Software Reuse in Agile Development Organizations - A Conceptual Management Tool

Wouter Spoelstra
University of Twente
P.O. Box 217
7500 AE Enschede
+31 (0) 53 489 4134
w.j.t.spoelstra@student.utwente.nl

Maria Iacob
University of Twente
P.O. Box 217
7500 AE Enschede
+31 (0) 53 489 4134
m.e.iacob@utwente.nl

Marten van Sinderen
University of Twente
P.O. Box 217
7500 AE Enschede
+31 53 489 3677
m.j.vansinderen@ewi.utwente.nl

ABSTRACT
The reuse of knowledge is considered a major factor for increasing productivity and quality. In the software industry knowledge is embodied in software assets such as code components, functional designs and test cases. This kind of knowledge reuse is also referred to as software reuse. Although the benefits can be substantial, software reuse has never reached its full potential. Organizations are not aware of the different levels of reuse or do not know how to address reuse issues. This paper proposes a conceptual management tool for supporting software reuse. Furthermore the paper presents the findings of the application of the management tool in an agile development organization.

Categories and Subject Descriptors
D.2.13 [**Reusable Software**]: reuse models.

General Terms
Management, Measurement, Design, Performance and Theory.

Keywords
Software reuse, knowledge management, agile development, reuse maturity model, maturity levels, reuse factors and assessment method.

1. INTRODUCTION
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 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* in literature [26]. Mili et al. describe software reuse as a two-fold concept: first of all it is '*building software that is reusable by design*' and secondly it is '*building with reusable software'* [31].

SAC'11, March 21-25, 2011, TaiChung, Taiwan.

The two-fold definition explicitly indicates that software reuse it not purely 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 [19, 26, 31]. Systematic software reuse on its turn is defined as reuse which is repeatable and excludes ad-hoc reuse events [33].

The reuse of existing software assets offers intuitively great benefits, but despite its promises it has never reached its full potential. The first wave of researchers focused on the technical aspects to optimize software reuse. The second wave of researchers shifted towards the organizational aspects of software reuse [33].

Software reuse processes are at best a secondary concern as the focus of a normal project is on project specific goals and outcomes [33]. Both organizational and technical issues have to be addressed to facilitate software reuse within and outside the normal project scope in order to reach higher levels of reuse.

1.1 Problem statement
When analyzing the problem regarding software reuse in greater detail, it becomes evident that there is no holistic approach available to address software reuse issues. Literature assumes that systematic and formalized processes are key for achieving higher levels of reuse, but agile development organizations and the open source community are also successfully practicing software reuse without having these, often extensively documented and formalized procedures or task definitions.

Software reuse in agile development settings and the open source community indicate that there is a wide range of practices that have to be addressed to utilize software reuse [21]. Without a holistic approach to software reuse organizations are not able to address related issues and even more important they are not able to identify potential improvement areas for achieving even greater benefits.

1.2 Research goal and approach
The purpose of this research is to develop a conceptual software reuse management tool for addressing both technical and organizational reuse issues. In order to set up such a management tool a solid theoretical basis is required. A literature survey has been conducted to evaluate the current state of software reuse literature, existing reuse models and frameworks. Furthermore, the reuse issues are analyzed in greater detail through a systematic literature review of the top 25 IS journals. The proposed management tool is validated through an expert panel at a medium sized software development organization, providing valuable

insights in the software reuse practices and the validity of the management tool. The results are presented in the following chapters. In Section 2 we explain the core components of our approach. Section 3 is devoted to an account on the results of the validation we performed for our tool by applying it in an agile software development organization. We conclude the paper with a discussion of some conclusions and of several pointers to future work.

