patterns for addressing the software reuse factors. When linking reuse patterns to a maturity stage, the relevance of a pattern can be used to determine whether a factors should be addressed and to what extend it should be addressed. The identification of reuse patterns requires further research and is out of scope for this research.

Continuous interpretation

The continuous approach of the CMMI-DEV model [59] is more applicable for the management tool at the moment. The relevance variable can be used to identify potential strong and weak points as discussed in the previous section. Furthermore there is little known about the reuse patterns at the moment, it is unclear which factors should be addressed first and to what extent they exactly should be addressed. By using the continuous approach an organization can determine for itself in which order and to which depth the reuse factors are addressed.

4. Validation approach

This chapter describes what aspects of the management tool are validated and how they are validated. In the first section the validation model is discussed. The validation model describes what aspects of the management tool are measured to consider it as a valid and successful tool. After that the validation method is presented, which describe how the validation model is put into practice. Lastly, the expert selection and the case study are discussed. The results of the validation approach are presented in the next chapter.

4.1 Validation model

The validation model describes what aspects of the management tool are measured in order to consider it a successful model. Similar to the work of Lagerström et al. [91] three important questions have to be answered to consider the management tool as a valid tool. These questions are:

- Are there any attributes missing that should be added?
- Are there superfluous attributes that could be removed?
- Does the management tool work in practice?

In order to answer these questions a validation model is set up. The validation model describes three validity factors, which are measured and described by several attributes. An overview of the validation model is presented in Figure 13 and elaborated on below the figure.

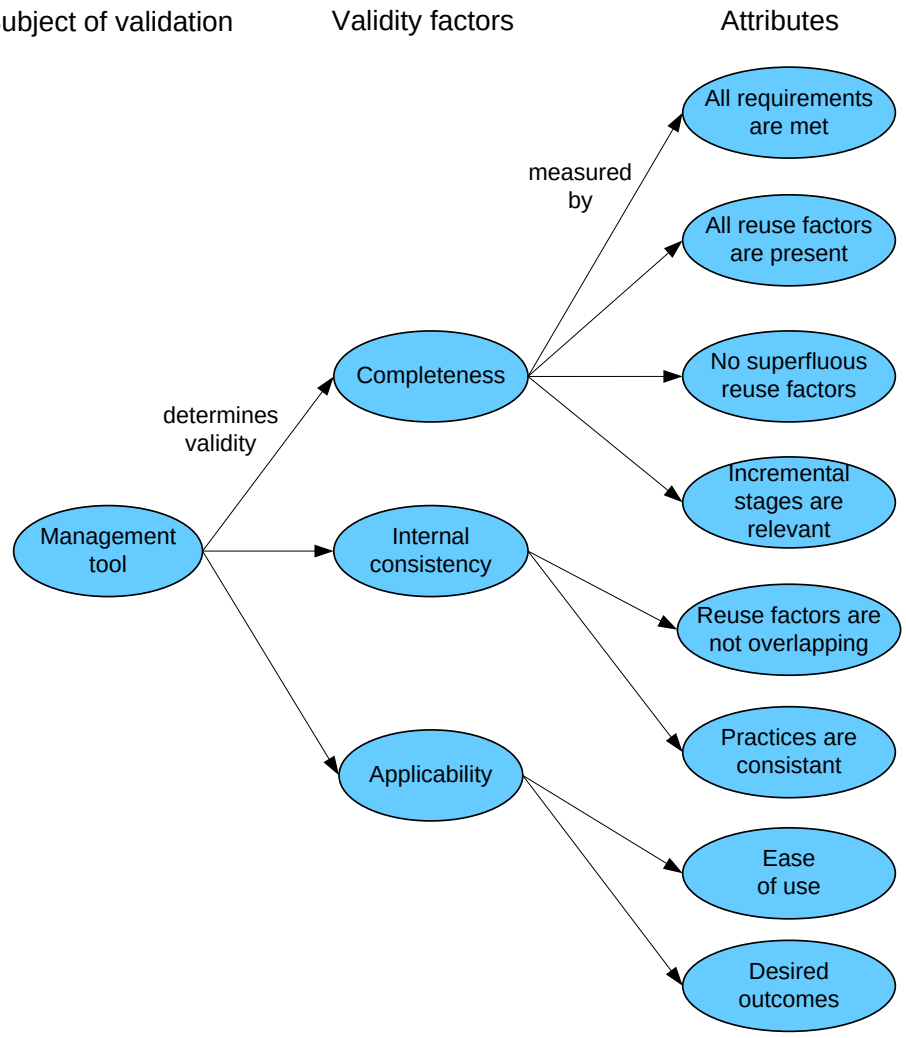


FIGURE 13: VALIDATION MODEL

In Figure 13 three validity factors are presented for the validation of the management tool, these are ‘completeness’, ‘consistency’ and ‘applicability’.

The completeness factor is used to check whether all the elements of the management tool are considered as complete set and to whether this set is representative for software reuse. This means that both the reuse factors and the maturity stages have to be complete and representative. The reuse factors are evaluated for their relevance and whether they are superfluous or not. Beside that the set of reuse factors is evaluated to determine if reuse factors have to be added. The maturity stages are also evaluated in order to determine if they are relevant for software reuse. The practices are not evaluated within this validity factor, as they are the logical result of a reuse factor and maturity level combination. The completeness validity factor addresses the first two questions of Lagerström et al. [91] concerning the attributes of the management tool.

The consistency factor is used to check if the management tool is internally valid. This validity factor does not check whether all elements are complete, rather it checks if all the elements present in the framework are behaving as intended to. The causal relationships among the various elements in the management tool should therefore be consistent. The management tool is considered internally consistent when the reuse factors are not overlapping and when the reuse practices provide a good way to assess the maturity of a reuse factor. When reuse factors are overlapping the decision should be made to merge them together or to make the differences between them more explicit. Beside that a reuse factor can be too complex to assess, in that case a reuse factor should be simplified or split up. The complexity of a reuse factor is not directly taken into the validation model, but will appear from the consistency of the reuse practices. The consistency of the reuse practices can only be evaluated through the application of the management tool. During the application process the reuse practices are used to determine the maturity of a reuse factor. When the users of the management tool have trouble identifying the state of a reuse factor, it can be used an indicator for the fact that the reuse practices may not be well defined. Beside that a large standard deviation among the results can, depending on the context of the application, be used as an indicator for a difficult to assess reuse factor. When the reuse factors are not overlapping and the reuse practices can successfully be used to identify the maturity state of a reuse factor, the management tool is considered to be internally valid. If the model is not internally valid it is unlikely that the management tool can be successfully applied in practice, which related to the third question of Lagerström et al. [91].

The applicability factor is used to check if the management tool can be successfully applied. For this research it is interesting to check if it can be applied in an agile development environment, nevertheless the tool should also be applicable in another environment. In order to say something about the applicability of the management tool, the ease of use and the outcomes of its application have to be evaluated. If the management tool is difficult to use then it is unlikely that it will be applied at all and if it is applied in such a state it will likely not lead to the desired results. The desired result is that the management tool is able to address software reuse issues and that it is able to identify strong and weak points leading to potential improvement areas. This validity factor also related to the third question of Lagerström et al. [91].

The three validity factors and related attributes are evaluated through the use of several validation steps. These are elaborated on in the next section.

4.2 Validation method

This section elaborates on the validation method, which is use to check the validity of the management tool. Where the previous section defined several points on which the management tool is evaluated, this section defines the steps which are taken in order to validate the management tool based on those points. First of all the identified validation steps are presented. Secondly the selected validations steps for this research are discussed. Lastly a mapping between the validations steps and the validation model is presented.

4.2.1 Identified validation steps

The validation can be done based on different research methods [92], leading to different validation steps. A research method is a way to do the data collection. Some research methods are more suitable than others, which is elaborated on below. After that the identified validation steps, based on a suitable research method, are presented.

In order to validate the management tool it is important that the tool is applied in practice, to determine its applicability. Therefore a field setting is considered more suitable than a laboratory setting. A field setting provides a natural setting in which the applicability of the management tool can be evaluated. A laboratory setting will not provide an answer to this part of the validation model, as it is more suitable for checking the validity of a theoretical model based on causal relations between dependent and independent variables. Within a field setting several research methods are possible. The use of a survey is not considered suitable, as it remains unclear whether the user of the management tool has understood the tool and if he or she has applied it as was intended to. In order to check whether the user has understood the management tool it is necessary to observe the application process and to engage in interaction to let him or her explain relevant actions. It is therefore important that the researcher has an active role. The research method that comes closest to the desired approach is action research. With action research a field setting is maintained where the researcher has an active role during the research process.

With action research selected as the most suitable research method, several validation steps have been identified. The validation steps take the validation model into account. An overview of the validation steps is presented in Figure 14.

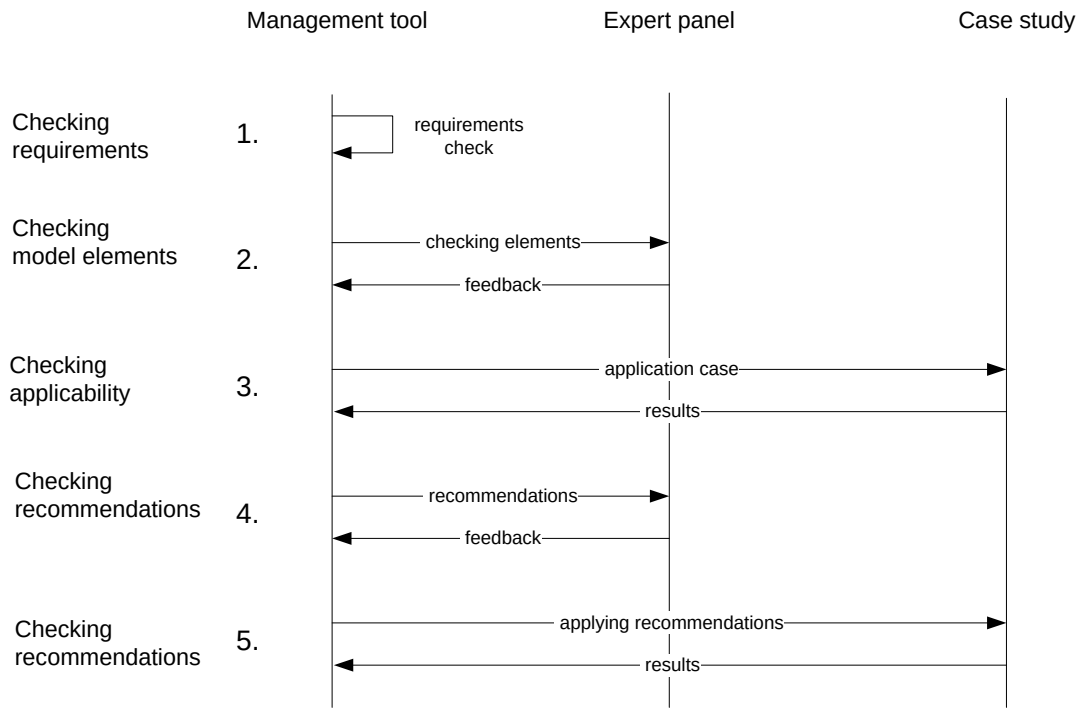


FIGURE 14: VALIDATION STEPS

Figure 14 describes five validation steps presented in a chronologic order. During the first step the management tool is validated based on requirements. These requirements are the requirements defined earlier in Chapter 2 and should be included in the management tool as described in Chapter 3. The evaluation of the requirements provides input for the completeness and thus the validity of the management tool. The second step consists of checking the management tool with an expert panel. The expert panel is asked to judge the tool based on its elements such as the reuse factors and the

defined maturity stages. The experts can also be asked to estimate the applicability of the management tool. In the third validation step the management tool is applied to a relevant case. The users applying the management tool can be observed and the results can be used to determine if the management tool is applicable or not. The group of users can also be asked to evaluate the tool afterwards. The results obtained from the application process are referred to as the case study. The case study is thus the result of the application process combined with the insights gained from the group of users and possible other knowledge sources such as other documents. The insights gained from the evaluation by the expert panel and the application of the management tool can be used for recommendations. These improvements can be about the management tool, but can also be about the improvement of the evaluated situation. This means that a group of relevant users can also be asked to judge the recommendations based on the assessment method. If the recommendations are valid recommendations it is an indicator for a working management tool. If the assessment results do not influence or negatively influence software reuse it can mean that the management tool is unsuccessful.

4.2.2 Used validation steps

In the ideal situation the validation of the management tool is performed in the order of the steps presented in Figure 14. Due to time and other resource constraints the decision is made to focus on the first three validation steps. Furthermore the assumption is made that the users of the management tool first have to apply the tool, before they are able to validate it. The selected experts of the expert panel are not only individuals who have experience with software reuse, but also have to be representative individuals for a case study. By combining the second and the third step validation step the recommendations of the experts are not used to improve the model before it is applied to the case study. This has been identified as a potential risk factor, as large mistakes can automatically lead to an unsuccessful application of the management tool. This risk factor is partially overcome by the active role of the researcher as mentioned earlier. A passive role of the researcher is expected to lead to failure of the validation process. Details regarding the validation procedure can be found in Appendix J. The documents used for the validation can be found in Appendix K. The next sections relate the validation steps to the validation model.

4.2.3 Mapping the validation steps against the validation model

The previous sub-sections indirectly claim that the validation steps can be used to put the validation method into practice. This sub-section describes how the validation steps are related to the validation model. An overview is presented in Table 28.

Mapping validation steps with the validation model			
Validity factors	*Main attribute*	*Validation step*	*Measurement method*
Completeness	All requirements are met	1	Personal judgment
	All reuse factors are present	3	Expert opinion
	No superfluous reuse factors	2 and 3	Observation and expert opinion
	Incremental stages are relevant	3	Expert opinion

Consistency	Practices are consistent	2 and 3	Observation and expert opinion
	Reuse factors are not overlapping	3	Expert opinion
Applicability	Ease of use	2 and 3	Observation and expert opinion
	Desired outcomes	2 and 3	Expert opinion and application results

TABLE 28: MAPPING VALIDATION STEPS WITH THE VALIDATION MODEL

The completeness of the model is evaluated through the use of the expert panel and the requirements check. Prior to the evaluation by the experts, the experts are asked to apply the model. As such the results of the application process are also present indirectly. The consistency of the model is evaluated through the observations made at the application process and the opinion of experts. When an expert is not able to place a situation in an assessment table it can be an indicator that the practices in that assessment table are not well defined. Furthermore when there is a high standard deviation of the application process results it can be an indicator of not well defined practices. Together with the evaluation of the completeness the experts are asked to give an opinion about the reuse factors. This is also used as an indicator for describing the overlap between reuse factors. The applicability of the model is evaluated through observations made when the experts applied the management tool and their opinion afterwards. Beside that the top 3 of identified improvement factors is evaluated against the insights of the experts.

4.3 Expert selection and case setting

The experts are industrial experts and not scientific experts. The experts are selected based on relevant experience with software reuse, their business unit and their job description.

First of all it is important that the expert have experience with software reuse. They have to be employees within the case setting, actively working with software reuse and have thought about the subject. Employees who are not working with software reuse or have never thought about the possibilities cannot be considered experts. Employees are considered experienced when they have 3-5 years of relevant work experience.

Beside that it is important that the selected group of experts is representative for the case setting. As explained in chapter one several business units are involved with the use of code components. This set of business units is based on a natural separation between programming languages and the core business activities of the business units. The business units that do not have software development as their core activity or that do not use the same programming languages as the selected business units are therefore excluded from the research.

Lastly not only does the group of experts have to represents the business units, but they also have to represent the natural population of those individuals working with software reuse, within these business units. A distinction was made between the various roles, which are analysts, developers and project managers. An overview of the expert panel with relevant background information is presented in Appendix J.

5. Validation results

This chapter presents and discusses the validation results of the reuse management tool. The validation results are divided into three separate sections according to the steps of the validation process. In the first section the results of the requirement check are presented. In the second section the results of the application process are presented. In the third second the expert evaluation is presented. In the last section the validation results are evaluated against the validation model. The validation model was discussed in the previous chapter.

5.1 Requirements evaluation

This section checks whether all the requirements are met as formulated in chapter 2. An overview of the requirements and related satisfaction is presented in Table 29. The results are discussed below the table.

Nr.	Requirements	Satisfaction
1	The management tool has to address the incremental reuse levels in line with the view of Griss [27].	Satisfied
2	The management tool has to address relevant reuse factors influencing software reuse activities.	Satisfied
3	The management tool has to support reusable artefacts, which are inherent to changes through several iterations.	Not satisfied directly
4	The focus of the management tool should be on code components, the tool should, however, not discard other reusable assets such as functional designs and test cases.	Satisfied
5	The management tool should not be limited to agile development environments, but also has to be applicable in other environments.	Satisfied
6	The management tool has to provide options to make it specific for the context of an individual organization.	Satisfied
7	The management tool has to address communication and control issues, relevant for leveraging knowledge over multiple projects.	Satisfied
8	The management tool has to provide guidance in its application and assessment.	Satisfied

TABLE 29: REQUIREMENTS SATISFACTION

The first requirement, mentioned in Table 29, states that the management tool has to support the incremental reuse levels of Griss [27]. The reuse levels of the management tool are based on a combination of the incremental reuse levels of Griss and the CMMI-DEV model [59]. The mapping results can be found in Chapter 3. The mapping itself it not new, but making it explicit provides a more solid basis for defining the used maturity stages. Because the incremental reuse levels of Griss are integrated in the used maturity stages, this requirement is considered satisfied.

The second requirement states that the management tool has to address relevant factors influencing software reuse. In chapter 3 the reuse factors are identified through a literature review based on the top 25 IS journals. This requirement is therefore considered satisfied.

The third requirement states that the management tool has to address changing reusable assets. This requirement is not directly satisfied as the changing aspect of reusable assets, as originally identified, is addressed at a more abstract level. At a more abstract level it is taken into the reuse factor about configuration and change management.

The second fourth states that the management tool has to focus on code components. By using the incremental levels of Griss [27] as the basis for the management tool, all the reuse factors at all reuse levels address code components directly or indirectly. The code components are ranging from ad-hoc copied pieces of code, to white-box leveraged components, black-back components and architected components. This requirement is therefore also satisfied.

The fifth requirement states that the management tool should not be limited to agile development environments alone. Although this seems to contradictive, it appeared from the literature review in chapter 2 that agile development methodologies can be easily scaled up to more complex environments. It would therefore be a pity to limit the management tool to traditional agile development environments. During the set up of the management tool, as presented in chapter 3, the tool was not limited to agile development environments. This requirement is therefore also considered satisfied.

The sixth requirement states that the management tool should provide options to make it specific for the context of an individual organization. This requirement was introduced to be able to make the management tool tailorable for an agile development environment. With the introduction of the relevance variable, discussed in chapter 3, the possibility is provided to tailor the management tool to the context of an individual organization. This requirement is therefore also considered satisfied. As will become clear later in this chapter the use of the relevance variable can be further improved on.

The seventh requirement states that the management tool has to address relevant communication and control issues, to leverage knowledge outside the project scope. During the set up of the management tool in chapter 3, two reuse factors are made explicit to focus emphasis the focus on communication and control issues. These are the reuse factors ‘communication channels and organization support’ and ‘communication tool support’. Therefore this requirement is also considered satisfied.

The eighth and the final requirement states that the management tool should provide guidance in its application and assessment. In chapter 3 the assessment component of the management tool is introduced, which makes it possible for organization to identify strong and weak points regarding the software reuse factors. Furthermore the chapter provides some basic guidance regarding the application of the management tool and the interpretation of the results. This reuse factors is therefore also considered satisfied.

Summarizing the results of this section the conclusion is drawn that the requirements are, as far as possible, satisfied. The coming two sections elaborate on the application results and the expert opinions.

5.2 Application results

The section provides an overview of the application results. The application results are the assessment results and the insights gained by the application of the management tool. The application of the management tool is done by the experts presented in Appendix J. An overview of the factor scores is presented in Appendix O and an overview of the factor relevance results is presented in Appendix P. This section elaborates on the individual reuse factors and discusses the results. At the end of this section the results are summarized.

5.2.1 Business factors

This sub-section discusses the application results of the business factors. Although the term ‘business factors’ is used, it is actually only one factor namely the ‘domain focus’. After presenting the results of the domain focus the reuse factor is evaluated.

Domain focus

The assessment of the domain focus leads to an average score of 6,3 indicating that the experts noted characteristics of moderate levels of reuse. The relative high standard deviation indicates that several experts indicated higher reuse levels, while others indicated lower reuse levels. By analyzing the comments provided by the experts, the conclusion is drawn that the differences in scores cannot be related directly to the business units, rather differences depend on the interpretation of the reuse factor by the experts. The factor was considered very relevant with an average relevance of 4,0 and a low standard deviation. An overview of the assessment results is presented in Table 30 and further elaborated on below the table.

Reuse factor: Domain focus			
Score (0-10)		Relevance (0-5)	
Average:	6,3	Average:	4,0
Standard deviation:	1,3	Standard deviation:	0,9

TABLE 30: DOMAIN FOCUS RESULTS

All business units have defined their own market niche. In some cases there may be overlap between the defined market niches, offering opportunities for collaboration between the business units. Some business units have matured their solutions more for their market niche than others, but all of them have relevant domain experience. One of the experts noted that: '*the domain in which we operate is for a large part defined'*. Because the domain focus is for a large part defined it is also architected to a certain extend. An expert mentioned the following about that: '*many components assume that you have a person, an address, an organization and that kind of things*'. '*You can almost pull them out of a box in order to use them*'. An expert of another business unit adds to this that: '*everything we do is tailored work and afterwards we make components of it, in order to sell them in the future*'. Furthermore he noted that: '*a true architecture is not used*'. Based on these findings the conclusion is drawn that the domains are defined in such a way that certain architectural characteristics of components are found.

By analyzing the assessment results in more depth it becomes clear that two reuse situations are possible. First of all the business units subject to research all make use of the so called REPA components. The REPA components are a set of generic components providing functionality which can be used over multiple projects and multiple domains. Beside the general set of components, each business unit also uses a set of domain specific components. The two sets of components are managed differently which will become clearer based on the analysis of the other reuse factors. Because the two sets of components are managed differently, the interpretation of the experts was also different. An expert working closely with the REPA components often scored the reuse factors according to the characteristics of these components. And an expert working with domain specific components scored the reuse factor according to characteristics of domain specific components. Beside that several experts mentioned that the scoring of a reuse factors was difficult because the two sets of components are managed differently. A score in between was added in the later case.

The use of an architecture is investigated further. The experts noted that if component X has been create for project A. And component Y and Z are created for project B. It is relatively easy to use component X and Z in project C. This is also the way components are sold and has implications for the way a product architecture is reused. The market demands, embedded in the projects, are leading the development of components instead of a predefined product architecture. Because the market demands are leading the development of components, a dynamic architecture becomes visible. It is, however, imaginable that a large organization is still able to create a product based on predefined product architecture, but this is not the way how it works for this case study. The way an architecture is reused is in contrast with the way product architectures are defined in literature [27, 71]. Literature almost

always refers to predefined product architecture, where components have to fit within that architecture. Related to the reuse example mentioned at the beginning of this paragraph, another way of reusing was mentioned by the experts. In some cases a previous solution is used as the basis for creating a new solution. In that case the previously created solution is tailored to the demands of the customer. By basing the new solution on a previous solution its architecture is reused as well. This way of reusing can significantly reduce development time, but if it is not possible a new product is built based on the demands of the customer.

Summarizing the results of these reuse factors: the case organization has a domain focus with moderate levels of similarities among products. In some cases product architectures are reused and components often have characteristics common for most solutions. The use of an architecture is, however, not leading the development components as proposed by traditional literature. Furthermore the domain focus is judged as very relevant with a score of 4,0 and a relatively low standard deviation. The experts also mentioned that the factor does not require scaling to a higher level, as the results then would not match the way how reuse is occurring at the case study anymore.

Evaluation business factors
The business element 'domain focus' is considered a very relevant reuse factor for software reuse. The way the domain focus was interpreted by the experts was different than expected. The experts emphasized that the demands of the customer are determining the domain focus and not the software development organization itself. It is, however, imaginable that a large organization is able to determine its domain focus by creating a product family and putting it into the market. This research did, however, not confirm such possibilities. No additional business factors where suggested by the experts.

5.2.2 Organizational factors
This sub-section discusses the organizational factors. The organizational factors are 'top management support and instrumental mechanisms', 'team structure and reuse roles', and 'communication channels and organizational support '.The final paragraph of this sub-section evaluates the organizational factors.

Top management support and instrumental mechanisms
Top management support and instrumental mechanisms appeared to be more difficult to assess. The experts added an average score of 5,4 to top management support, with a large standard deviation of 2,1. The factor was, however, quite constantly considered very relevant with an average score of 4,2. An overview of the results is presented in Table 31 and elaborated below.

Factor: Top management support and instrumental mechanisms			
Score (0-10)		Relevance (0-5)	
Average:	5,4	Average:	4,3
Standard deviation:	2,1	Standard deviation:	0,5

TABLE 31: TOP MANAGEMENT AND INSTRUMENTAL MECHANISMS RESULTS

Software reuse activities are stimulated and encouraged within all business units. As one of the experts noted: '*all the directors of the business units support reuse opportunities and software reuse is considered an integrated part of the selling strategy'.* Top management also assigned dedicated resources to the REPA developers. The REPA developers are senior developers who act as code owners of generic code components. The REPA developers have time available to work on the generic components and to manage these. Furthermore they have time available to plan meetings and discuss the most recent developments. Reuse is also regularly stimulated and encouraged through the various phases of the development process. All developers are encouraged to create code components in a

generic way, but often in the later stages of a project the priorities tend to shift to the project itself again, as deadlines have to be met. Several experts noted that the production of components is barely happening outside the project scope, as such a component often has to be created within the normal project scope. Some other experts indicated that it is possible to request additional time to create components, but that it is preferable to do this at the beginning of the project.

The reason why the high standard deviation is present is that most developers are struggling with the dedicated resources. Resources are provided, but within a business unit it is simply not possible to assign full-time job roles to software reuse. As a project leader mentioned: *'I don't see it happening that I add another project member to my project group for reuse activities only'*. Of course there are other ways to assign dedicated resources, but this statement perfectly illustrates that there are no such resources available. Beside that, the data set includes one extreme score, where one expert scored this reuse factor with a 10. He scored the reuse factor with a 10 because he felt that reuse was integrated in the culture. During the scoring process he likely did not check the practices thoroughly. The score of a 10 blurs the results a bit. Without this score a standard deviation of 1,2 is found, which is a relatively normal standard deviation.

To conclude the results of this reuse factor, top management supports and stimulates reuse initiatives. Some experts mentioned that this factor could require some scaling, as developers would like to have more resources. This is, however, not always possible due to resource constraints. The average score of 5,4, despite the scoring difficulties , nicely represents the state of the case organization. The factor is considered very relevant, because reuse events will simply not happen without top management support.

Team structure and reuse roles

The reuse factor ‘team structure and reuse roles' proved to be stable factor during the assessment process, with both a low standard deviation in its score and relevance. The reuse factor scored an average of 4,6 with an average relevance of 3,9. The insights gained from the experts confirm that this is a stable factor. An overview of the results is presented in Table 32 .

Factor: Team structure and reuse roles			
Score (0-10)		Relevance (0-5)	
Average:	4,6	Average:	3,9
Standard deviation:	1,1	Standard deviation:	0,9

TABLE 32: TEAM STRUCTURE AND REUSE ROLES RESULTS

Almost all the experts doubted between a score of 6 and a score of 4 during the assessment process. The reason for this is that the generic set of REPA components is managed by a separate reuse team, but the other code components are not managed by such a team. All the components are stored within a revision management system, this can be a project specific revision management system or a central one. By checking the component in such a system the developers are able to identify the person who created the component. This person can be viewed as the code owner, as he or she is contacted for additional information regarding the functionality of the component. Formal code ownership is only acknowledged for the REPA components. Based on the background information provided by the experts it is questionable whether code ownership for all components is possible or desired at all. As one expert noted: *'I wonder how one does it when you have 10 components in a project, one does not have time to meet all the component owners for negotiation and explanation'*. Another expert added that: ‘*it sounds nice in theory, but I wonder how one would do it in practice'*.

When taking a closer look at the team structure a combination of the models of Fafchamps [70] and

Bosch [71] is found. An overview of the combined approach is presented in Figure 15 and elaborated on below the figure.

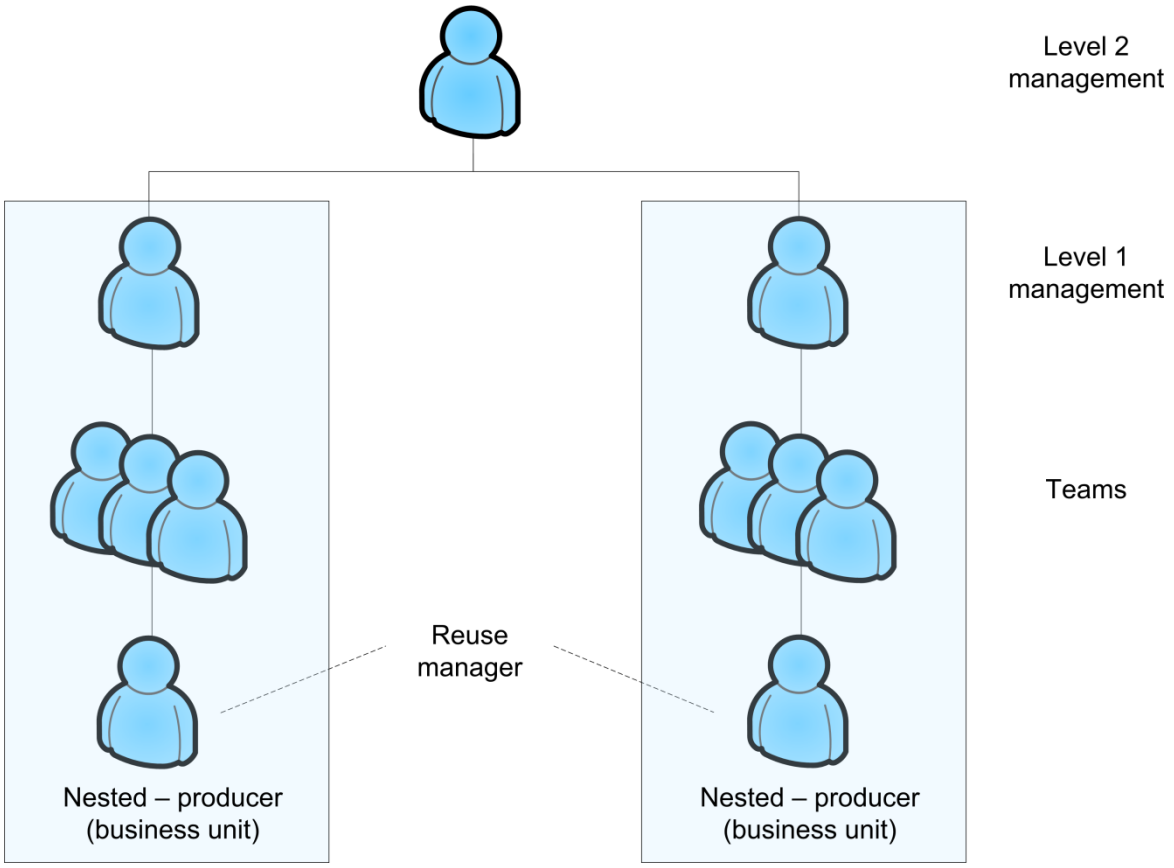


FIGURE 15: NESTED STRUCTURE DERIVED FROM [70].

The nested producer model of Fafchamps defines [70] a producer as a member of a regular development team, where he or she has a dual reporting structure. Both the development team and the reuse manager have to be informed of achieved activities. The reuse manager at the case organization is a REPA developer or a code owner itself, no additional management layers are needed or implemented. Fafchamps reported the nested producer structure as an unsuccessful model, because the dual reporting structure is likely to lead to conflicting tasks and responsibilities [70]. The allocation of reuse resources is covered by multiple business units similar to the business unit model of Bosch [71]. Within the business unit model Bosch distinguishes various levels of authority [71]. For the case organization a combination of a constrained and an unconstrained model is used. In practice this means that everybody can change the reusable assets (unconstrained) when the outside behaviour of the component (the interface) does not change. When the outside behaviour does change it is considered good practice to contact the code owner to discuss the changes. In that case a constrained model is used. Both Fafchamps and Bosch report situations related to the used team structure with potential delays, due to change negotiation and conflicting roles for the code owner [70-71]. To deal with these situations some business units extract the components from the central repository and adjust them locally. When doing this the risk is high that multiple versions of the same component are created, which have to be merged together at a later point in time. The way the tasks and roles are divided seems to work for the case organization at the moment.

Together with the description of the team structure the roles become clear as well. Within the case organization the role of a REPA developer is present and more generally, code owners. The REPA developer is a rather formal role and he or she also has dedicated resources available to spend on the

REPA components. The code owner role is more informal. A developer becomes a code owner once he has created a potentially reusable component and other developers start reusing that component. During the discussion of the people factors results it will become clear that the roles of producer and consumer of normal code components is interchangeable. A consumer reusing a normal software asset is likely to become a producer rather quickly, when he or she adjust the existing software assets.

Now the background has been provided regarding the team structure and the reuse roles, the score can also be explained. The average score of a 4,6 lies between the presence of an informal reuse teams and an extended reuse team responsible for managing work products. Both cases are true depending on the component itself. The REPA components are managed by an extended reuse team and the other components are managed by a code owner. This code owner can be a formal or an informal code owner, as explained in the previous paragraph.

Summarizing the results of this reuse factor it is evident that the organizational structure reuse roles are a very relevant factor. In case a formal reuse team is used to manage the set of generic REPA components, informal code owners are used to manage the other components. The experts noted that it would be nice to have code owners for all components, but this does not seem feasible at the moment. The score of 4,6 nicely represents the current state of the case organization.

Communication channels and organizational support

The assessment process resulted in an average score of 4,5 with a relatively normal standard deviation. The average relevance resulted in 3,3 with a similar normal standard deviation. The relevance of this reuse factor was consistently lower than the other organizational reuse factors. The assessment results of communication channels and organizational support are presented in Table 33.

Factor: Communication channels and organizational support			
Score (0-10)		Relevance (0-5)	
Average:	4,5	Average:	3,3
Standard deviation:	1,1	Standard deviation:	1,0

TABLE 33: COMMUNICATION CHANNELS AND ORGANIZATIONAL SUPPORT RESULTS

The communication channels at the case organization are installed through the use of REPA developers. REPA developers are senior developers responsible for managing the set of generic REPA components. These senior developers are located within the various business units and have frequent meetings with each other to discuss the latest reuse events. The results of such meetings are spread out through the REPA developers to the other members of a business unit. The other way around, the local developers within a business unit are contacting the REPA developers of their business unit for knowledge regarding reusable assets. The REPA developers are often able to forward them to the relevant persons. Within the case organization it is considered good practice to first contact the persons within the business unit, before contacting persons outside the business unit. The thresholds for contacting individuals outside the business units are kept low. The organization provides regular team building activities for the various business units. Aside from that members of business units may be hired to another business unit. Also various individuals who are working in one business have worked at another business unit in the past. Most employees know each other informally and share mutual feelings of trust. An overview of the installed communication channels is presented in Figure 16.

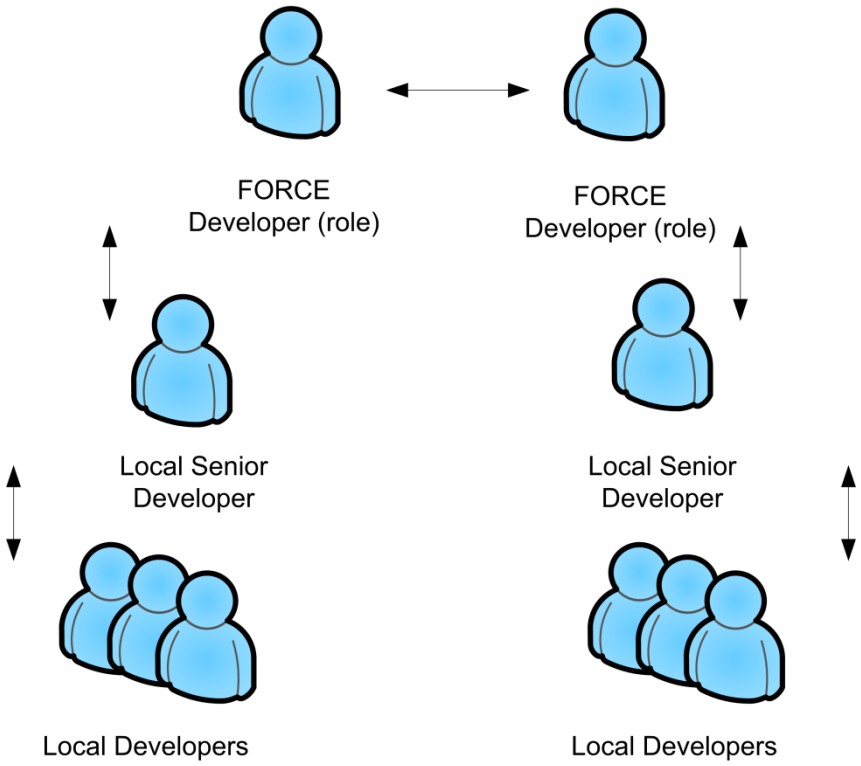


FIGURE 16: INFORMAL COMMUNICATION CHANNELS

Several experts did note that not all information is flowing to the other business units. In addition to that, some developers would like to have more information regarding reusable assets, as they have no clear insight into what the other business units are doing. The assumption is made that those components which are not made centrally available are not relevant to the other business units. This is illustrated by a quote of one of the experts stating: '*I would like to know more about components from other business units, but on the other side they are most likely not relevant for me*'. The general opinion of the experts is that the communication channels should remain informal and that the best way to obtain information is by simply asking someone. Good documentation in general could support the communication. Furthermore a REPA developer noted that communication could be better when new components are being developed, changes should be more regularly discussed. When the components are most stable the communication issues are not identified as a problem by this expert.

After analyzing the results in more depth it appears that not all the business units have a REPA developer in the REPA team. This also appeared from the assessment results where the two individuals within this business unit scored the reuse factor lower than the other business units. Although information is also flowing to that business unit it obviously helps that a senior developer is involved in the regular REPA team to stay up-to-date of the latest reuse events. This point is further elaborated in the recommendations in section 6.2.

In conclusion, the results found for this reuse factor make it clear that communication channels and organizational support are important for addressing software reuse issues. The reuse factor did appear to be less relevant than the other organizational factors, which is logical as this reuse factor only emphasizes certain organizational aspects. The average score of 4.5 matches the insights gained from the experts.

Evaluation organizational factors

The organizational factors are all considered relevant. The experts did not suggest additional factors were to improve the management tool. Furthermore no factors were identified as superfluous. However, a number of notes were added. First of all the setting of priorities appears to be important for software reuse. This can be related to the dedicated resource taken into the reuse factor of ‘top

management and instrumental mechanisms'. The setting of priorities indicates that reuse activities require additional time and resources. Beside that the priorities of creating reusable assets moved to the background once a project deadline is approaching. This is also in line with literature, Morisio et al. [7] for example noted that reuse is always a secondary activity. Another frequently made comment was that there are no resources available to set up dedicated reuse teams as suggested by Fafchamps and Bosch [70-71]. The organizational factors together provided a coherent way to address software reuse issues, the way components are addressed differs depending on the component itself, so different scores are added based on different interpretations. This might have to be taken into account for further research.

5.2.3 Process factors

This sub-section discusses the process factors. The process factors that are taken into the management tool are: 'reuse planning', 'reuse measurement', 'requirements management', 'quality management', 'supplier management' and 'configuration and change management'. After presenting and discussing the reuse factors individually the last paragraph of this sub-section evaluates the process factors as a whole.

Reuse planning

The reuse factor 'reuse planning' scored an average of 4,2 with a rather high standard deviation. The factors is identified as relevant with 3,2 and a relatively normal standard deviation. As will appear from the results of the other process factors their relevance is lower compared with the organizational and business factors. An overview of the assessment results for reuse planning is presented in Table 34.

Factor: Reuse planning			
Score (0-10)		Relevance (0-5)	
Average:	4,2	Average:	3,2
Standard deviation:	1,9	Standard deviation:	1,2

TABLE 34: REUSE PLANNING RESULTS

In all four business units the planning of reuse events occurs on different levels. Some components are so obvious that they are identified as high potential reuse areas and are therefore planned. Normally the reuse events are planned within the normal project scope, but in some rare cases this can also be planned outside the normal project scope. A separate project is then set up to create that specific reusable asset. Furthermore reuse events can be planned at the beginning of a project or it can occur in a later stage when an experienced developer recognizes a potential reuse opportunity. In the first cases the reuse events are still planned, in the later case the reuse events occur ad-hoc based on the experience of the developers. The different ways of reuse planning, or in some case the ad-hoc creation of components, explains the high standard deviation found by the assessment process.

After investigating the results in more depth based on the insights gained from the experts, it appears that the developers need a certain amount of freedom to stimulate innovation. It would, however, be better if more reuse events could be planned at the start of a project. One of the experts made this comment explicitly by stating that '*it would help if reuse planning would occur at the start of a project*'. When reuse events are planned at the start of a project it is more likely that the components are really created, even when deadlines are approaching. Planning the reuse events outside the project scope is more difficult and is only considered to be suitable for the very obvious components, as mentioned earlier.

Summarizing the results found by the assessment process for this reuse factor it is clear that reuse planning is an important reuse factor. It also appeared that the scoring of the reuse factor is difficult as

multiple ways exist within an organization to plan and create reusable assets. The factor is identified as a relevant factor and the experts noted that reuse planning can be improved more for the case.

Reuse measurement and cost models

The reuse factor 'reuse measurement and costs models' resulted in an average score of 2,1 with an above average standard deviation. The relevance of the reuse factor was rather low with an average of 2,4. Both the relevance low score and the low relevance indicate that this reuse factor is less interesting for the case. An overview of the assessment results is presented in Table 35.

Factor: Reuse measurement and cost models			
Score (0-10)		Relevance (0-5)	
Average:	2,1	Average:	2,4
Standard deviation:	1,5	Standard deviation:	1,3

TABLE 35: REUSE MEASUREMENT RESULTS

The vast majority of experts noted that the tracking of reuse events is barely present. In some case it is possible to track the reuse events manually, but not always. In the four business units no software reuse related cost models are used. Although the results are not as expected it works for the case organization. One of the experts illustrated this by noting that: *'often we have a feeling when something is reusable or not. In practice it means that you set up a generic component based on that feeling. Eventually it will appear that the components which we created are relevant'.* Another expert adds to this that: *'we never make components, which are not used at all*'. A project manager noted that '*if you don't make a generic component you will always have a customer specific solution*', which is not a desired situation for the product development strategy. The project manager also indicated that he is able to estimate the costs roughly, which was sufficient for him. Another developer carefully noted that the measurement of reuse events would be relevant, because it has to do with priority. When it appears that a component is frequently reused more resources should be made available to manage this component. This last remark is in line with literature, as frequently used component should be set up more carefully. Interestingly, this view was also contracted by a project manager who stated that: '*poorly set up components will be reused as well, but it simply take more time to do this successfully'*.

In conclusion, the statements about this reuse factor show that the factor is a relevant factor for software reuse, but not a necessary factor. The experts noted that reuse events and costs can be estimated based on the feeling of developers. The experience of the developers and the relevant domain knowledge likely plays a crucial role for making such estimations. Both the score and the relevance are representative for the findings of the case study.

Requirements management

The reuse factor 'requirements management' scored an average of 3,3 with a relatively low relevance of 1,1. Both the score and the relevance have a remarkably low standard deviation, indicating that this factor was scored quite consistently. An overview of the assessment results is presented in Table 36.

Factor: Requirements management			
Score (0-10)		Relevance (0-5)	
Average:	3,3	Average:	1.1
Standard deviation:	0,9	Standard deviation:	0.6

TABLE 36: REQUIREMENTS MANAGEMENT RESULTS

The management of requirements is done in several ways. Discussion with the experts revealed that it is mostly the individual who recognizes similar requirements and matches them with previously created components. Beside that it appeared that the requirements are analyzed at project level, the requirements document itself is set up specifically for the customer. Therefore the requirements document is only usable in a limited manner for sharing across multiple project as they are tailored to the demands of the customer. This means that the requirements are at customer level and not at product or component level. Of course some information can be obtained from the requirements document as all the artefacts represent the final solution. For some components a separate document is set up with some requirements. These are often the components of which multiple versions are created and criteria are required to merge the versions back together again. The criteria to merge the version back are then the requirements set up specifically for the component.

During the discussion with the experts it appeared that the idea of keeping track of requirements and linking them to the components has been frequently proposed by the business. But so far, nothing has been done with it. The difficulty seems to be the requirements, as they are currently not reusable across multiple projects. This confirms the statement made in the previous paragraph that the requirements are set up at customer level.

Furthermore, the experts noted that negotiation with the customer happens in order to promote the reuse of available functionality embedded in existing components. One of the project managers illustrated this: '*When you are sitting in a workshop with the customer you try to work towards the available components. You explain to the customer if you do this in such a way then you have the same functionality. So, we absolutely take the available functionality into account, but there is no document where the requirements for the components are stored'*. Another expert complemented this statement by noting that '*a component always exists within a project and depending on the requirements of the customer a component is developed into a certain direction'.* In other words negotiation with the customer regarding the reuse of existing components occurs, but in the end the wishes of the customer are leading the development of reusable assets. The knowledge regarding the reusable assets is stored in the minds of employees, there is no component specific requirements document or other documents to support the knowledge exchange. At a higher level there is a document available though which provides an overview of the available components.

The experts did note that the tracking of requirements for individual components could be useful, mostly in the form of additional documents. Setting up requirements in advance and strictly managing them is not perceived as useful.

When summarizing the findings of this reuse factor it appears that requirements management can provide a valuable starting point and it is regularly discussed by the company. The strict management of requirements does not seem to be useful which seems to be the reason why this factor scored a low relevance. Approaching requirements management in the broad sense, by also including other documents, it is even more valuable. The assessment results of this reuse factor are in line with the insights gained from the experts.

Quality management

The reuse factor 'quality management' scored an average of 2,3 with a relatively normal standard deviation. The relevance of the factor was rather low again with an average of 2,7 with a similar normal standard deviation. An overview of the assessment results is presented in Table 37.

Factor: Quality management			
Score (0-10)		Relevance (0-5)	
Average:	2,3	Average:	2,7
Standard deviation:	0,9	Standard deviation:	1,0

TABLE 37: QUALITY MANAGEMENT RESULTS

The insights gained from the experts showed that quality is mostly based on the experience of senior developers. There are some tools running in the background to check the component structure, but warnings do not necessary touch on the quality and are not systematically checked. The experts also note that unit testing is applied, but this does not necessary say anything about the quality of components. One of the experts noted that: 'W*ith component testing we do not make a distinction between the quality of the component and its implementation. It could be the case that there are mistakes in both, but that the component works'*. Although some things are happening in the background no quality model is used to check the quality of components. Furthermore the quality of components is not necessarily checked before components are populated into a repository.

After engaging in discussion with the experts it became evident that they do not rate the use of quality models as important. When a components appears to be of insufficient quality it will be improved the next time it is used. The disadvantage of this strategy is, however, that the other projects using this component may have to be updated as well. Furthermore this train of thought indicates that there may be a relation between the amount of times a component is reused and its quality. This is also illustrated by a statement of one of the experts: '*There are components which are used in one or two projects and obviously work fine, so quality issues play no role for these components. But, we also have components which are used frequently, or which are very important for the customer. For example a mistake in the interest checking component costs a lot of money. So depending on the component itself testing and the quality of components becomes more important.*' Not only does this statement identify the relation between the quality of components and the amount of times it is used, it also indicates that the impact of mistakes in a component is relevant to the required quality. When the impact of a mistake is great it is likely that a component is more carefully tested and that the quality of a component is high.

Another expert noted that the quality of the components is often in the minds of developers, it would help to provide additional documentation. Otherwise, this factor is not something the experts would like to see scaled.

When summarizing the results found by assessing this reuse factor, the conclusion is drawn that the quality of components is important, but also that quality issues possibly solve themselves through various iterations based on the demands imposed by several projects. The quality of components can be expected to be high when a component is used frequently or when the impact of mistakes in a component is high. The use of quality models was not applicable for this case study, but it may be more applicable in larger organizations. The assessment results of the reuse factor are representative for the findings of the case study.

Supplier management

The reuse factor 'supplier management' scored 1,8 with a relatively low standard deviation. The factor also scored a low relevance of 1,1 with a similar standard deviation. An overview of the assessment results is presented in Table 38.

Factor: Supplier management			
Score (0-10)		Relevance (0-5)	
Average:	1,8	Average:	1,1
Standard deviation:	0,7	Standard deviation:	0,6

TABLE 38: SUPPLIER MANAGEMENT RESULTS

The various experts mentioned that there are some reuse events where externally acquired assets are used. For example a semantic component is bought, which is able to compare strings. An expert noted that the use of such components is: '*not very exciting, as there are 100 and 1 implementations available, so it is not useful to create another one*'. The experts were in general not positive about externally acquired assets, black-box components often do not provide the required functionality. The result is that components are bought but not used at all because a slightly different functionality is required. Based on that, the experience with black-box component markets is rather bad. It only seems to work for components which are (re)invented several times and have matured so far that they do not have to be invented again. The amount of black-box components used within the case study is minimal and no supplier management information is stored formally.

The use of open source projects, in contrary with black-box component markets, was received very well by the experts. Several open source components are used such as .Spring and NHibernate. The use of components from open source projects can be considered as white-box reuse, as the developers have insights in the internal logic of the components and can adjust the component to the demands imposed the projects. Because the user can view the internal characteristics supplier management is less relevant for such component. For this case study also no supplier management information was stored for the reusable assets.

After summarizing the results of this reuse factor it becomes clear that the factor is less relevant for the case study. In large organizations in which more externally acquired assets are used the factor can be expected to be more important. Furthermore the factor may become more important when components markets are becoming more mature. Based on this research no direct conclusion can be drawn regarding this research factor, additional research is required to say something about it.

Configuration and change management

The reuse factor 'configuration and change management' scored 3,3 with a relatively low standard deviation. The factor scored average relevance of 3,5 with a similar standard deviation. An overview of the assessment results is presented in Table 39.

Factor: Configuration and change management			
Score (0-10)		Relevance (0-5)	
Average:	3,3	Average:	3,5
Standard deviation:	1,1	Standard deviation:	1,0

TABLE 39: CONFIGURATION AND CHANGE MANAGEMENT RESULTS

The insights gained based on the discussions with the experts lead to several findings. First of all it appeared that configuration and change management is performed at two different levels, similar to what was mentioned earlier during the discussion of other reuse factors. The REPA components are guarded more systematically by the REPA developers and are managed centrally in a separate revision management system. Consumers can freely download the reusable assets and use them in new projects. The consumers can also place change requests at the REPA developers. The REPA

developers then determine whether a change is accepted or not. If the change is accepted the requester, the consumer, populates the change into the specific reusable asset. In that sense the REPA developers function as a change request board, who are responsible for evaluating the change requests. From this perspective characteristics are found of ‘configuration management based on reusable assets ‘and ‘configuration management for reusable assets done by a separate group’. The other reusable assets are managed within the project. They make use of the revision management system of the project and they are subject to the changes within that project. When another project wants to use these components they often have to pull the specific component out of the original projects’ repository first, before the component can be reused. In order to do this the developer in question will have to remember where the reusable asset is stored. In the original project the component has hopefully been set up in such a way that it can be pulled out of the project with relatively little effort. The related planning process became clear from the reuse factor ‘reuse planning’, which has been discussed earlier in this chapter. Furthermore, the expert noted that for the components in general there is little configuration management required, but much change management.

As evident from the previous paragraph, the experts saw characteristics of several levels of reuse, but this did interestingly enough not result in a very high standard deviation. The reason for this is that all the experts consistently recognized the similarities and added a score in between.

Based on the discussion with the experts it also became clear that configuration and change management is not as easy as it seems. Central configuration and change management is difficult as individual projects often want to reuse a component slightly different than the way they are made available, leading to different versions. The experts also noted that communication is very important when different versions are created, otherwise the components will certainly start to ‘live their own life’. Several experts also emphasized the importance of configuration and change management when multiple projects are simultaneously working on a component. This might require additional attention in the future.

When summarizing the results of this reuse factor the conclusion can be drawn that configuration and change management is a very relevant factor, but also one which is difficult to assess. Configuration and change management becomes increasingly important when multiple projects are working on the same component. Not only does configuration and change management become more important, but also the related communication issues. The results found by the assessment methodology are in line with the insights gained by the experts. The reuse factor can therefore be considered as a valid factor. The way reuse events are taking place within the case may require additional attention in the future.

Evaluating process factors

After evaluating the process factors the conclusion is drawn that they are well set up in general. The CMMI-DEV is successfully used as a guideline for mapping the reuse factors against the maturity stages. The experts did sometimes experience problems with filling in the assessment tables, mostly because of the fact that the REPA components are managed differently compared with the other components. The process factors scored consistently lower than the other reuse factors, such as the business and organizational factors. This results can be used as a valuable indicator for noting that agile development environments do not have systematic reuse processes in place. The lower relevance of the reuse factors indicates that such systematic reuse processes are also not desired. Based on the outcomes of the various process factors it seems to be best to plan the reuse events as early as possible in the reuse process, to make sure that there is enough time available to create the components. Evaluating the reuse factors with the use of the experts the conclusion is drawn that the process areas of software reuse are covered. One expert noted that the main pillars are *'planning, quality and costs'* and these are more than covered. No additions were mentioned for the reuse factors.

5.2.4 People factors

This sub-section discusses the people factors. The people factors that are discussed are 'producer skills and experience and 'consumer skills and experience'. After presenting and discussing the application results the last paragraph evaluates the people factors.

Producer skills and experience

The assessment of the reuse factor 'producer skills and experience' resulted in an average score of 5,0 with a relatively high standard deviation of 1,7. The average relevance was 3,4 with a relatively normal standard deviation. An overview of the results is presented in Table 40.

Factor: Producer skills and experience			
Score (0-10)		Relevance (0-5)	
Average:	5,0	Average:	3,4
Standard deviation:	1,7	Standard deviation:	1,2

TABLE 40: PRODUCER SKILLS AND EXPERIENCE RESULTS

Based on the practices defined for this reuse factor the experts noted that experienced developers are used as producers, some best practices are defined and the code owners function sometimes as coaches to the consumers. Experienced developers are used for several reasons. First of all the use of experienced developers results in higher quality components. Beside that the decisions made by experienced developers are better accepted by other developers. This statement is also confirmed by literature (e.g. [70]). Lastly a more experienced developer with relevant domain knowledge is better able to separate the project specific functionality from the general functionality. It is the general functionality which is applicable throughout the entire domain. The project specific functionality should not be taken into the component, as the component will then be usable in a limited manner. The best practices are common rules of behaviour and are not formalized into procedures. It is for example considered best practice to contact the code owner before a consumer starts working on a component. Throughout this chapter several other best practices are described and are not further elaborated on at this point. The coaching role mentioned by the experts is an interesting role. When a consumer of a reusable assets wants to reuse the assets he often contacts the code owner. The code owners provides the user with the required knowledge and basic instruction for reusing that component. When a consumer starts working with the component and changes it he actually becomes a producer as well. During the process of transforming from a consumer to a producer the code owner functions as a coach teaching the required skills and knowledge. The changes proposed by the consumer may be populated into the original component, depending on the approval of the code owner. Within the case study no formal trainings are provided regarding software reuse.

The experts emphasized during the discussion that the producers roles are not full time roles, but rather part time roles. The use of full-time roles is not a desired situation at the moment either. Furthermore they noted that they noted that the score granted to the producer role is lower than the way how they perceived it. According to them the producer role should score a least an 8 as they are really satisfied about the experience and skills of the producers.

The high standard deviation can be related to the fact that the experts mentioned several characteristics of multiple stages. But also another issue plays at this reuse factor, namely that the People Capability Maturity Model [84] may be less suitable for its application in an agile development environment. The People Capability Maturity Model seems to be better applicable in a large organization. Smaller organizations do not formally evaluate or address people issues. This is well illustrated by a comment made by one of the experts, stating that: '*the development of skills is present, it is only not structurally present*'.

When summarizing the results found for this reuse factor the conclusion is drawn that the factor is able to address relevant producer aspects. The way how this factor addresses the people aspects might require additional research in the future, as the People Capability Maturity Model seems to be more suitable for larger organizations. The producer experience and skills do not require further scaling for the case subject to research.

Consumer experience
The assessment of the reuse factor 'consumer skills and experience' leads to an average score of 4,7 with a relatively high standard deviation of 1,5. The average relevance was 2,9 with a standard deviation of 1,3. An overview of the assessment results is presented in Table 41**.**

Factor: Consumer skills and experience			
Score (0-10)		Relevance (0-5)	
Average:	4,7	Average:	2,9
Standard deviation:	1,5	Standard deviation:	1,3

TABLE 41: CONSUMER SKILLS AND EXPERIENCE RESULTS

During the assessment process the experts mentioned several times that the consumers do not differ so much from the producers. As illustrated by an expert: '*it is difficult to separate consumers from producers, as you become a producer rather quickly when working with the components*'. When consumers are working on multiple projects in the same domain they become aware of relevant domain knowledge. During these projects they also get in touch with the reusable assets. As such, along with the domain knowledge also the knowledge of reusable assets is increasing. The characteristics of the consumers were not further elaborated on by the experts.

The expert did note that the consumer may have to be supported more in the future, as case organization is continuously growing and expanding at an increasing pace. When more and more new employees are accepted it would be wise to provide additional guidance to support the starting consumers. Chapter 6 elaborates on the possible recommendations identified by the experts.

When summarizing the results found for the consumer skills and experience factor the conclusion can be drawn that there are many similarities between the producer skills and experience factor. This can also be related to the interchangeable roles, which are characteristic for agile development methodologies. See for more information about the agile development characteristics also Chapter 2. Consumer skills and experience was judged slightly less relevant than the producers, this may be explained by the fact that consumers are likely to quickly turn into producers themselves. A formal separation between producers and consumers as reported by Bosch [71] with software product lines was not found. The separation of the two reuse roles does provide a way to structure the concept of software reuse. Lastly, this reuse factor was based on the People Capability Maturity model, which seems to be more suitable for larger organizations instead of smaller ones. Additional research is required to specify the practices in such a way that this factor is also applicable for smaller organizations.

Evaluating people factors
When evaluating the people factors it became clear that the distinction between producers and consumers is less relevant for the case study, but it does provide a systematic way to analyze the people factors and discuss related issues. The division of the two reuse factors is done according to the People Capability Maturity Model [84]. The division according to this model is arguably more suitable for larger organizations. Additional research is required to identify relevant practices which are better applicable for both smaller and larger organizations. No further additions were suggested for the people factors by the experts.

5.2.5 Technology factors
This sub-section discusses the technology factors. The technology factors taken into the management tool are: 'repository support', 'CASE tool support' and 'communication tool support'. After presenting and discussion the technology factors the last paragraph of this sub-section evaluates the technology factors as a whole.

Repository support
The reuse factor 'repository support' scored an average of 4,7 with a relatively low relevance of 0,7. The factor got a relevance of 3,9 with a similar low standard deviation. An overview of the assessment results is presented in Table 42**.**

Factor: Producer			
Score (0-10)		Relevance (0-5)	
Average:	4,7	Average:	3,8
Standard deviation:	0,7	Standard deviation:	0,7

TABLE 42: REPOSITORY SUPPORT RESULTS

The discussion with the experts revealed that for this reuse factor a distinction can be made between REPA components and the other reusable assets in the form of domain specific components. The REPA components are stored on a central repository and the domain specific components are often stored within the repository of a project. In both case the available revision management system is used. Within some business units a set of domain specific components is leveraged outside the project scope by storing it on the intranet or a separated revision management system. These components are frequently used within a business unit, but of little value outside the business unit. The amount of the latter set of components is very limited compared to the other two component sets. The revision management system only stores the code components, but not other reusable assets, such as functional designs and technical designs. In some cases such documentation can be found on the intranet, but this is only for the larger and more complex components. In general component specific information is scarcely available and when it is available it can be stored on multiple location, such as in project folders on the intranet. When several projects are making use of a component it is possible that information can be stored in multiple places possibly in multiple folders. In order to obtain related information a consumer can best contact the producer of a reusable asset, which is also what is happening at the various business units.

The experts were actively encouraged to go into discussion about the used repository and the future possibilities. They noted that the revision management can be further supported by the use of Wikipedia pages in order to create a more systematic way of providing documentation. Furthermore several experts where thinking about using stable releases, an idea derived from the open source community. This idea also lies closely to the knowledge library as presented in literature. The ideas are further elaborated at the recommendations in chapter 6.

When evaluating the results found by this reuse factor the conclusion can be drawn that the average score of 4,7 is a bit high. Based on the insights gained by the experts the score should be between 2 and a 4. No explanation could be provided based on the conversations with the experts why they scored the reuse factor higher than it should have been according to the assessment table. Possible the notion of knowledge libraries and stable versioning was not well explained by the reuse practices. This could be a point of attention for further research. The reuse factor itself was received as quite relevant by the experts.

CASE tool support
The reuse factor 'CASE tool support' scored a low average of 1,3 with an extremly high standard deviation of 2,3. The reuse factor was considered scarcely relevant by the experts with a relevance of 1,6 and a standard deviation of 1,3. An overview of the assessment results is presented in Table 43**.**

Factor: CASE tool support			
Score (0-10)		Relevance (0-5)	
Average:	1,3	Average:	1,6
Standard deviation:	2,3	Standard deviation:	1,3

TABLE 43: CASE TOOL SUPPORT RESULTS

Within the various business units Visual Studio is used as the CASE tool for software development. This CASE tool is normally not integrated with the repository of available components. This means that components have to be inserted manually into the CASE tool before they can be deployed into a project. There is one exception regarding the integration of the CASE tool with the components: the interface components. The interface components have to be integrated with the CASE tool before they can be used at all. Along with this statement one of the experts noted that: '*the integration of the CASE tool with the components is supported as far as necessary*'. In the past several experts have done attempts were application generators are created. Based on those attempts the conclusion was drawn that the solutions created within the case organization differ too much to make real benefits. A certain amount of flexibility among the solutions is required to be able to respond to fast changing markets, something which is difficult to maintain with an application generator. The costs of maintaining the application generator are expected to be much higher than the gains that can be made by them. Some experts did note that they were interested in the possibilities and gave a slightly higher relevance score. The general opinion, however, remains that CASE tool support has been integrated as far as possible.

The high standard deviation noted at the score of this reuse factor can be explained by a blurring score. One of the experts ranked the reuse factor with a score of 7, apparently measuring the use of CASE tool support in general, instead of for software reuse specifically. After noting the misinterpretation during the application process, the reuse factor was explained slightly different to the other experts. When ignoring the polluting data entry a standard deviation of 0,6 can be found, indicating that the experts were actually quite consistent in scoring the reuse factor.

After evaluating the results of this reuse factor it becomes evident that the experts consider the reuse factor as integrated as far as possible. The experts do not see direct potential in further investments in CASE tool support. The experts did note that CASE tool support may be interesting for some cases, possibly in an even more narrow and steady domain. Based on the insights gained by the experts no recommendations are suggested for this reuse factor.

Communication tool support

The assessment of communication tool support resulted in an average score of 2,9 with an above average standard deviation of 1,5. The relevance of the reuse factor was 3,2 with a standard deviation of 0,7. An overview of the assessment results is presented in Table 44**.**

Factor: Communication tool support			
Score (0-10)		Relevance (0-5)	
Average:	2,9	Average:	3,2
Standard deviation:	1,5	Standard deviation:	0,7

TABLE 44: COMMUNICATION TOOL SUPPORT RESULTS

The experts already noted several times during the assessment process that communication mostly happens informally. In most cases a component owner is contacted when information is required about

that component. Based on the revision management system the consumers or potential producers are able to identify the specific component owner. The discussion also revealed that there are some tools used for communication support. These include the use of a bug tracking system, mailing lists and in some cases Wikipedia pages. An expert added to this that '*the thing that matters most is that communication is happening, the way in which it is communicated is not interesting*'. The bug tracking system is used to facilitate change requests and actually supports the 'configuration and change management' factor. In most cases the bug tracking system is used per project and not for individual components. The mailing list is a general lists, which includes all business units and is also for software development issues in general. An expert noted that *'the use of mailing lists can become rather extensive'*. And that *'when you are working on a component, you also know who else is working on it*'.

After discussion with the experts again the Wikipedia pages were mentioned as a possible way to further support the communication through the use of tools. Supporting the communication through another way was not considered as the most essential. It matters most that communication issues are communication.

When evaluating the results of this reuse factor the conclusion can be drawn that it is a valuable reuse factor to identify possible reuse communication issues. The reuse factor, however, is not end by itself. The experts mentioned several times that communication issues have to be addressed, but how they are addressed is a secondary concern.

Evaluating technology factors

The experts considered all the technology factors as relevant. The use of CASE tool support did score substantially lower, but the expert are still interested in its possibilities. The technology factors in general were also scored a bit lower than the other reuse factor. The primary reason for this seems to be that the technology factors are rather supportive. The basic technology has to be installed and the rest can be developed according to the needs of an organization. The most basic technology to be installed is a component repository.

5.3 Expert evaluation

After the application of the management tool the experts were asked to give their opinion and to fill in an evaluation form. In most cases they provided their train of thought here as well leading to valuable insights. The evaluation form is presented in Appendix N, the results are summarized in Appendix Q. This section elaborates on the results and presents the insights gained.

5.3.1 General evaluation

The general evaluation consists of a five general questions catching the first thoughts about the management tool.

The first question was if the management tool provides good guidance for analyzing the management of software reuse issues. The experts ranked the question with an average of 3,4. Interestingly the individuals who were actively working on solving some relevant management issues ranked the management tool higher than others who were less engaged with the subject. In general they stated that the management tool is a good way to capture 'the big picture'. It addresses all the management issues and provides a way to talk about these issues.

The second question was if the incremental steps are relevant for development with software reuse. At the start of the validation process the experts were explained the mapping between the CMMI maturity levels and the incremental stages of Griss. With a score of 4,1 this was perceived very well.

The third question was if the framework is easy to understand. This questions was ranked with a score of 3,2. At the time of validating the framework the experts missed some introduction sections and examples. The thought that they could fill in the framework with my assistance quite well, but still would like to see some additional information.

The fourth question was if the experts found the management tool easy to apply through the use of the assessment method. This question was ranked with a score of 3,7. In general the applicability was quite well, but the experts would like to spend more time on the management tool. In that sense 1 hour, which was used for the validation, was a bit short.

The fifth question assessed the first thoughts of the experts regarding the found results. The question asked whether the results are relevant for an on organization like this one. This was ranked with a score of 4,0. This question can be related back to the second question, about the incremental steps, as the experts found that the management tool in general provides a good way to assess relevant issues.

An overview of the results is presented in Figure 17.

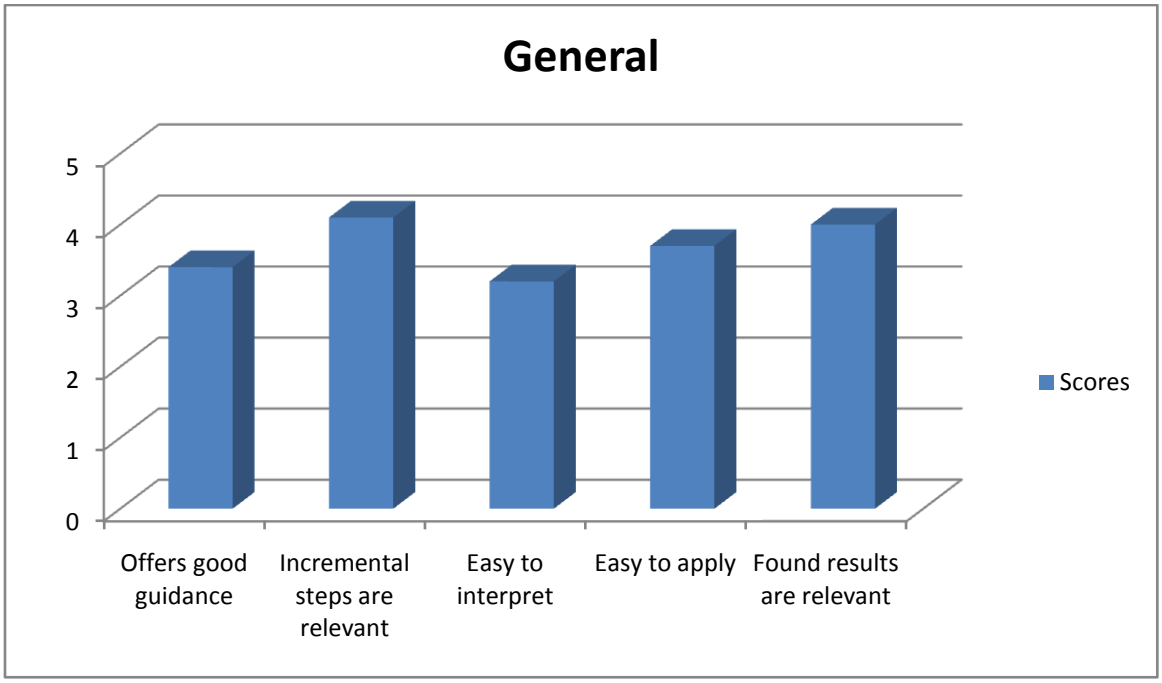


FIGURE 17: GENERAL EVALUATION RESULTS

5.3.2 Management tool elements

The second part of the evaluation form assessed the elements of the management tool, as taken into the framework. Although an evaluation was also done after the experts filled in the assessment method of the management tool, this part of the evaluation form provides another way to check the opinion of the experts.

The first questions assessed the maturity levels and its application in the management tool. This resulted in a score of 4,2. The maturity levels were not only perceived as relevant, but its application was also found to be suitable by the experts.

The second question assessed the reuse factors and its application in the management tool. The result was a score of 4,1. Indicating that the reuse factors were also received very well. This is in line with the results presented earlier where the experts did not directly miss any reuse factors. Some additional comments were provided though, e.g. the commercial aspects are not taken into the domain focus and the setting of priorities is always running on the background.

The third question assessed the various practices, which represent key aspects of reuse factors on a certain maturity stage. The score given by the experts is 4,0. In general they considered the use of practices a valuable way to assess the combination of maturity stages and reuse factors. Also the division of practice in three categories (approach, deployment and results) was perceived well. The quick readers liked the way of first reading the approach and then diving more in depth by reading the deployment and results columns of the assessment tool. For case organization the experts did find characteristics of multiple stages, likely because the general components are treated differently than the domain specific components. Furthermore the experts noted that some of the highest score will never be reached for the case organization, making some elements less applicable for them.

Although there are some comments the elements were received very well. An overview is presented in Figure 18

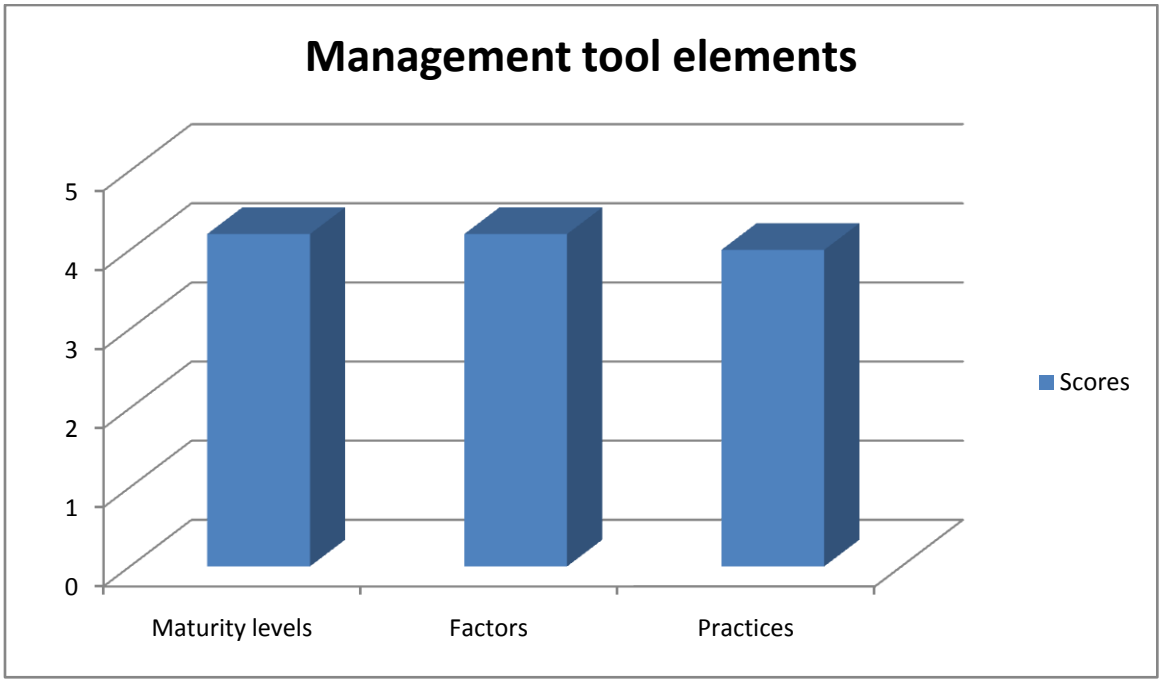


FIGURE 18: ELEMENT EVALUATION SCORES

5.3.3 Applicability

The final part of the evaluation form evaluates the applicability of the management tool through the use of the assessment methodology. Where the previous part focuses on the framework elements this part focuses on the assessment methodology.

The first questions evaluated the assessment method as a way to apply the framework. This was given a score of 4,0. Although this is almost exactly the same question as the fourth question of the general evaluation, the results are a bit higher. Still it provides an indication that in general the applicability was perceived to be quite good.

The second question was if the use of relevance was a good way to apply the management tool. This resulted in of a score of 3,5. The reason for this score is that the relevance variable was not always as clear as it was intended to. The experts noted that they sometimes found a factor very relevant, but do not wish to scale the factor, even if it had a low score. Based on this premise some experts would like to see another variable, which indicates that they want to scale a certain factor and possibly to which level they want to scale the factor. An additional variable would certainly be recommendable for improvement of the management tool.

The third question evaluates the use of a fixed score, based on the combination of approach, deployment and results. This resulted in a score of 4,2. The use of a fixed score provides a valuable way to operationalize the management tool. This input can be related back to the question whether a pre-defined or a customized assessment methodology would be best suitable (chapter 3). For a quick assessment the fixed score proved to be most useful. The experts did note that it was sometimes difficult to put something into a box, as characteristics of multiple boxes were found.

The fourth question evaluates the use of BTOPP elements. This resulted in a score of 4,1. The BTOPP elements provide a good way to structure the management issues. A comment by one of the experts suggests that a division of BTOPP elements according to strategic, tactical and operational levels would also provide a good way to assess the management issues. The train of thought provided by this expert goes as following: '*You need to create an environment for software reuse, this starts at the top, where the top management is creating an environment where individuals start to think about software reuse. This is later transferred down to the process levels. This is also why I find these*

factors more relevant than others'. Expanding the management tool with different levels is considered a valuable addition.

The fifth question checked whether the top 3 found by management tool matched with the top 3 based on their own insights. The score granted to this question is 3,0. Although the finding of the top 3 is based on a rule of thumb and the interpretation of the relevance factor it did match in about 50% of the cases. In the other cases the ordering of the top 3 was slightly different or the focus was on other factors. Sometimes the focus was on certain practices, instead of a factor itself. E.g. some developers would like to have additional resources and mentioned this as an important factor, but did not directly match it with top management support as a factor.

The sixth question checked if the application of the management tool led to new insights. This was important as the framework should provide guidance in different phases of an organization. The score added to this question was 3,6. In most cases the framework led to new insights, but sometimes also confirmed existing insights. No radical results were found at this point.

An overview of the scores is provided in Figure 19.

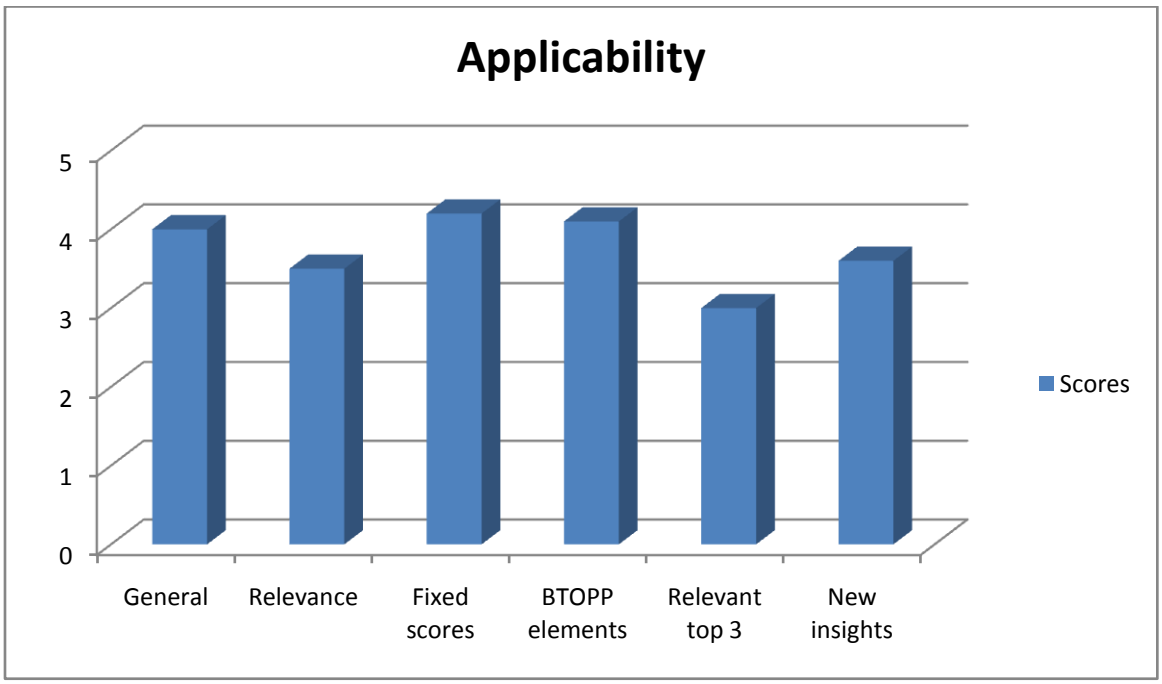


FIGURE 19: APPLICABILITY EVALUATION SCORE

5.4 Evaluating validation results

This section evaluates the validation results found during the various validation steps. Although the results are already presented in the previous sections they were not directly linked to the validation model as described in Chapter 4. An overview of the validation model and the results is presented in Table 45 and elaborated below.

Evaluation of validity		
Validity Factor	*Attribute*	*State of validation*
Completeness	All requirements are met	Confirmed
	All reuse factors are present	Confirmed
	No superfluous reuse factors	Confirmed
	Incremental stages are relevant	Confirmed
Internal consistency	Reuse factors are not overlapping	Confirmed
	Practices are consistent	Partially confirmed
Applicability	Ease of use	Partially confirmed
	Indicates relevant improvement areas	Partially confirmed

TABLE 45: EVALUATION OF VALIDITY

The requirements evaluation is satisfied as much as possible. In section 5.1 the requirements are mapped against the management tool after which the conclusion was drawn that all the requirements are satisfied. Most requirements as satisfied directly, the changing aspects of software reuse are addressed more indirectly by the reuse factor 'configuration and change management'. As such this part of the validity factor is confirmed.

The reuse factors were evaluated after each part of the application process described in section 5.2. Furthermore the total set of reuse factors was also evaluated in section 5.3, after which the conclusion was drawn that no reuse factors are missing. Several experts did mention other aspects, for example that the commercial aspects was not taken into the domain focus and that the setting of priorities within the project scope can lead to conflicting results for software reuse activities. In both cases the additional comments are considered as background information against which software reuse is presented. Software reuse always has a commercial aspect and the normal project priorities are always ranked higher, was also confirmed by the literature survey mentioned in Chapter 2. The completeness of the reuse factors is therefore considered confirmed.

During the application process as described in section 5.2 it appeared that all the reuse factors are relevant. Although the relevance of the individual factors is varying they are all considered as essential parts of the management tool. The low ranked reuse factors are expected to be able to score higher in different situations. Therefore, this part of the validity factor is also confirmed.

The defined incremental stage were used during the application process described in chapter 5.2 and evaluated by the experts in chapter 5.3. The experts noted that the incremental stages are logical stages and added an average score of 4,2 out of 5 indicating that the experts received the stages quite well. No improvements or comments were placed at the defined stages, so this part of the validity factor is confirmed as well.

During the application process as described in section 5.2 it appeared that all the reuse factors present a unique view on the concept of software reuse. Although there is some overlap between the reuse factors they all add value by analyzing the concept from their perspective. As such it is confirmed that there are no superfluous reuse factors.

The internal consistency of the reuse practices was partially confirmed. The validation of this attribute appeared to be more difficult than expected. In section 5.2 a high standard deviation emerged an indicator for difficulties with assessing the practices. Interestingly enough it appeared during the application process that there the set of REPA components is managed differently from the other sets of components. Therefore the experts saw characteristics of both sets in the reuse practices and had difficulties with scoring the reuse factor. The discussion with the experts revealed that the people factors might require additional research. Additional research can also be used to improve the practices of the other reuse factors. The experts did note that the practices are valuable for describing software reuse and granted a score of 4,0 out of 5 during the evaluation described in section 5.3. Based on these insights this point of validity is partially confirmed. Although the reuse factors are defined as such they are likely to be improved by additional research.

The ease of use of the management tool is partially confirmed. The experts noted that they could fill in the assessment tables with the additional explanation provided by the researcher. To make it more applicable they would like to see additional information, such as introduction sections and examples. The results from section 5.3 support the notion that the management tool is applicable, but also that this can be further improved.

Lastly, the outcomes of the framework were evaluated and matched with the expectations of the experts. In section 5.3 the relevance of the top 3 potential improvement areas was evaluated. The experts did not always find the expected top 3, resulting in a score of a 3,0. In that section it was also noted that the top 3 did not always meet the expectations, but often it did contains relevant areas as also mentioned according to the experts. It is difficult to determine whether the management tool has got the most relevant areas or the experts, but in both cases they try to address the software reuse issues and show great similarities in doing so. Using the outcomes of the management tool as input for the experts proved to be very valuable for discussing software reuse issues.

6. Recommendations

This chapter presents the recommendation based on the results presented in the previous section. The recommendations are divided in two parts, namely those who are meant for the improvement of the management tool and those who are meant for the improvement of software reuse practices at the case organization.

6.1 Recommendations for improving the management tool

This section discusses the recommendations for improving the management tool. Although the management tool was received quite well by the experts, there are also some suggestions for improvement. The improvement suggestions are based on observations made during the application by experts, the opinion of the experts and the suggestions made by the experts for future improvement. The recommendations are elaborated on in related sub-sections.

6.1.1 Organization of reuse factors

There are various options to organize the reuse factors. At the moment the reuse factors are organized around the BTOPP (business, technology, organization, process and people) model. One of the experts noted that the reuse factors can be rearranged according to organizational levels, varying from strategic to tactical and operational. When the reuse factors are organized around the organizational levels the order in which the reuse factors have to be addressed becomes visible. It most cases it is logical to first address the strategic levels, then the tactical levels and afterwards the operational levels. The expert also mentioned that there are likely many other ways to organize the reuse factors, but this can still be a point of improvement.

6.1.2 Additional variable

During the application process the experts mentioned that they sometimes found a reuse factor very relevant, but at the same time do not wish to scale the factor. In addition to the score and the relevance variable another variable is suggested, which measures the saleability of an individual factor. The experts did note that the scalability is indirectly addressed by the relevance variable, but at the same time want to see it more explicitly. The scalability of a factor is indirectly addressed, because a factor with a very low relevance does not require scaling and a factor with a higher relevant might require scaling. The additional variable can make the need to scale a factor more explicit.

6.1.3 Integrate documentation more explicitly

Several experts mentioned that documentation of reusable assets should be addressed more explicitly by the management tool. When looking back at the requirements of the management tool, one of them stated that the focus should be on code components, indicating that the management tool should not put the focus on documentation. It was, however, never the intention to leave out documentation completely. When looking back at the management tool itself, one can note that documentation is actually integrated in several reuse factors. These reuse factors are: ‘requirements management’, ‘communication channels and support’ and ‘repository support’. A requirements document is one of the documents that can be used for supporting reusable assets. The reuse factor describing requirements management addresses it from different perspectives, where requirements are management per project, per product or per product component. The same can be done for other documents such as test cases and functional designs. In that sense the reuse factor addressing requirements management can be extended or renamed to a more general name addressing all kinds of documents.

6.1.4 Extending the management tool with methods

One of the experts mentioned that tool support is covered by the technology factors, but that software reuse can also be supported by the use of methods. When looking back at the management tool one can note that certain reuse practices actually contain methods. The reuse factor ‘reuse planning’ contains for example the use of domain analysis as a method to systematically scan the domain for reuse opportunities. The use of methods can also be approached from another perspective, namely through the use of best practices. When the management tool is validated through the use of more case

studies it is likely that a set of best practices can be defined. The best practices can then be used to guide the scaling of reuse factors to new stages.

6.1.5 Improving individual factors

The experts mentioned that they sometimes had difficulties with the interpretation of individual factors. Additional documentation in the form of introductory sections or examples can be used to improve the understandability of individual reuse factors. Particularly difficult to interpret for the experts were the business factors and the people factors. The problem with the business factor 'domain analysis' is that it misses the commercial aspect of software reuse. The components are created based on the demands of the market and not the other way around. The people factors are operationalized using the People Capability Maturity Model (P-CMM) [84]. Although it provides a good way to analyse certain characteristics of the people factors, it is mentioned as more applicable for a larger organization. The organization of the people factors across the maturity levels likely needs a review. The only way which is considered suitable is to do multiple cases studies and evaluate the people factors in these case studies. The results have to be mapped against the maturity levels in order to further improve the framework.

6.2 Recommendations for the case organization

This section has been left out in the unrestricted version.

7. Discussion and conclusion

This chapter reflects on the results presented by this thesis. First of all a general conclusion is presented, which reflects on the research questions formulated in Chapter 1. After that the theoretical and practical contribution of this research is discussed. Then the points of improvements presented in the previous chapter are summarized. Lastly the limitations of this research and points for further research are elaborated on.

7.1 Conclusion

This thesis proposes a management tool for addressing software reuse issues. The management tool is created in such a way that it is generally applicable, with options to make it specific for the context of an individual environment. By doing so, the management tool is applicable in an agile development environment, but not limited to it.

The management tool is based on two main components, a reuse maturity model and an assessment method. The reuse maturity model consists of five maturity levels and fifteen reuse factors. The maturity levels are incremental plateaus for addressing software reuse. The reuse factors are factors influencing software reuse, quite similar to success factors. The reuse factors are organized in categories according to the BTOPP (business, technology, organization, process and people) model. Each combination of a reuse factor and a maturity level results in a set of relevant reuse practices. These reuse practices are the result of operationalizing the reuse factor and can be measured by the assessment method component. The assessment method component adds a score and a relevance to each reuse factor. The score represents the state of a reuse factor in a maturity level and the relevance variable serves as an indicator for the importance of a reuse factor in a specific context. By using the relevance variable an organization can make the management tool specific for their individual context, which can be one where agile development methodologies are utilized.

According to the validation by the expert panel the conclusion is drawn that the management tool is complete. The defined software reuse factors are a complete set of factors for addressing software reuse issues. Furthermore, the defined maturity levels are considered a good way to incrementally address software reuse. The assessment component of the management tool also proved its value during the application process, where the experts added a score and a relevance to each reuse factor. The experts provided some recommendations, among which the most important is adding an additional variable, for measuring the desired scalability of a reuse factor more explicitly.

The application of the management tool also led to some recommendations for the case organization. These include the management of domain specific components, the set up of teams, the use of Wikipedia pages for documentation and the support of new employees for engaging in reuse activities.

The results found through the application process and the opinions of the experts are in line with expectations of agile organizations. The software reuse processes are not standardized and the focus is on the skills and knowledge of individuals. The basis for software reuse is installed by creating an environment in which software reuse opportunities are identified and seized. The use of tool support plays an important role, but in the end the focus is on the sharing of knowledge itself. The use of CASE tool support was not directly identified as a potential factor, but may offer opportunities in the future.

Lastly, it appeared that the combination of the factor score and the factor relevance variable alone is not enough at the moment. During the assessment process by the management tool it is important to also gather contextual information, which can be done by asking the users of the management tool to give feedback on their train of thought. During the research this was done by asking them to think out loud and engage in discussion. The results were recorded on a voice recorder and processed later to provide a valuable background. Additional case studies should do the same in order to gather information for further improving the management tool.

7.2 Contributions

This research reveals some important practical and theoretical contributions. First of all this research combines existing insights regarding software reuse in a new way by proposing a software reuse management tool. Most elements of the management tool are, however, based on the insights gained by previous maturity models and frameworks and are therefore not new. The use of the assessment component, with a factor score and factor relevance, is the first attempt to really operationalize a management tool. The assessment component of the management tool contributes to both practice and literature. In practice it makes the method applicable and its results can be used to further validate the management tool. The application of the management tool can thus be used to extend the existing knowledge regarding software reuse. Furthermore, the management tool was applied in an agile development environment. Little to no existing literature is available which addresses software reuse in such settings.

7.3 Points of improvement

Although the management tool was received quite well by the experts, they also indicated several points of improvement. The points of improvement have been divided among five main areas, which are the following:

- The organization of reuse factors can be according to the steps in which they should be addressed and implemented.
- An additional variable should be added to measure the saleability of a reuse factor.
- The integration of documentation should be made more explicitly.
- The management tool can be extended with best practices for software reuse.
- Individual factors and related practices can be further improved.

The details regarding the improvement areas can be found in section 6.1 and are not further elaborated on in this section.

7.4 Limitations and further research

This study has several limitations that need to be discussed. First of all the validation process is based on quantitative data and is strongly colored by the perception of the researcher and the opinions of experts. It is therefore possible that there are some misconceptions within the research that have not been identified as such. The use of additional cases is required to further validate the results. If possible the use of qualitative data should be included in further research.

The use of additional case studies can lead to the identification of reuse patterns. The reuse patterns can lead to a catalogues which can serve as a guideline for organizations implementing software reuse programs. The factor relevance results of the assessment methodology can be used to identify such patterns. In certain situations a factor will be more relevant to scale than in other situations. In order to identify the reuse patterns the relevance of the reuse factors have to be linked to specific situations leading to the reuse patterns.

Another limitation is that the maturity component of the management tool assumes a mapping between reuse factors and maturity stages based on the CMMI-DEV model. Although there is research available that states CMMI level 5 can be reached in an agile development environment, it is not widely accepted. Further research is required to investigate the assumption that CMMI can indeed be successfully blended with agile practices in order to confirm that the management tool can indeed but applied in agile development environments.

The reuse practices, representing a combination of a reuse factor with a reuse maturity level, have to be further investigated. The use of additional case studies and related insights can be used to improve the description and the completeness of the reuse practices.

The opportunities of software reuse based on open source projects should be further investigated, as they seem to provide a valuable basis for software reuse in commercial organizations. Software reuse happens naturally in these projects, without monetary rewards or face-to-face communication.

Lastly the use of tool support should be further exploited. There are some tools available which support software reuse, but they limited address software reuse over multiple projects. Further investigation the collaboration mechanisms and the desired communication support can lead to improved tool support.

7.5 Recommendations for the case organization

This section has been left out in the unrestricted version

List of abbreviations

CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CMMI-DEV	Capability Maturity Model Integration for Development
CSF	Critical Success Factor
KPA	Key Process Area
RMM	Reuse Maturity Model
RCM	Reuse Capability Model
REPA	Codified term used in this restricted version
REPA component	General component which can be reused in multiple projects
REPA developer	Developer working on REPA components
KM	Knowledge management
KMS	Knowledge management system
BTOPP	Business, Technology, Organization, Process and People
SPI	Software Process Improvement
CBD	Component Based Development
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
ARC	Appraisal Requirement for CMMI

List of figures

List of tables

List of references

[1] P. S. W. Fong and C. W. C. Kwok, "Organizational Culture and Knowledge Management Success at Project and Organizational Levels in Contracting Firms," *Journal of Construction Engineering and Management*, vol. 135, pp. 1348-1356, 2009.

[2] M. Dowson, "The Ariane 5 software failure," *SIGSOFT Software Engineering Notes*, vol. 22, p. 84, 1997.

[3] B. Nuseibeh, "Ariane 5: Who Dunnit?," *IEEE Softw.*, vol. 14, pp. 15-16, 1997.

[4] M. Alavi and D. E. Leidner, "Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," *MIS Quarterly*, vol. 25, pp. 107-136, 2001.

[5] B. T. Keane and R. M. Mason, "On the Nature of Knowledge: Rethinking Popular Assumptions," presented at the Proceedings of the 39th Annual Hawaii International Conference on System Sciences, 2006.

[6] M. D. McIlroy, "Mass Produced Software Components," presented at the Software Engineering, Report on A Conference Sponsored by the NATO Science Committee, P. Naur, B. Randell (eds.), Garmisch, Germany, 7th to 11th October, 1968, NATO Science Committee, 1969.

[7] M. Morisio, *et al.*, "Success and Failure Factors in Software Reuse," *IEEE Trans. Softw. Eng.*, vol. 28, pp. 340-357, 2002.

[8] A. Mili, *et al.*, "An integrated cost model for software reuse," presented at the Proceedings of the 22nd international conference on Software engineering, Limerick, Ireland, 2000.

[9] W. B. Frakes and K. Kang, "Software reuse research: Status and future," *IEEE Transactions on Software Engineering*, vol. 31, pp. 529-536, 2005.

[10] D. Turk, *et al.*, "Limitations of Agile Software Processes," in *Third International Conference on eXtreme Programming and Agile Processes in Software Engineering*, 2002.

[11] M. Levy and O. Hazzan, "Knowledge management in practice: The case of agile software development," presented at the Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering, 2009.

[12] K. Beck. November). *Agile Manifesto*. Available: http:/agilemanifesto.org/

[13] I. Nonaka, "A Dynamic Theory of Organizational Knowledge Creation," *Organization Science*, vol. 5, pp. 14-37, 1994.

[14] T. J. Ellis and Y. Levy, "Towards a Framework of Problem-Based Research: A Guide for Novice Researchers on the development of a research-worthy problem," *Information Science Journal*, vol. 11, pp. 17-33, 2008.

[15] R. Wieringa, "Design science as nested problem solving," presented at the Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, Philadelphia, Pennsylvania, 2009.

[16] K. C. Desouza, *et al.*, "Four dynamics for bringing use back into software reuse," *Communications of the ACM*, vol. 49, pp. 96-100, Jan 2006.

[17] C. W. Holsapple and K. D. Joshi, "Knowledge Management: A Threefold Framework," in *Information Society* vol. 18, ed: Routledge, 2002, pp. 47-64.

[18] C. W. Krueger, "Software reuse," *ACM Computing surveys*, vol. 24, pp. 131-183, 1992.

[19] H. Mili, *et al.*, "Reusing software: issues and research directions," *Software Engineering, IEEE Transactions on*, vol. 21, pp. 528-562, 1995.

[20] M. L. Griss, "Software reuse: from library to factory," *IBM Syst. J.*, vol. 32, pp. 548-566, 1993.

[21] C. Atkinson and T. Kuhne, "Model-Driven Development: A Metamodeling Foundation," *IEEE Softw.*, vol. 20, pp. 36-41, 2003.

[22] J. Favre and T. Nguyen, "Towards a Megamodel to Model Software Evolution Through Transformations," *Electronic Notes in Theoretical Computer Science*, vol. 127, pp. 59-74, 2005.

[23] R. G. Lanergan and C. A. Grasso, "Software engineering with reusable designs and code," in *Software reusability: vol. 2, applications and experience*, ed: ACM, 1989, pp. 187-195.

[24] D. C. Rine and N. Nada, "An empirical study of a software reuse reference model," *Information and Software Technology*, vol. 42, pp. 47-65, 2000.

[25] B. A. Burton, *et al.*, "The Reusable Software Library," *Software, IEEE*, vol. 4, pp. 25-33, 1987.

[26] N. A. M. Maiden and A. G. Sutcliffe, "People-oriented software reuse: the very thought," in *Software Reusability, 1993. Proceedings Advances in Software Reuse., Selected Papers from the Second International Workshop on*, 1993, pp. 176-185.

[27] M. Griss. (1996, April). *Systematic Software Reuse: Architecture, Process and Organization are Crucial*. Available: http://martin.griss.com/pubs/fusion1.htm

[28] V. R. Basili, *et al.*, "How reuse influences productivity in object-oriented systems," *Commun. ACM*, vol. 39, pp. 104-116, 1996.

[29] R. W. Selby, "Empirically based analysis of failures in software systems," *IEEE Transactions on Reliability*, vol. 39, pp. 444-454, 1990.

[30] S. Henninger, "An evolutionary approach to constructing effective software reuse repositories," *ACM Transactions on Software Engineering and Methodology*, vol. 6, pp. 111-140, 1997.

[31] A. Tomer, *et al.*, "Evaluating software reuse alternatives: A model and its application to an industrial case study," *IEEE Transactions on Software Engineering*, vol. 30, pp. 601-612, 2004.

[32] J. A. Livermore, "Factors that impact implementing an agile software development methodology," *Journal of Software*, vol. 3, pp. 31-36, 2008.

[33] J. Highsmith and A. Cockburn, "Agile Software Development: The Business of Innovation," *Computer*, vol. 34, pp. 120-122, 2001.

[34] J. Highsmith and A. Cockburn, "Agile software development: the business of innovation," *Computer*, vol. 34, pp. 120-127, 2001.

[35] B. Boehm and R. Turner, "Management Challenges to Implementing Agile Processes in Traditional Development Organizations," *IEEE Softw.*, vol. 22, pp. 30-39, 2005.

[36] D. Robey, *et al.*, "Traditional, iterative, and component-based development: A social analysis of software development paradigms," *Information Technology and Management*, vol. 2, pp. 53-70, 2001.

[37] B. Boehm, "Get ready for agile methods, with care," *Computer*, vol. 35, pp. 64-69, 2002.

[38] S. Nerur, *et al.*, "Challenges of migrating to agile methodologies," *Commun. ACM*, vol. 48, pp. 72-78, 2005.

[39] B. Crawford, *et al.*, "Knowledge Management in Different Software Development Approaches," in *Advances in Information Systems*, ed, 2006, pp. 304-313.

[40] T. Chau, *et al.*, "Knowledge sharing: agile methods vs. Tayloristic methods," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, 2003, pp. 302-307.

[41] A. Cockburn and J. Highsmith, "Agile software development, the people factor," *Computer*, vol. 34, pp. 131-133, 2001.

[42] B. Ramesh, *et al.*, "Can distributed software development be agile?," *Commun. ACM*, vol. 49, pp. 41-46, 2006.

[43] D. Batra, "Modified agile practices for outsourced software projects," *Communications of the ACM*, vol. 52, 2009.

[44] S. W. Ambler, "The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments," *IBM*, 2009.

[45] W. Frakes and C. Terry, "Software reuse: Metrics and models," *ACM Computing Surveys*, vol. 28, pp. 415-435, 1996.

[46] V. C. Garcia, *et al.*, "Towards a maturity model for a reuse incremental adoption," presented at the SBCARS 2007 - Brazilian Symposium on Software Components, Architectures and Reuse, Brazil, 2007.

[47] R. Holibaugh, *et al.*, "Reuse: where to begin and why," presented at the Proceedings of the conference on Tri-Ada '89: Ada technology in context: application, development, and deployment, Pittsburgh, Pennsylvania, United States, 1989.

[48] P. Koltun and A. Hudson, "A reuse maturity model," in *4th Annual Workshop on Software Reuse*, Hemdon, Virginia: Center for Innovative Technology, 1991.

[49] R. Prieto-Díaz, "Making software reuse work: an implementation model," *SIGSOFT Softw. Eng. Notes*, vol. 16, pp. 61-68, 1991.

[50] M. J. Davis, "Stars Reuse Maturity Model: Guidelines for Reuse Strategy Formulation," in *Proceedings of the Fifth Annual Workshop on Institutionalizing Software Reuse*, Palo Alto, California, USA, 1992, pp. M 1-6.

[51] R. E. Creps, "The STARS Conceptual Framework for Reuse Processes," in *Proceedings of the Fifth annual Workshop on Institutionalizing Software Reuse*, Palo Alto, California, USA, 1992.

[52] M. Griss and W. Tracz, "WISR'92: Fifth Annual Workshop on Software Reuse: Working Group Reports," *SIGSOFT Softw. Eng. Notes*, vol. 18, pp. 74-85, 1993.

[53] T. Davis, "The reuse capability model: A basis for improving an organization's reuse capability," presented at the proceedings of the 2nd International Workshop on Software Reuse, Lucca, Italy, 1993.

[54] T. Davis, "Adopting a policy of reuse," *IEEE Spectrum*, vol. 31, pp. 44-48, 1994.

[55] S. Wartik and T. Davis, "A phased reuse adoption model," *Journal of Systems and Software*, vol. 46, pp. 13-23, 1999.

[56] SEI, "CMMI-DEV, V1.2 - Improving processes for better Products," Software Engineering Institute2006.

[57] M. Niazi, *et al.*, "A maturity model for the implementation of software process improvement: an empirical study," *J. Syst. Softw.*, vol. 74, pp. 155-172, 2005.

[58] M. K. Daskalantonakis, "Achieving Higher SEI Levels," *IEEE Softw.*, vol. 11, pp. 17-24, 1994.

[59] SEI, "CMMI-DEV, V1.2 - Improving processes for better Products," Software Engineering Institute 2006.

[60] M. A. Rothenberger, *et al.*, "Strategies for software reuse: A principal component analysis of reuse practices," *IEEE Transactions on Software Engineering*, vol. 29, pp. 825-837, 2003.

[61] CMMI-ProductTeam, "CMMI-DEV, V1.2 - Improving processes for better Products," Software Engineering Institute2006.

[62] D. Lucrédio, *et al.*, "Software reuse: The Brazilian industry scenario," *Journal of Systems and Software*, vol. 81, pp. 996-1013, 2008.

[63] M. S. Morton, *The Corporation of the 1990s: Information Technology and Organizational Transformation*: Oxford University Press, USA, 1991.

[64] S. D. Haes and W. V. Grembergen, "Information Technology Governance Best Practices in Belgian Organisations," presented at the Proceedings of the 39th Annual Hawaii International Conference on System Sciences - Volume 08, 2006.

[65] M. L. Griss, "Architecting for large-scale systematic component reuse," presented at the Proceedings of the 21st international conference on Software engineering, Los Angeles, California, United States, 1999.

[66] R. Joos, "Software Reuse at Motorola," *IEEE Softw.*, vol. 11, pp. 42-47, 1994.

[67] W. B. Frakes and C. J. Fox, "16 Questions About Software Reuse," *Communications of the ACM*, vol. 38, pp. 75-&, Jun 1995.

[68] S. Haefliger, *et al.*, "Code reuse in open source software," *Management Science*, vol. 54, pp. 180-193, 2008.

[69] R. D. Banker and R. J. Kauffman, "Reuse and productivity in integrated computer-aided software engineering: an empirical study," *MIS Quarterly*, vol. 15, pp. 375-401, 1991.

[70] D. Fafchamps, "Organizational Factors and Reuse," *IEEE Softw.*, vol. 11, pp. 31-41, 1994.

[71] J. Bosch, "Organizing for Software Product Lines," in *Software Architectures for Product Families*, ed, 2000, pp. 117-134.

[72] V. Traas and J. v. Hillegersberg, "The software component market on the internet current status and conditions for growth," *SIGSOFT Softw. Eng. Notes*, vol. 25, p. 114, 2000.

[73] T. Ravichandran and M. A. Rothenberger, "Software reuse strategies and component markets," *Commun. ACM*, vol. 46, pp. 109-114, 2003.

[74] J. S. Poulin, "Technical opinion: reuse: been there, done that," *Commun. ACM*, vol. 42, pp. 98-100, 1999.

[75] R. Prieto-Díaz, "Implemeting Facted Classification for Software Reuse," *Communications of the ACM*, vol. 34, pp. 88-97, May 1991.

[76] M. Morisio, *et al.*, "Success and failure factors in software reuse," *IEEE Transactions on Software Engineering*, vol. 28, pp. 340-357, 2002.

[77] S. Isoda, "Experience report on software reuse project: its structure, activities, and statistical results," presented at the Proceedings of the 14th international conference on Software engineering, Melbourne, Australia, 1992.

[78] S. Rosenbaum and B. d. Castel, "Managing software reuse - an experience report," presented at the Proceedings of the 17th international conference on Software engineering, Seattle, Washington, United States, 1995.

[79] G. Caldiera and V. R. Basili. (1991) Identifying and Qualifying Reusable Software Components. *Computer*. 61-70. Available on: http://doi.ieeecomputersociety.org/10.1109/2.67210

[80] V. R. Basili, *et al.*, "A reference architecture for the component factory," *ACM Trans. Softw. Eng. Methodol.*, vol. 1, pp. 53-80, 1992.

[81] D. C. Rine, "Success factors for software reuse that are applicable across domains and businesses," presented at the Proceedings of the 1997 ACM symposium on Applied computing, San Jose, California, United States, 1997.

[82] J. S. Poulin, "Integrated support for software reuse in Computer-Aided Software Engineering (CASE)," *SIGSOFT Softw. Eng. Notes*, vol. 18, pp. 75-82, 1993.

[83] M. A. Rothenberger, "Project-level reuse factors: Drivers for variation within software development environments," *Decision Sciences*, vol. 34, pp. 83-106, 2003.

[84] SEI, "Overview of the People Capability Maturity Model," Software Engineering Institute 1995.

[85] R. D. Banker, *et al.*, "Repository evaluation of software reuse," *IEEE Transactions on Software Engineering*, vol. 19, pp. 379-389, 1993.

[86] N. Y. Lee and C. R. Litecky, "An empirical study of software reuse with special attention to ada," *IEEE Transactions on Software Engineering*, vol. 23, pp. 537-549, 1997.

[87] R. D. Banker, *et al.*, "Automating output size and reuse metrics in a repository-based computer-aided software engineering (CASE) environment," *IEEE Transactions on Software Engineering*, vol. 20, pp. 169-187, 1994.

[88] SEI. (2001, 2010). *Standard CMMI Appraisal Method for Process IMprovement (SCAMPI), Version 1.1 Method Definition Document*. Available on: http://www.sei.cmu.edu/reports/01hb001.pdf

[89] SEI. (2006, April). *Standard CMMI Aprraisal Method for Process Improvement (SCAMPI) A, Versio 1.2: Method Definition Document*. Available: www.sei.cmu.edu/

[90] SEI. (2006, April). *Appraisal requirements for CMMI, Version 1.2 (ARC, v1.2)*. Available: http://www.sei.cmu.edu/reports/06tr011.pdf

[91] R. Lagerström, *et al.*, "Architecture analysis of enterprise systems modifiability: a metamodel for software change cost estimation," *Software Quality Journal*.

[92] R. Yin, *Case Study Research: Design and Methods, Third Edition, Applied Social Research Methods Series, Vol 5*: Sage Publications, Inc, 2003.

[93] R. B. Schwartz and M. C. Russo, "How to quickly find articles in the top IS journals," *Commun. ACM,* vol. 47, pp. 98-101, 2004.

[94] W. B. Frakes and C. J. Fox, "Quality improvement using a software reuse failure modes model," *IEEE Transactions on Software Engineering,* vol. 22, pp. 274-279, 1996.

[95] W. C. Lim, "Effects of reuse on quality, productivity, and economics," *IEEE Software,* vol. 11, pp. 23-30, 1994.

[96] P. Mohagheghi and R. Conradi, "An empirical investigation of software reuse benefits in a large telecom product," *ACM Transactions on Software Engineering and Methodology,* vol. 17, 2008.

[97] S. Purao and V. C. Storey, "Evaluating the adoption potential of design science efforts: The case of APSARA," *Decision Support Systems,* vol. 44, pp. 369-381, Jan 2008.

[98] D. C. Rine and R. M. Sonnemann, "Investments in reusable software. A study of software reuse investment success factors," *Journal of Systems and Software* vol. 41, pp. 17-32, 1998.

[99] R. W. Selby, "Enabling reuse-based software development of large-scale systems," *IEEE Transactions on Software Engineering,* vol. 31, pp. 495-510, 2005.

Appendix A – List of interviewees

This appendix provides an overview of the interviewees used to formulate the problem statement. The interview results are not elaborated in the appendix, but can be found throughout the thesis as also discussed in chapter 1.

Name	Function	Business unit
Person 1	Project Manager	1
Person 2	Software Engineer	1
Person 3	Software Engineer	1
Person 4	Information Architect	2
Person 5	Software Engineer	3
Person 6	Software Engineer and Architect	4

**The business unit names have been left out in the unrestricted version*

Appendix B – Interview guidelines

This appendix provides an overview of the interview guidelines, used to formulate the problem statement. Not all the questions are asked to all the interviewees, as some questions were already answered. These answers were so obvious that the questions were not repeatedly asked again, hence the name interview guidelines and not interview protocol.

Background variables

1. What is your job function?
2. How long do you do this job or a similar job?

Knowledge usage

3. Which knowledge resources do you use over multiple projects?
    a. Are these the REPA components or also domain specific components?
    b. Who is able to access the knowledge sources?
    c. How do you find these knowledge resources?
    d. Are they easy to find?
    e. Which tools are involved for storing and finding knowledge assets?
    f. How do you know where to look for knowledge assets?
    g. Is there sufficient documentation available to understand these knowledge resources?
    h. How do you get or did you get knowledge to understand the reusable asset otherwise?

Knowledge management

4. How do you manage these resources?
    a. Are there formal guidelines for managing knowledge resources?
    b. If so, are these guidelines available?
    c. Have roles been defined to manage these resources?
    d. Which tools support the exchange of knowledge artifacts?
    e. How successful are these tools?
    f. When are knowledge assets created?

Management of changes

5. How do you deal with changes?
    a. How do you know you have the latest version?
    b. How do you communicate changes? (e.g. notification?)
    c. How do you know who to contact for relevant changes?
    d. How do you get informed of changes yourself?
    e. Is everybody able to make changes?

Improvements

6. Are there any improvements which can be made?
7. How do you see the future for reusable components?

Appendix C – Top 25 IS journals

This appendix provides an overview of the top 25 journals as mentioned in the article of Schwartz and Russo [93]. Below the table the used databases are elaborated on.

Ranking of IS journals			
World rank	Journal	World rank	Journal
1	MIS Quarterly	14	Data Base
2	Communications of the ACM	15	Organization Science
3	IS Research	16	Information Systems Journal
4	Journal of MIS	17	Academy of Management Journal
5	Management Science	18	Communications of the AIS
6	IEEE Transactions (various)	19	IEEE Computer
7	Harvard Business Review	20	Journal of Strategic IS
8	Decision Sciences	21	Admin. Science Quarterly
9	Decision Support Systems	22	Academy of Management Review
10	Information and Management	23	International Journal of E-Commerce
11	European Journal of IS	24	ACM Computing Surveys
12	Sloan Management Review	25	Accounting, Management & IT
13	ACM Transactions (various)		

Schwartz and Russo compared 7 indexes for searching through the top 25 Information Systems (IS) journals [93]. The indexes Ingenta and ACM digital library cover the top 25 journals, but provide limited supported for exporting multiple results and abstracts. Several different combination are also possible to cover the top 25 IS journals. For this thesis Web of Knowledge, Scopus and the Communications of the AIS are used to cover the top 25 IS journals.

Appendix D – Included empirical studies

This Appendix provides an overview of the included empirical studies for identifying reuse factors.

Nr	Authors	Main results	Method	Context and details	Journal
1	Banker (1991) [69]	The article reports on productivity gains from implementing an IS strategy. Central within this strategy is the reuse of software and the integration of its processes.	Case study	First Boston Corporation (n = 21 software development projects)	MIS Quarterly
2	Banker (1993) [85]	Uses repository evaluation to study software reuse at two sites that are perusing reuse by means of the same CASE tool.	Case study	The First Boston Corperation (FBC) and and Carter Hawley Hale Information Services (CHH) (n = 2 cases)	IEEE Transactions on Software Engineering
3	Desouza (2006) [16]	Reporting on 4 findings regarding software reuse.	Interviews	25 software engineering companies, of which 12 were ranked in the fortune 100. (n > 100 interviews)	Communications of the ACM
4	Fafchamps (1994) [70]	Abstracted 4 models for managing software reuse programs.	Case study	HP development sites (n = 10 development sides)	IEEE Software
5	Frakes and Fox (1995) [67]	Answered 16 questions to software reuse. Defined process, education, asset quality, feasibility and type of industry are crucial.	Survey	Mainly US (n = 113 based on 23 US companies,1 EU company and 6 universities)	Communications of the ACM
6	Frakes and Fox (1996) [94]	Failure node analysis concluded that especially the understanding of components is trivial.	Question-naire	Mainly US (n = 113 based on 23 US companies,1 EU company and 6 universities)	IEEE Transactions on Software Engineering
7	Griss (1993) [20]	Systematic reuse requires improved processes, careful management and well-designed and packaged reusable software.	Experience report	HP	IBM Systems Journal
8	Griss (1998) [65]	Reporting on lessons learned based on 2 cases and more practical work done at AT&T and Netron.	Experience report	Based on research at HP and Ericsson, and practical experience at AT&T and Netron	Proceedings of the 21st international conference on Software engineering
9	Haefliger (2008) [68]	Investigates software reuse in the setting of open source projects.	Case study	Various open source projects (n = 6, 6 core samples based on a larger set of 15 samples)	Management science
10	Issoda (1995) [77]	Description of structure. Processes and technology. Concluded that it is primary a managerial issues.	Experience report	NTT Software Laboratories.	Proceedings of the 14th international conference on Software engineering
11	Joos (1995) [66]	Recommendation based on several reuse program attempts at Motorola	Experience report	Motorola	IEEE Software

Nr	Authors	Main results	Method	Context and details	Journal
12	Lee and Litecky (1997) [86]	Software reuse is mostly a management issue. Only a few technical factors.	Questio-nnaire	Random sample of Ada professionals (n = 75)	IEEE Transactions on Software Engineering
13	Lim (1994) [95]	Results of 2 reuse programs for economics measurements. Increased productivity, reduced time to market. Reuse can provide a substantial return on investment	Experience report	HP	IEEE Software
14	Lucredio et al. (2008) [62]	Identifications of factors that influence software reuse	Survey	Brazilian industry (n = 52)	The Journal of Systems and Software
15	Mohagheghi (2007) [96]	Studying reuse levels of a large telecom product.	Case study	Two Ericsson organizations	ACM Transactions on Software Engineering and Methodology
16	Morisio et al. (2002) [76]	Identification of success factors.	Interviews	European process improvement experience projects (PIE's) (n = 19 based on 19 companies covering 24 projects)	IEEE Transactions on Software Engineering
17	Prieto Diaz (1991) [75]	Reporting on implementing facetted classification.	Experience report	GTE data services	Communications of the ACM
18	Purao (2008) [97]	Evaluating the adaption of reuse-based design approaches.	Experiment	Demonstrating of a reuse-based design approach, after which a survey was held. (n =57 majoring computer science information system students)	Decision support systems
19	Rine (1997) [81]	Identified leading indicators for software reuse capability.	Survey	Mostly USA organizations. (n = 109. Respondents covered 99 projects and 83 organizations)	Journal of Systems and Software
20	Rine and Sonneman (1998) [98]	Identified 6 factors which strongly influence software reuse.	Mail survey	Mostly USA organizations. (n = 109. Respondents covered 99 projects and 83 organizations)	The Journal of Systems and Software
21	Rosenbaum (1995) [78]	Identified 7 factors that contribute to successful reuse.	Experience report	Schlumberger Austin Systems Center (ASC) supports	Proceedings of the 17th international conference on Software engineering
22	Rothenberger (2003a) [73]	Strategies for software reuse.	Survey	Random sample of companies related to popular reuse papers (n = 71)	IEEE Transactions on Software Engineering

Nr	Authors	Main results	Method	Context and details	Journal
23	Rothenberger (2003b) [83]	Identification of project-level reuse factors in the success of software reuse.	Case study	MBA Technologies (n= 5 projects and 1 NGT session with 9 members)	Decision sciences
24	Selby (2005) [99]	Analyzed reuse levels and factors in large scale projects. Suggests higher levels of reuse compared with smaller projects.	Case study	NASA Goddard Space Flight Center in Greenbelt, Maryland. (n = 25 repositories for software projects). Data mining was used	IEEE Transactions on Software Engineering

Appendix E – Overview reuse factors

This appendix presents an overview of the identified reuse factors. Below the last table the meaning of the contents of the table is explained.

	Banker (1991)	Banker (1993)	Desouza (2006)	Fafchamps (1994)	Frakes and Fox(1995)	Frakes and Fox(1996)	Griss (1993)	Griss (1998)	Haefliger (2008)	Isoda (1995)	Joos (1995)	Lee and Lictecky (1997)	Lucredio et al. (2008)	Morisio et al. (2002)	Prieto Diaz (1991)	Rine (1997)	Rine and Sonneman (1998)	Rosenbeaum (1995)	Rothenberger (2003a)	Rothenberger (2003b)
Product family approach / domain focus	-	P	-	P	-	P	P	P	-	P	P	P	P	P	-	P	P	-	P	P
Top management support	-	P	-	P	-	P	P	P	N	P	P	P	P	P	P	P	P	P	P	-
Reuse champion / sample solutions	-	-	-	P	-	-	P	-	-	-	P	-	-	-	-	-	-	P	-	-
Separate team structure	-	-	P	P	-	P	-	P	-	P	P	-	P	P	P	-	-	P	-	-
Different team structures	-	-	P	P	-	-	-	-	-	P	-	-	-	P	P	-	-	-	-	-
Key reuse roles / ownership	P	P	-	P	-	-	-	P	-	P	P	-	-	P	P	-	-	P	-	-
Development / programming experience (producer)	P	-	P	P	-	-	-	-	-	-	-	N	P	P	-	-	-	-	N	P
Reuse experience (consumer)	P	-	-	-	N	-	P	-	N	-	P	P	-	N	-	-	-	-	P	P
Project team reuse education / training	P	P	-	P	P	-	P	P	-	-	P	-	N	P	P	-	-	-	-	-
Rewards and incentives	P	P	-	P	N	P	P	P	P	P	P	-	-	N	P	-	-	-	-	-
Communication channels and support	-	-	-	P	-	P	-	-	P	P	P	-	-	P	P	-	-	P	-	P
Repository usage		P	-	-	N	P	P	-	P	-	P	P	N	N	P	P	-	P	-	-
Search and retrieval	P	P	-	-	N	N	-	-	P	P	P	-	-	N	-	P	-	-	-	-
CASE tool usage	P	P	-	-	N	-	-	-	N	-	P	-	P	-	-	P	-	-	-	-
Systematic / repeatable software reuse processes	P	-	-	P	P	-	P	P	-	P	P	-	P	P	-	P	-	-	P	-
Planning for reuse	P	-	-	P	-	-	-	-	-	-	P	-	-	-	-	-	-	-	P	-

	Banker (1991)	Banker (1993)	Desouza (2006)	Fafchamps (1994)	Frakes and Fox(1995)	Frakes and Fox(1996)	Griss (1993)	Griss (1998)	Haefliger (2008)	Isoda (1995)	Joos (1995)	Lee and Lictecky (1997)	Lucredio et al. (2008)	Morisio et al. (2002)	Prieto Diaz (1991)	Rine (1997)	Rine and Sonneman (1998)	Rosenbeaum (1995)	Rothenberger (2003a)	Rothenberger (2003b)
Kind of reused assets (more the better)	-	-	-	-	-	-	-	-	-	-	-	-	P	-	-	-	P	-	-	-
Origin reuse assets	P	P	-	-	P	P	-	-	-	-	-	-	P	-	-	P	-	P	-	-
Software reuse measurements		-	-	-	N	-	-	-	-	P	P	-	-	P	-	-	-	-	-	-
Quality model usage / ranking	-	-	-	P	N	-	-	-	P	-	-	-	P	-	P	P	-	P	-	-
Certification process / Legal issues	-	-	-	-	N	-	-	-	P	-	-	-	-	-	P	-	-	-	-	-
Configuration management of reusable assets	-	P	-	-	-	-	-	-	P	-	-	-	P	P	-	P	-	-	-	-
Manage change requests	-	-	-	P	-	-	-	-	P	-	-	-	-	P	-	-	-	P	-	-
Domain analysis	-	-	-	P	-	-	P	-	-	-	P	-	-	N	P	-	P	P	-	P

A “P” indicates positive influence. An “N” indicates that the factor has not been reported as a factor of influence. It does not mean that the factor negatively influence software reuse. A “-“ indicates that no relationship was found for the factor.

Appendix F – Mapping reuse factors to the BTOPP model

This appendix presents the mapping of the reuse factors to the BTOPP model.

BTOPP elements	Reuse factors
Business	Product family approach / domain focus
Organization	Top management support
	Reuse champion / sample solutions
	Separate team structure
	Different team structures
	Key reuse roles / ownership
	Development / program language experience
	Reuse experience
Process	Systematic / repeatable software reuse processes
	Planning for reuse
	Kind of reused assets (use of different types results in better reuse results)
	Software reuse measurements
	Quality model usage / ranking
	Certification process / Legal issues
	Configuration management of reusable assets
	Manage change requests
	Domain analysis
	Kind of reused assets (more the better)
	Origin reusable assets
People	Producer experience
	Consumer experience
Technology	Repository usage
	Search and retrieval
	CASE tool usage

Appendix G – Mapping the CMMI-DEV model to reuse factors

This appendix presents the mapping of the CMMI-DEV model to the reuse factors.

Mapping CMMI-DEV with Software Reuse literature					**(1)**
CMMI-DEV				**Software Reuse**	
Maturity Level	*Process Area*	*Acronym*	*Category*	*Relate reuse factor*	*Strength of mapping*
2	Configuration management	CM	Support	Manage change requests, Configuration management of reusable assets	Strong
2	Measurement and Analysis	MA	Support	Software reuse measurement	Strong
2	Project Monitoring and Control	PMC	Project management	Systematic reuse process, top management support	Average
2	Project Planning	PP	Project management	Systematic reuse process, top management support.	Average
2	Requirements Development	RD	Engineering	Requirements management	Strong
2	Process and Product Quality Assurance	PPQA	Support	Quality ranking model, certification.	Strong
2	Supplier Agreement Management	SAM	Project management	Quality ranking model, certification in the case of an external component market.	Average
3	Decision Analysis and Resolution	DAR	Support	Configuration management	Average
3	Integrated Project Management	IPM	Process management	Part of systematic reuse process, but not explicitly addressed	Weak
3	Organizational Process Definition	OPD	Process management	Part of systematic reuse process, but not explicitly addressed	Weak
3	Organizational Process Focus	OPF	Process management	Part of systematic reuse process, but not explicitly addressed	Weak
3	Organizational Training	OT	Process management	Reuse training and education	Strong

Mapping CMMI-DEV with Software Reuse literature (2)					
CMMI-DEV				**Software Reuse**	
Maturity Level	*Process Area*	*Acronym*	*Category*	*Relate reuse factor*	*Strength of mapping*
3	Product Integration	PI	Engineering	Domain analysis	Average
3	Requirements Development	RD	Engineering	N/A	Weak
3	Risk Management	RM	Project management	Part of domain analysis and reuse planning. Risk management itself has not been explicitly mentioned in literature.	Weak
3	Technical Solution	TS	Engineering	Origin of reusable assets, domain focus and part of systematic reuse processes	Average
3	Validation	VAL	Engineering	Validation can be considered as the fit within the architecture. This is related to the domain focus	Average
3	Verification	VER	Engineering	Quality model usage, but then also based on predefined requirements.	Average
4	Organizational Process Performance	OPP	Process management	Evaluations of aspects of the software reuse process.	Average
4	Quantitative Project Management	QPM	Project management	Evaluations of aspects of the software reuse process.	Average
5	Causal Analysis and Resolution	CAR	Support	Evaluations of aspects of the software reuse process.	Average
5	Organizational Innovation and Deployment	OID	Process management	Evaluations of aspects of the software reuse process.	Average

Appendix H – Mapping reuse factors to maturity stages

This appendix presents the mapping of the reuse factors to the maturity stages according to the three defined steps in section

Nr.	Maturity Stage	Reuse Factor	Allocated basis
1	Stage 1 and onward.	Domain focus	Incremental stages of Griss and insights gained by literature.
2	Stage 2 and onward.	Top management support	Incremental stages of Griss and insights literature.
3	Stage 2 and onward.	Reuse organization / responsibilities	Incremental stages of Griss and insights literature.
4	Stage 2 and onward.	Communication channels and support	CMMI-DEV and insights gained by literature
5	Stage 2 and onward.	Planning for reuse	CMMI-DEV
6	Stage 2 and onward.	Software reuse measurements	CMMI-DEV
7	Stage 2 and onward.	Requirements management	CMMI-DEV
8	Stage 3 and onward.	Quality management (internal)	CMMI-DEV
9	Stage 2 and onward.	Supplier management (external)	CMMI-DEV
10	Stage 2 and onward.	Configuration and change management	CMMI-DEV and incremental stages of Griss
11	Stage 1 and onward.	Producer experience	Insights literature
12	Stage 1 and onward.	Consumer experience	Insights literature
13	Stage 2 and onward.	Repository support	Incremental stages of Griss and insights literature
14	Stage 2 and onward.	Case tool support	Incremental stages of Griss and insights literature
15	Stage 2 and onward.	Communication support	Incremental stages of Griss and insights literature.

Appendix I – Organizational models

This appendix summarizes the advantages and disadvantages of the organizational models proposed by Fafchamps and Bosch [70-71].

Models of Bosch [71]			
Model	***Applicability***	***Advantage(s)***	***Disadvantage(s)***
Development department	Relatively small organizations (< 30 staff members)	Simplicity and ease of communication No organizational change required	Not scalable Particular types of engineering get more attention than others
Business unit	Medium organization (between 30 till 100 staff members)	Effective sharing (access to assets and evolution) Better saleable	No entity or explicit incentive to focus on shared assets Leads slowly to erosion of product line architecture Timely and reliable evolution relies on individuals
Domain engineering unit	Large (> 100 staff members)	Communication overload reduced to 1 to n Controlled evolution by the domain engineering unit. Scalable is higher than previous mentioned models	Managing flow of requirements Delays in implementation
Hierarchical domain engineering units	When the variability of system is large or very large and considerable numbers of staff members, i.e. hundreds, are involved	Capable of dealing with complex product lines Maximum scalability	Large overhead

Models of Fafchamps [70]			
Model	***Applicability***	***Advantage(s)***	***Disadvantage(s)***
Lone producer	Two or more teams	Networking opportunities Broad knowledge base Product delivery	Communication overload Priority setting Work overload Multiple reporting lines No home team
Nested producer	Not.	A home team Product delivery	No specialization Isolation from reuse peers Double reporting Negotiation with hardware manager Dispatched to other task Resource diverted
Pool producer	Works best when a reuse program has limited scope, such as sharing of a few selected components common to the products of a small number of teams.	No changes to organizational structure	Communication overload Unresolved conflicts Priorities of the different marketing lines Different time-to-market Different priorities for R&D and marketing Documentation
Team producer	Applicable or successful long-term reuse strategies.	One manager Team membership Control over resources Specialization opportunities Flexibility Ownership Knowledge about other projects	On call to consumers Work with too many projects No complete projects Communication overhead Maintenance overhead

Appendix J – Expert panel

Name	Function	Business unit *	Age	Years of relevant work experience
Person 1	Analyst / project leader	1	28	3
Person 2	Software Engineer	1	30	6,5
Person 3	Software Engineer	2	28	4
Person 4	Software Engineer / Project leader	3	28	5,5
Person 5	Software Engineer	3	32	9
Person 6	Software Engineer / Analyst	4	30	3
Person 7	Software Engineer	1	27	3
Person 8	Project leader	4	28	2,5
Person 9	Information architect	2	27	4

**The name of the business units is left out in this restricted version*

Appendix K – Validation procedure

This appendix describes the exact validation procedure maintained for the validation. A time indication is added as the appointments are meant to last 1 hour. The application and evaluation process was recorded on a voice recorder.

Preparation (10-30 minutes)

All the expert received a document in advance with general information about the management tool. The management tool itself was also added as an attachment. In the email sent to them the importance of the validation was stated and some information about the validation procedure was provided. Namely, that they were expected to apply the management tool during the meeting.

General explanation (5-10 minutes)

During the first 10 minutes of the meeting the experts were asked if they understood where the management tool was about and a brief explanation followed to make sure they knew. The explanation briefly addressed the set up of the tool in terms of factors and maturity stages. The goal of the management tool was addressed and the procedure of filling in the management tool.

Filling in the management tool (25-35 minutes)

The experts were asked to add a score to the reuse factors. After they have done that they were asked to add a relevance to the score. This was an indicator for how important they perceived the factor for software reuse, mostly biased on their experience within the company. The experts were asked to provide insights in their train of thought. Some of them talked loudly when filling in the assessment method, others were asked to explain their scores and why they considered it as relevant.

Evaluating the management tool (5-10 minutes)

After the experts filled in the management tool they were asked if they considered all the factors as relevant, or that they missed certain factors. The experts were also asked if they had any remarks at all and if they found that the tool was a logical way to fill it in. The exact way of filling in a factor for the five maturity stage was not evaluated directly, as this would take way too much time.

Evaluating the results (5 -10 minutes)

After evaluating the tool itself a closer look was taken at the results. By filling in the scores and combining them with the relevance a top 3 of weak points was identified. A common rule of thumb was maintained for identifying weak points. Where:

- Relevance is leading, not score.
- A weak point has a score lower than 5 (everything above 5 is considered as strong).
- The ordering of weak points is according to the relevance.

This means that a factor with a score of 4 and a relevance of 5 is considered as a weaker point than a factor with a score of 2 and a relevance of 4.

A top 3 of weak points was evaluated and discussed. Where the top 3 was not representative for the thoughts of the expert his own top 3 was asked. Because relevant insights were gained when filling in the management tool this step could be done rather quickly.

Evaluating the results (5-10 minutes)

After the experts provided their insights, they were asked to fill in an evaluation form. This form again evaluates certain elements of the model and the perceived usefulness on a scale from 1-5.

Appendix L – Documents for validation: background

This document contains a brief introduction of my research project with relevant background information about the proposed management tool.

Introduction

This organization frequently reuses existing knowledge when developing new solutions. Through knowledge reuse the organization is able to develop application faster, increase product quality and reduce time to market. The reuse of knowledge is manifested in code components. Within this organization several sets of components have been identified. When the domain is becoming narrower defined and solutions are becoming more mature it is expected that higher reuse levels can be achieved. By achieving higher reuse levels the benefits can be exploited even further.

With this ongoing trend several questions are asked, such as how do we achieve higher reuse level? What are these higher reuse levels actually and who should be responsible for managing reusable assets?

Research question

This research attempts to answer several of these question through the formulation of the following research question:

How to address software reuse issues through the use of a management tool?

The proposed management tool needs to be applicable in agile development organizations. Besides that the tool should address several reuse levels and relevant aspects of software reuse.

Software reuse

Software reuse is well defined as a two-fold concept. First of all it is about '*building software that is reusable by design*' and secondly it is about '*building with reusable software*'. This two-fold definition explicitly indicates that software reuse it not pure a technical issue, but also an organizational one.

While there are numerous advantages for the reuse of software assets it doesn't happen by itself. Investments have to be made to achieve higher levels of reuse. These investments do not always results in the expected gains and achieving higher reuse levels also depends of other factors such as the domain focus and the maturity of existing solutions.

Result

Literature describes an increasingly systematic becoming process for reaching higher levels of reuse. Beside that literature also describes several approaches such at the 'library approach' and the 'product line approach'. The elements found are combined into a reuse maturity model and operationalized through an assessment method. The result is a reuse management tool for addressing software reuse issues.

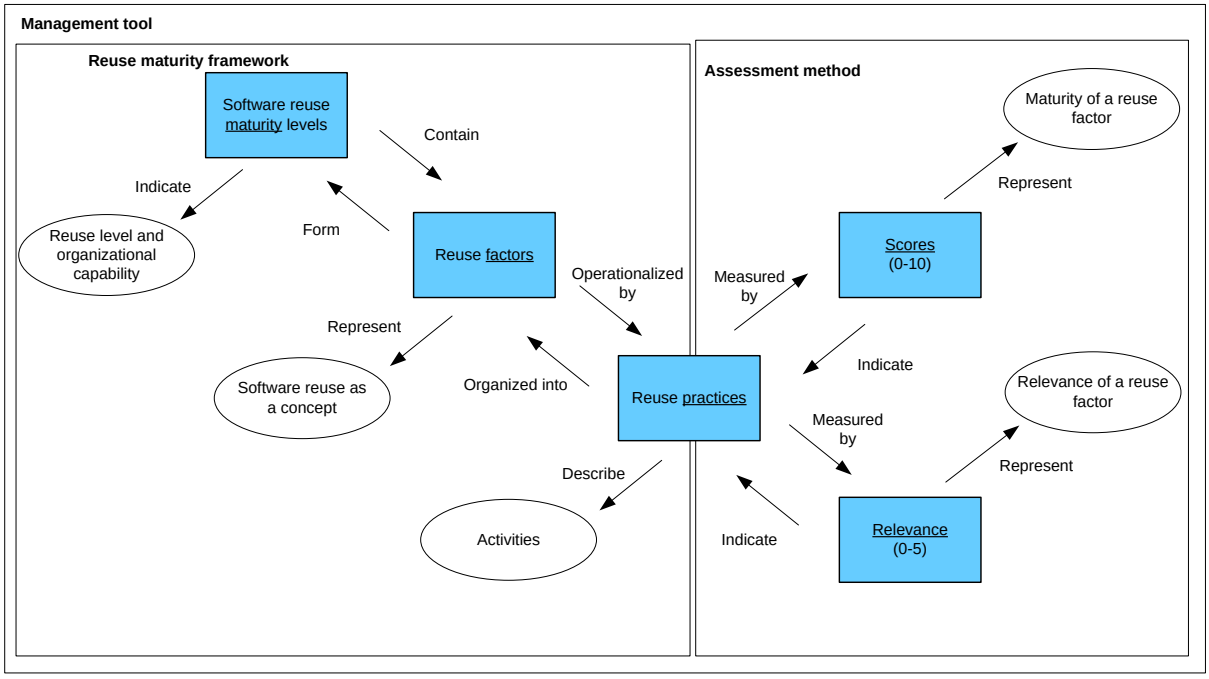
Reuse maturity model

The management tool provides the structure and the mechanisms for addressing software reuse issues. Although the tool was intended to be specific for typical agile development environments, the decision was made not to do this, as agile development methodologies can be successfully applied in more complex environments.

The management tool is based on two main components, a reuse maturity model and an assessment method. The reuse maturity model consists of five maturity levels and fifteen reuse factors. The maturity levels are incremental plateaus for addressing software reuse. The reuse factors are factors

describing relevant aspects of software reuse. The reuse factors are allocated incrementally to the maturity levels. In order to provide the user of the management tool structure, the reuse factors are allocated

The management tool is based on two main components, a reuse maturity model and an assessment method. The reuse maturity model consists of five maturity levels and fifteen reuse factors. The maturity levels are incremental plateaus for addressing software reuse. The reuse factors are factors described in literature as factors influencing software reuse, quite similar to success factors. In order provide the users mechanisms to structure the reuse factors they are organized in categories according to the BTOPP (business, technology, organization, process and people) model. Each combination, of a reuse factor and a maturity level, is represented by a set of reuse practices. These reuse practices are the result of operationalizing a reuse factor and can be measured by the assessment method component. The assessment method component adds a score and a relevance to each reuse factor. The score represents the state of a reuse factor in a maturity level and the relevance variable serves as an indicator for the importance of a reuse factor in a specific context. By using the relevance variable an organization can make the management tool specific for their individual context. The assessment method component can be used to indicate strong and weak points of an organization. An overview of the components is presented below.



Maturity levels

The maturity levels are predefined incremental plateaus for addressing software reuse. An overview of the 5 maturity levels is provided below.

Level	Description
0 - None	Indicates no reuse activities at all. Both the company and its individual members have not become aware of reuse opportunities yet. While this can be a possible situation, it will almost never be the case.

1 – Ad-hoc	Reuse opportunities are seized by individual members in an ad-hoc manner. The organization itself has not become aware of reuse opportunities yet. The project members that are reusing existing pieces of code are doing so mostly based on experiences gained in previous projects. Level 1 reuse does not only contain pieces of code, but can also include knowledge about architecture designs are code patterns.
2 – Managed reuse of leveraged code	Reuse opportunities have become aware to the company and are supported. Pieces of code are leveraged and may be used as black-box or white-box components. Both approaches have their advantages and disadvantages. Various elements of a systematic reuse process are instantiated to manage reuse activities, including configuration management and reuse measurement.
3 – Defined reuse of work products	Reuse opportunities are seized through a defined process. A separate team is responsible for the management of reusable assets and guide these assets through their evolution. Additional elements of a systematic reuse process are added, which include quality management.
4 – Quantitatively managed architecture reuse	Reuse opportunities are quantitatively managed. The reuse process is systematically evaluated even as the architecture. The reusable assets have to fit within the defined architecture, optimizing the use of reusable assets in a certain domain.
5 – Domain optimizing reuse	The final stage consists of optimizing all the reuse opportunities for a domain. Not only do the reusable assets have to fit within the architecture, but also is the architecture leading the development of reusable assets. Each new product is composed of, or created as reusable asset. Individual components are continuously evaluated and further optimized.

The maturity levels are represented by scores in the assessment method. By dividing the score by 2 the related maturity level can be found.

Reuse factors

Within the management tool 15 reuse factors have been identified based on a systematic literature review. A brief description is provided below.

Nr.	BTOPP element	Reuse factor	Brief description and related factors
1.	Business	Domain focus	The domain focus is used as an indication for the level of similarities among products in a certain domain. The domain is the environment in which a software development organization is operating and producing solutions for. The domain focus is also referred to as the product family approach and it includes implicitly the origin of reusable assets.
2.	Organization	Top management support and instrumental mechanisms	Top management support is taken as an indicator for the level of top management support. Top management support can be both passive and active. Top management can utilize instrumental mechanisms to promote or enforce desired behaviour. The instrumental mechanisms are a combination of other reuse factors, including the use of reuse champions, sample solutions, rewards and incentives, reuse education and training.

3.	Organization	Organizational structure and reuse roles	The organizational structure for software reuse elaborates on the separation of producers and consumers of reusable assets. The producers or reusable assets create the components, where the consumers use the reusable assets in building new applications. Beside the consumers and producers also several other reuse roles are identified.
4.	Organization	Communication channels and organizational support	The identification of communications channels and organizational support is also identified as a reuse factor. This factor is embodied in a systematic software reuse process and includes the communication of change requests. The factor is emphasized by agile methodologies, as stated by one of the requirements set up in the previous chapter.
5.	Process	Reuse planning	Reuse planning is an indicator of a systematic software reuse process. It also includes the reuse factor which describes origin of a reusable asset. Domain analysis may be deployed to perform systematic reuse planning. Domain analysis is a tool which evaluates the domain for potential reuse opportunities.
6.	Process	Reuse measurement and cost models	The measurement of reuse activities is also an indicator of a systematic software reuse process. When the reuse activities are measured it is possible to link them to cost models. The result is that systematic cost benefit analysis of individual components can take place.
7.	Process	Requirements management	Requirements management is also an indicator of a systematic software reuse process. Managing the requirements over multiple projects can provide a good basis for identifying their similarities.
8.	Process	Quality management (internal)	Quality management is once again an indicator of a systematic reuse process, but this indicator was also explicitly mentioned as a reuse factor in several papers. The use of quality models and ranking systems can provide the basis for quality management.
9.	Process	Supplier management (external)	Supplier management is also related to a systematic reuse process, but then with a focus towards the outside world. Supplier management deals with the acquisition of external assets, e.g. from black-box components markets. Related reuse factors include legal issues and the use of certifications.
10.	Process	Configuration and change management	The last aspect of a systematic reuse process is embodied in configuration and change management. This reuse factor is not only an aspect of a systematic reuse process, but both configuration and change management are recognized as separate reuse factors in literature. In this thesis they are again combined, as they show great similarities when they are operationalized.
11.	People	Producer skills and experience	Producer skills and experience relates to all the knowledge and practices which have to present at the producer side. This includes development and programming experience. A certain amount of domain knowledge is also required to be able to estimate what is project specific for a component and what can be made generic.

12.	People	Consumer skills and experience	Consumer skills and experience relates to all the knowledge and practices deployed at the consumer side. This includes knowledge of the reusable assets and best practices for integrating them into new applications.
13.	Technology	Repository support	Repository supports refers to the use of a repository for software reuse. It also includes the search and retrieval factors and the use configuration management tools.
14.	Technology	CASE tool support	Computer Aided Software Engineering (CASE) tool support refers to the use of additional programming tools. Effective software reuse provides tools where the repository is integrated in programming tools for daily use, reducing barriers to software reuse. By further integrating CASE tools with the repository the idea of an application generator can be realized.
15.	Technology	Communication tool support	Communication tool support addresses the need to support communication through the use of tools. These tools actively facilitate communication about reusable assets. E.g. the changes in a component or the latest developments. Also, the use of discussion forums or mailing lists can be considered communication tool support. This tool support differs from the other tools because they are passive. A user can for example populate a change on repository, but the other users are not aware of it until they check out a component and note a difference. Communication tool support is a reuse factor added based on the requirements formulated earlier, the literature review did not directly indicate this as a reuse factor.

In the assessment method the factors are organized around elements of the BTOPP (Business, Technology, Organizational, Process and People) model. This organization should provide the user of the tool mechanisms to identify strong and wear areas. The assessment method describe per combination of a reuse factor and a maturity level a set of practices. The practices are not further elaborated on in this document and can be found in the assessment method document.

Appendix M – Documents for validation: assessment method

Purpose:

The purpose of the management tool is to identify strong and weak points of an organization related to the management of software reuse issues. During the assessment process different levels of reuse are taken into account. Not every organization will be able to or desire to achieve a high reuse level.

Application procedure:

1. Determine the score per reuse factor

First of all read the column 'approach' in order to determine what comes close to the current situation. After that read the other two columns 'deployment' and 'results' to determine if this really is the one coming closest to the current situation. Based on the outcomes go 1 level up or 1 level down. In case the reuse factor shows characteristics of two reuse levels the choice can be made to add an odd score.

2. Determine the relevance per reuse factor

Determine the relevance per reuse factor. The relevance can be low (score of 1), relevant but not critical (score of 3) or critical (score of 5). In case the relevance shows characteristics of both levels an odd score can be added, similar to first step of the application procedure. If the reuse factor is not considered to be relevant at all a score of 0 can be added to the relevance. This means that the reuse factor should be removed from the framework.

3. Determine strong and weak points

The identification of strong and weak points follows from the analysis of the added scores and relevance variables. A high score can indicate a strong factor where a low score can be a possible point of improvement depending on its relevance. The BTOPP (Business, Technology, Organization, Process en People) model can be used to identify relevant improvement areas.

1. Business factor: domain focus

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

1. Business - Domain Focus for reuse			***Score:***
Points	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- No defined focus. - Domain is new to the organization.	-	-
Weak (2)	- No defined focus. - Domain is relatively new.	- Individuals transfer knowledge based on previous projects. - Reuse remains rather invisible.	- Reuse of features created by individuals in previous projects.
Fair (4)	- Some similarities across multiple projects in possible multiple domains.	- Previous knowledge is recognized as potential for reuse opportunities. - Pieces of knowledge are leveraged outside the project scope.	- Reuse of common features.
Good (6)	- Moderate levels of similarities across products in the same domain.	- Domain analysis is performed to analyse the level of similarities in the domain.	- Reuse of domain specific features and functionality.
Excellent (8)	- High levels of commonalities in a narrow domain led by a common architecture.	- Development of a common architecture. - Use of domain experts. Building applications using the architecture.	- Product families are created. Reuse is based on the product family.
Outstanding (10)	- Domain is architected and optimized.	- The common architecture is evaluated and optimized. - The common architecture is leading development activities.	- Development activities are based on optimizing the common framework. - All new development activities are in the form of reusable assets.

1. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

2. Organizational factor: top management support and instrumental mechanisms

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

2. Organizational – Top management support and instrumental mechanisms			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployed practices*	*Results*
Poor (0)	- Management does not support reuse activities.	-	-
Weak (2)	- Management does not support reuse activities, but reuse events occur by individuals.	- Individuals recognize reuse opportunities for personal benefits, e.g. less working pressure or time constraints.	- Initial reuse events occur, but remain rather invisible to the organization.
Fair (4)	- Reuse activities are encouraged and stimulated. - - Investments are still kept rather low.	- Top managements supports reuse champions, which are key producers of reusable assets. - Examples are used to stimulate reuse activities. - Rewards and incentivises are adjusted to stimulate reuse activities.	- Reuse activities are addressed and stimulated, but in practice cover only a small amount of people.
Good (6)	- Reuse is encouraged and dedicated resources are made available.	- Dedicated resources are made available for reuse activities. - Reusing work products is highly encouraged at design level, e.g. by frequently asking questions.	- Reuse events are focussed on high potential reuse areas.
Excellent (8)	- Total management commitment to software reuse. - Reuse activities are enforced by top management.	- Instead of stimulating and supporting reuse activities, not reusing is considered as bad behaviour in this stage. - Employee performance is judged on working according to the common architecture.	- Applications are built according to product family architectures. - Reusable components provide the basis building blocks.
Outstanding (10)	- total management commitment - striving for organizational excellence	All activities are executed in the form of reusable assets. Reuse is the way of building applications.	Reuse is integrated in the culture.

2. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

3. Organizational factor: organizational structure and reuse roles

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

3. Organizational – Organizational structure and reuse roles			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployed practices*	*Results*
Poor (0)	- Reuse activities are individual if recognized at all.	-	
Weak (2)	- Reuse activities are mostly individual.	- Reuse events occur based on previous project experience. - No reuse roles and responsibilities have been defined	- Reuse events occur ad-hoc and benefits are only visible for individuals or a selected group of people.
Fair (4)	- Informal reuse teams are integrated into the organization.	- Experienced developers are assigned the role of component owner for guiding the development of reusable assets. - Component owners meet regularly to exchange knowledge.	- The code owner is leading the development of reusable assets. - Component owners are also stimulating reuse activities through sharing their knowledge.
Good (6)	- An extended reuse team or community is responsible for managing work products.	- Code owners are supported by partners. Partners are representatives of groups working with the reusable assets. Together they form a reuse team. This can be both formal as informal.	- The development of reusable assets is a shared responsibility, where the code owner is leading the development process.
Excellent (8)	- Reuse team are set up according to the needs of the architecture.	- Code owners and partners have to ensure that components fit within the architecture. Additional roles may be defined to create and guard the architecture.	- A common architecture is leading the development process. - Reuse is the result of dedicated roles and shared responsibilities.
Outstanding (10)	- Consumer and producer teams are separated.	- A strong product line approach is maintained. - Consumers and producers are completely separated.	- All assets are produced as reusable assets.

3. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

4. Organizational factor: Communication channels and support

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

4. Organizational – Communication channels and support			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- Reuse activities are not communicated.	-	-
Weak (2)	- Reuse activities are not or informally communicated.	- Individuals share previous project experience.	- Ad-hoc way of communicating reuse activities.
Fair (4)	- Informal use of communication channels.	- Reuse issues are communicated by code owners to relevant parties. - Code owners have meetings on regular basis.	- Unsystematic flow of information, user has to select what is interesting.
Good (6)	- Communications channels are set up and used regularly.	- Tools and channels such as mailing lists or discussion forums are used to communicate changes to relevant parties.	- Regular flow of communication. - Use of communication channels depends on the sender.
Excellent (8)	- Communication channels are set up and enforced by top management.	- All reuse issues are centrally communicated. - Users can mine communication flows.	- Systematic flow of communication. - User has the risk to become overwhelmed with information.
Outstanding (10)	- Communication channels are optimized and possible integrated with development tools.	- All issues are communicated. - User has tools available to mine the stream of information for relevant issues.	- Systematic flow of communication optimized from the perspective of the receiver.

4. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

5. Process factor: Reuse planning

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

5. Process – Reuse planning			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- Reuse events are not planned.	-	-
Weak (2)	- Reuse is viewed as a single point opportunity.	- No plan is deployed, reuse events occur ad-hoc when experienced developers are allocated to new projects.	- Reuse is based on previous experience
Fair (4)	- Reuse is viewed as a way of leveraging projects, based on known similarities.	- Previous knowledge is used at design level. - Additional resources are made available to create reusable components before, during or after the project.	- Reuse activities are planned per project based on known similarities.
Good (6)	- Reuse is planned by identifying high potential reuse areas and related work products.	- Domain analysis is performed to systematically scan for high potential reuse opportunities.	- Reuse activities are planned based on systematically identified high potential areas.
Excellent (8)	- Reuse is planned according to a common architecture.	- Domain analysis and a common architecture are used to guide to plan reusable assets. - Reuse planning is used to compare the actual outputs versus the planned outputs.	- Reuse activities are planned based on systematic analysis and use of a common architecture.
Outstanding (10)	- Reuse is a business imperative and plans are frequently analysed and optimized.	- Reuse plans are part of the strategic plan. - Plans are frequently created and optimized to maximise reuse results for a given domain.	- Planning for reuse is part of the strategic plan, responsible for business success.

5. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

6. Process factor: Reuse measurement and cost justification

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

6. Process – Reuse measurement and cost justification			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- No reuse measurement or cost justification.	-	-
Weak (2)	- Reuse measurement and cost justification is individualized, if present at all.	- Individuals are able to estimate their reuse activities. - Cost justification is done based on personal gains such as saved time and reduced workload.	- No results are visible for the organization. - Reuse opportunities are seized based on personal gains.
Fair (4)	- Reuse events can be tracked back, but cost estimations are still opportunistic.	- Reuse events are registered or can be tracked back manually. - Cost models are not present at organizational level, but may be used by certain projects are people.	- Reuse measurement is possible for the organization, but does not occur systematically. - Cost models are limited addressed.
Good (6)	- Recent events can be tracked back and cost models are available.	- Reuse metrics are installed and reuse activities are transparent. - The actual costs related to reuse still have to be analyzed manually.	- Reuse events are visible for the organization. - Cost estimations are only executed when needed.
Excellent (8)	- Reuse metrics are installed and cost models are present, used systematically	- Reuse metrics and cost models are present. - Analysis is performed for expected pay offs.	- Costs and savings of reusable assets are known within the organization.
Outstanding (10)	- Reuse investments are estimated and are optimized	- Reuse investments can be estimated and investments are optimized. - Tools are adjusted for supporting reuse registration.	- Costs and savings of a product line are known and shared within the organization.

6. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

7. Process factor: Requirements management

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

7. Process – Requirements management for software reuse			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- No requirements management for software reuse.	-	-
Weak (2)	- Similar requirements are recognized by individuals.	- Individuals participate in multiple projects in the same domain.	- Ad-hoc and unsystematic identification of reuse opportunities.
Fair (4)	- Requirements are analyzed at project level.	- The developer has to relate project requirements to reusable assets at design level. - Linking it to other projects has to be done manually.	- Requirements management for software reuse is considered a single end-point.
Good (6)	- Requirements are analyzed at work product level.	- Requirements are leveraged to work products.	- Requirements at project level can be related to work products resulting in more systematic reuse.
Excellent (8)	- Requirements management is executed for a common architecture.	- Requirements are managed and analyzed over multiple projects in the same domain.	- Reuse opportunities are systematically identified based on requirements.
Outstanding (10)	- Requirements management is exploited.	- Requirements are managed, analyzed and exploited over multiple projects in the same domain.	- Applications can be built based on sets of requirements.

7. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

8. Process factor: Quality management

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

8. Process – Quality management (internal)			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- No quality estimations are made.	-	-
Weak (2)	- Quality is estimated based on individual experience.	- Experienced developers are used to solve quality issues.	- Opportunistic software reuse.
Fair (4)	- Quality models are available, but are limited adapted.	- Developers have quality models available, but still trust mainly on their own experience and judgements. - More practical solutions such as a ranking system may be implemented.	- Quality is mainly the result of experienced developers.
Good (6)	- Quality models are available and considered a good practice.	- Most components are judged on quality issues. - Some components may still be populated into a repository solely based on the experience of individuals.	- Quality models are known and regularly adapted.
Excellent (8)	- Quality models are available and obligatory.	- Quality models are used as judgement criteria before components are populated into a repository. - This way the use of quality models is enforced.	- Only quality components are populated into a repository.
Outstanding (10)	Reusable components are selected based on quality and its quality is frequently optimized.	Quality mode available across the entire organization and frequently reviewed for improvement.	- Continuous evaluation and improvement of quality related issues.

8. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

9. Process factor: Supplier management

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

9. Process – Supplier management (external)			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	No external assets are used.	-	-
Weak (2)	External assets are used, but no supplier management is present.	Past experience or perceived usefulness is guiding the selection of suppliers and related assets.	Initial benefits can be obtained from external acquisition of reusable assets, but are not obtained systematically.
Fair (4)	Supplier management is present and based on evaluating existing supplier relations.	Existing relationships are evaluated. Certificates or service level agreements are used to judge related quality.	Existing repositories are complemented with external assets.
Good (6)	Supplier management issues are evaluated before engaging in relationships.	Supplier selection and evaluation criteria are available. Supplier relations are regularly evaluated.	Existing repositories are complemented with external assets.
Excellent (8)	N/A *	N/A *	N/A *
Outstanding (10)	N/A *	N/A *	N/A

9. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

*N/A, not available because it is unknown how supplier relations for components can be further optimized. Outsourcing for optimizing is possible one of the solutions, but here the questions can be raised whether it is an optimal and wanted solution. For more information see also chapter 3.

10. Process factor: Configuration and change management

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

10. Process – Configuration and change management			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- No configuration management.	-	-
Weak (2)	- Configuration and change management is limited to a single project.	- Change management is addressed from the perspective of the relevant project. - At its best individuals use personalized tools to keep track of changes across multiple projects.	- Configuration management is considered as a single end-point.
Fair (4)	- Configuration management is based on reusable assets.	- Change requests are executed by users. - The component owner is responsible for managing the changes, using change management tool to go back to working versions or to release a new one.	- Reusable assets are maintained outside the scope of a single project. - A reactive approach is maintained.
Good (6)	- Configuration management is present for reusable assets and a separate group manages the change requests.	- A configuration management tool is used and change requests are formally submitted (e.g. bug tracking system). - The change requests are evaluated before they are become effective. - A separate reuse library is likely to be used for releasing stable versions.	- Reusable assets are maintained outside the scope of a single project. - A pro-active approach is maintained.
Excellent (8)	- Configuration and change management issues are addressed based on a common architecture.	- In addition to the previous level the change requests have to be in line with the common architecture.	- Reusable assets are maintained outside the project scope and within the defined architecture.
Outstanding (10)	- Configuration management is deployed, change request are managed and the process sis frequently reviewed and optimized.	- Advanced tool support is maximising the configuration management issues. E.g. tools for simultaneous development.	- Integrated and continuous configuration and change management through all projects and reusable assets.

10. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

11. People factor: Producer skills and experience

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

11. People – Producer skills and experience			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- Producer has no relevant skills and experience.	-	-
Weak (2)	- Producer uses present skills and experience.	- Experienced developers are used in multiple projects.	- Reusable assets may be created, but reuse is mostly based on 'code scavenging'.
Fair (4)	- Installed sense of responsibility and discipline in performing work practices.	- Experienced programmers are assigned a role for managing reusable assets.	- Results are repeatable, but performance is inconsistent.
Good (6)	- Identifying core competencies for reuse processes.	- Best practices are developed and installed.	- Established a foundation on which continuous development of knowledge and skills can be built.
Excellent (8)	- Mentoring both individuals and teams.	- Reuse activities performed by a producer are analyzed by mentors and improvements are made.	- Producer results can be measured and results become predictable.
Outstanding (10)	- Continuously improving individual competencies and finding innovative ways to improve workforce motivation and capability.	- Organizational support for continuous development. - Coaches are provided. Innovation is stimulated.	- Producers recognize reuse opportunities and exploit these.

11. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

12. People factor: Consumer skills and experience

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

12. People – Consumer skills and experience			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- No relevant reuse skills and experience.	-	-
Weak (2)	- Based on present people skills and knowledge.	- Little or no guidance is provided to consumers.	- Inconsistent performance.
Fair (4)	- Installed sense of responsibility and discipline in performing work practices.	Assigning roles will lead to the development of relevant practices.	Results are repeatable, but performance is inconsistent.
Good (6)	- Identifying core competencies for reuse processes.	- Best practices are identified. Training is provided.	- Established a foundation on which continuous development of knowledge and skills can be build.
Excellent (8)	- Mentoring both individuals and teams.	- Mentors use their experience to provide personal support, guidance and some skill development.	- Consumer skills can be measured and results are predictable.
Outstanding (10)	- Continuously improving individual competencies and finding innovative ways to improve workforce motivation and capability.	- Organizational support for continuous development. Coaches are provided. Innovation is stimulated.	- Consumer skills and practices are optimized.

12. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

13. Technology factor: Repository support

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

13. Technology – Repository support			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- No repository usage.	-	-
Weak (2)	- Project specific repositories are used.	- Repositories of previous projects are used to obtain reuse benefits.	- Reuse is based on experience of developers, where code is reuse in an ad-hoc manner.
Fair (4)	- A centralized repository is used, but its usage is poorly addressed and supported.	- Existing configuration management systems are used as repository.	- Leveraged code is placed in a repository. - Within proper management the repository has a high risk of resulting in a 'salvage yard'.
Good (6)	- A knowledge library is set up and the configuration management tool is placed to the background.	- Stable work products are released and made publicly available. - The configuration management tool is used to track changed related to the product evolution. This is done by a separate team.	- Consumers have a relatively stable set of components available. Its evolution is guided by a producer team running on the background. - Specifications and additional document can be placed with stable versions.
Excellent (8)	- The reuse repository is extended to meet the demands imposed by a common architecture.	- The reuse repository provides mechanisms to categorize components according to their fit with the architecture.	- Components are provided in the form of a basic tool-box of components. - The organization of components in an architecture is leading the development.
Outstanding (10)	- The reuse repository is optimized to meet reuse demands.	- The reuse repository provides advanced functionality tailored to the other used tools. E.g. exporting options to an application generator.	- Reuse efforts are minimized through advanced tool support.

13. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

14. Technology factor: CASE tool support

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

14. Technology – CASE tool support (consumer)			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- No CASE tool support.	-	-
Weak (2)	- No CASE tool support is provided for software reuse.	- No CASE tool support is deployed for software reuse; at its best individuals use their own tools.	- No visible results for the organization.
Fair (4)	- Project specific tool support is available	- CASE tool support is used and where possible integrated with the software reuse repository	- Reuse activities are supported through the use of CASE tool support.
Good (6)	- CASE tool support provides the basis for software reuse.	- Case tool support linked to the knowledge library provides the basis for advanced developers. - Additional documentation can be found on the knowledge library.	- Reuse activities are guided by CASE tool support. - Library is placed to the background and is used as a reliable knowledge source.
Excellent (8)	- CASE tool support is able to combine elements based on a common architecture.	- Case tool is integrated with the reuse repository and is able to provide a basic structure based on selected components.	- Basic applications are created which have to be customized for further development.
Outstanding (10)	- CASE tool support is able to generate applications.	- Case tool is integrated with the reuse repository and is able to automatically generate applications.	- Configurable basic applications are available.

*CASE stands for Computer Aided Software Engineering

14. Factor relevance					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

15. Technology factor: Communication tool support

A. Add a score to the reuse factor in the top right corner of the first table based on a suitable combination of '*approach*', '*deployment*', en '*results*'.

B. Circle the relevance of the reuse factor in the second table.

15. Technology – Communication tool support			***Score:***
Score	Key activity evaluation dimensions		
	Approach	*Deployment*	*Results*
Poor (0)	- No communication support is provided.	-	-
Weak (2)	- No communication support is provided.	- Reuse is based on individuals, they may share their knowledge with others, but the way of sharing is informal.	- No results are visible for the organization.
Fair (4)	- Communication support is provided, but not used systematically.	- Communication is supported through the use of a change management system and mailing lists. - Basis communication support is provided through the use of available tools.	- Reuse communication occurs and is separated from normal communication.
Good (6)	- Communication support is provided and used systematically.	- Dedicated tools are deployed to address communication issues.	- Reuse communication occurs regularly and is integrated.
Excellent (8)	- Communication support is evaluated and its results can be measured.	- Communication is supported through the use of dedicated tools.	- Relevant issues are visible organizational wide and can be mined.
Outstanding (10)	- Communication support is optimized.	- Communication support is integrated with development tools.	- Allowing simultaneous development across multiple projects.

15. Factor relevantie					
0	1	2	3	4	5
Absolutely not	Low		Relevant, but not critical		Critical

Final score:

BTOPP element	Reuse factor	Score per factor	Relevance per factor
Business	1. Domain focus		
Organization	2. Top management support and instrumental mechanisms		
	3. Organizational structure and reuse roles		
	4. Communication channels and organizational support		
Process	5. Reuse planning		
	6. Reuse measurement and cost justification		
	7. Requirements management		
	8. Quality management (internal)		
	9. Supplier management (external)		
	10. Configuration and change management		
People	11. Producer skills and experience		
	12. Consumer skills and experience		
Technology	13. Repository usage		
	14. CASE tool usage		
	15. Communication tool support		

Identified top 3*:

1: .....

2: .....

3: .....

* To identify a relevant top 3 the following rule of thumb was used: The relevance forms the basis for identifying the top 3 of factors to be improvement. The highest relevance is identified as the first point of attention. An attention point has a score lower than 5. A score higher than 5 is not identified as a point of attention.

Appendix N – Documents for validation: evaluation form

Name:

Function:

Business unit:

Age:

Years of relevant work experience:

General	1	2	3	4	5
	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
G1 The management tool offers me good guidance to analyze software reuse?	( )	( )	( )	( )	( )
G2 The incremental steps (presented by the maturity levels) are representative for the developments related to software reuse?	( )	( )	( )	( )	( )
G3 The management tool is easy to interpret?	( )	( )	( )	( )	( )
G4 The management tool is easy to use through the assessment methodology?	( )	( )	( )	( )	( )
G5 The found results are relevant for the company?	( )	( )	( )	( )	( )

Model specific	1	2	3	4	5
	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
S1: The maturity stages are a good way to address software reuse through various incremental stages?	( )	( )	( )	( )	( )
S2: The factors are a good way to describe software reuse?	( )	( )	( )	( )	( )
S3: The practices are a good way to evaluate the factors over the maturity stages?	( )	( )	( )	( )	( )

Applicability	1	2	3	4	5
	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
A1 The assessment method is a good way to apply the management tool?	( )	( )	( )	( )	( )
A2 The adding of a relevance variable is essential to successfully apply the management tool? (0-5)	( )	( )	( )	( )	( )
A3 The use of a fixed score is essential to apply the management tool? (0-10)	( )	( )	( )	( )	( )
A4 The subdivision in BTOPP (business, technology, organization, process and people) elements is useful to identify potential areas of improvement?	( )	( )	( )	( )	( )
A5 The found top 3 of potential improvement areas matches my own insights?	( )	( )	( )	( )	( )
A6 The application of the management tool has led to additional insights?	( )	( )	( )	( )	( )

Appendix O – Assessment scores

Score Table (1-10)												
	Results per person									*Total results*		
Reuse factor	***Person 1***	***Person 2***	***Person 3***	***Person 4***	***Person 5***	***Person 6***	***Person 7***	***Person 8***	***Person 9***	***Average***	***Standard deviation***	***BTOPP average***
1. Domain focus	6	6	7	5	8	4	6	7	8	6,3	1,3	6,3
2. Top management support and instrumental mechanisms	10	5	5	5	4	3	4	7	6	5,4	2,1	
3. Organizational structure and reuse roles	4	5	5	6	3	5	6	3	4	4,6	1,1	
4. Communication channels and support	5	6	5	3	3	5	4	4	6	4,6	1,1	4,9
5. Planning for reuse	6	7	5	4	2	4	2	2	6	4,2	1,9	
6. Reuse measurement and cost models	3	4	2	0	1	2	4	0	3	2,1	1,5	
7. Requirements management	2	4	4	3	4	3	4	2	4	3,3	0,9	
8. Quality management	3	3	2	2	2	1	2	2	4	2,3	0,9	
9. Supplier management	2	2	2	2	2	2	0	2	2	1,8	0,7	
10. Configuration and change management	4	4	4	5	2	3	2	2	4	3,3	1,1	2,9
11. Producer skills and experience	4	7	5	6	4	3	4	4	8	5,0	1,7	
12. Consumer skills and experience	4	5	2	6	4	4	4	6	7	4,7	1,5	4,8
13. Repository support	6	4	4	5	5	5	4	4	5	4,7	0,7	
14. CASE tool support	7	0	0	0	0	1	2	0	2	1,3	2,3	
15. Communication tool support	3	5	4	2	4	0	2	2	4	2,9	1,5	3,0

Appendix P – Assessment relevance

Relevance Table (1-5)												
	Results per person									*Total results*		
Reuse factor	***Person 1***	***Person 2***	***Person 3***	***Person 4***	***Person 5***	***Person 6***	***Person 7***	***Person 8***	***Person 9***	***Average***	***Standard deviation***	***BTOPP average***
1. Domain focus	5	4	2	4	4	4	4	5	4	4	0,9	4
2. Top management support and instrumental mechanisms	5	4	4	4	5	4	4	5	4	4,3	0,5	
3. Organizational structure and reuse roles	5	5	3	4	3	5	3	3	4	3,9	0,9	
4. Communication channels and support	4	3	1	3	4	3	4	4	4	3,3	1,0	3,9
5. Planning for reuse	4	3	3	3	4	5	1	2	4	3,2	1,2	
6. Reuse measurement and cost models	4	1	4	1	3	3,5	2	1	3	2,4	1,3	
7. Requirements management	3	4	2	2	2	4	3	2	3	2,8	0,8	
8. Quality management	4	2	3	3	3	4	2	1	2	2,7	1,0	
9. Supplier management	1	1	2	1	1	2	0	1	1	1,1	0,6	
10. Configuration and change management	3	5	2	4	4	3	4	2	4	3,4	1,0	2,6
11. Producer skills and experience	5	3	3	4	4	5	1	3	3	3,4	1,2	
12. Consumer skills and experience	3	3	4	3	3	1	1	5	3	2,9	1,3	3,2
13. Repository support	3	3	3	3	4	4,5	5	5	4	3,8	0,9	
14. CASE tool support	4	0	0	1	1	1	3	2	2	1,6	1,3	
15. Communication tool support	3	4	3	3	4	2	3	4	3	3,2	0,7	2,8

Appendix Q – Evaluation results

Evaluation Table (1-5)											
	Results per person									*Total results*	
Evaluation question	***Person 1***	***Person 2***	***Person 3***	***Person 4***	***Person 5***	***Person 6***	***Person 7***	***Person 8***	***Person 9***	***Average***	***Standard deviation***
G1	4	3	3	4	2	4	4	3	4	3,4	0,7
G2	5	4	3	4	4	5	4	4	4	4,1	0,6
G3	4	4	3	3	4	3	2	4	2	3,2	0,8
G4	4	3	3	4	4	3	5	4	3	3,7	0,7
G5	6	4	4	5	2	3	4	4	4	4,0	1,1
S1	5	4	5	4	4	5	4	3	4	4,2	0,7
S2	5	5	4	4	3	5	4	3	4	4,1	0,8
S3	5	4	4	4	4	4	3	4	4	4,0	0,5
A1	5	3	4	4	4	4	4	4	4	4,0	0,5
A2	5	2	4	4	3	3	3	4	4	3,6	0,9
A3	5	4	4	4	4	5	4	4	4	4,2	0,4
A4	5	3	4	4	4	5	4	4	4	4,1	0,6
A5	5	4	NA	3	2	2	2	4	2	3,0	1,2
A6	5	NA	NA	4	2	4	4	4	2	3,6	1,1