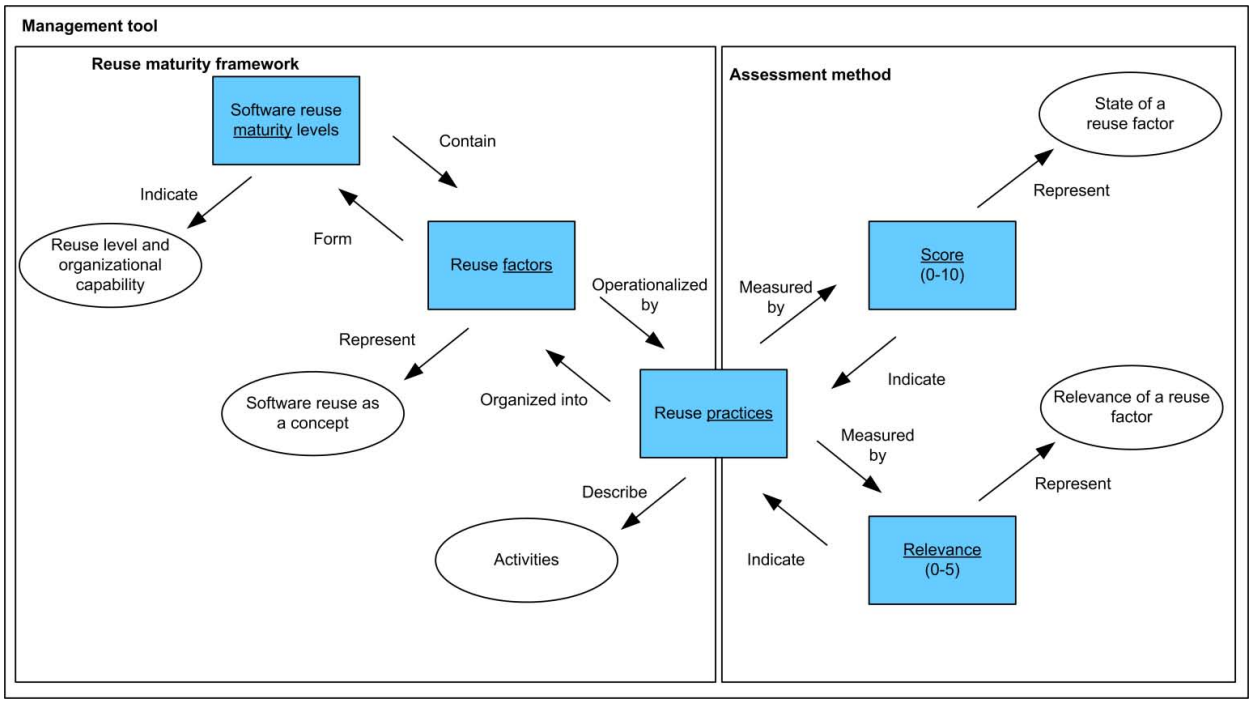

Figure 1: Overview of the conceptual management tool structure

2. REUSE MANAGEMENT TOOL

The proposed reuse management tool is a conceptual tool which has been set up after a careful analysis of existing reuse models and frameworks. The evaluated reuse models and frameworks include the work of: Holibaugh et al. [22], Koltun and Hudson [25], Prieto-Díaz [39], M. Davis [11], T. Davis [12], Wartik and Davis [52], Rine and Nada [42] and Garcia et al. [17]. All of them provide valuable insights, each from a slightly different perspective. None of them, however, address software reuse specifically for agile development. In fact most papers approach software reuse from a static perspective instead of a dynamic one. Lastly the vast majority of papers provides little or no guidance to the adaption of software reuse. In order to overcome the majority of these limitations we propose a new reuse management tool continuing on the basis provided by previous models. The basic structure is derived from the Software Process Improvement (SPI) framework defined by Niazi et al. [36]. The management tool contains a reuse maturity model based on maturity levels, reuse factors and a set of reuse practices. Together they form the first component of the reuse management tool. In order to operationalize the reuse management tool a second component is added which is the assessment method. This component defines a way to assess the state of a reuse factors by determining its score and relevance. The basis for the assessment method component is provided by the work of Daskalantonakis [10]. An overview is provided in Figure 1, all component elements are explained in greater detail in the following sections.

2.1 Maturity levels

The maturity levels are incremental plateaus for addressing reuse factors. After analyzing existing models and frameworks we decided to use a combination of the incremental reuse levels defined by Griss [18] and the maturity stages of the CMMI model [49]. Such a combination is possible because both approaches assume that additional investments will have to be made before higher levels of reuse can be achieved. Morisio et al. note for example that a library approach can be good with 30% reuse, but also that a product line approach can be bad with 70% reuse [33]. The investments for a product line approach are far more substantial than for a library approach and not all organization will or can reach these reuse levels. It remains up to the individual organization to determine which level is suitable for their business. The defined reuse levels for providing guidance are presented below:

At *level 1* ad-hoc reuse events are found. Ad-hoc reuse events are reuse events caused by individuals. The reuse events are not coordinated and not monitored. No formal 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 followed by the organisation. The code components are managed and controlled by a group, guiding reuse 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 analysis techniques. Also, the reuse processes are managed throughout the entire software 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. New components complement or extend the existing architecture. Each product is composed of or created by reusable assets exploiting software reuse to its limits.

2.2 Reuse factors

The second element of the maturity model component is the reuse factors. A reuse factor is describing a relevant aspect of software reuse, which can be related to both a technical and organizational issue. Previous reuse models and frameworks do not always organize their reuse factors into categories, making it difficult to address relevant areas. When looking at the models of Garcia et al. , Lucrédio et al. [30] and Nerur [35] a slightly different mix of reuse factors can be found. Returning to the work of Morton [34] a synthesis between the various defined categories is suggested. An overview of the model of Morton is presented in Figure 2.

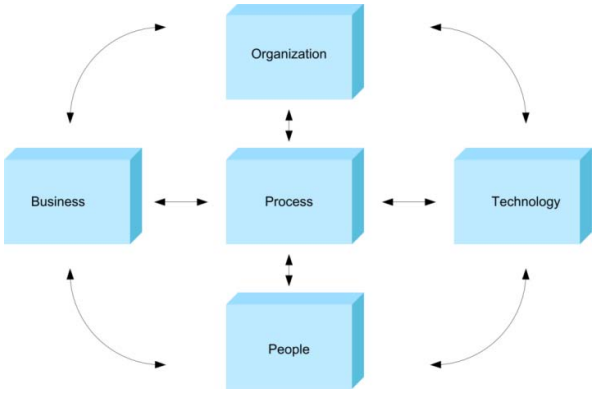

Figure 2: BTOPP model [34].

The BTOPP model stands for the Business, Technology, Organizations, Process and People model. The model is used to organize the reuse factors in a coherent way. The reuse factors are identified through a systematic literature review to the top 25 IS journals as identified by Schwartz and Russo [47]. Only the empirical papers were selected found by the search queries 'software reuse' or 'component reuse'. After removing the false positives and performing forward and backward searching on the remaining articles, a total of 24 articles were found. These articles include the work of: Banker [2], Banker [4], Desouza [13], Fafchamps [14], Frakes and Fox [15], Frakes and Fox [16], Griss [19], Griss [20], Haefliger [21], Isoda [23], Joos [24], Lee and Litecky [28], Lim [29], Lucredio et al. [30], Mohagheghi [32], Morisio et al. [33], Prieto Diaz [38], Purao [40], Rine [41], Rine and Sonneman [43], Rosenbaum [44], Rothenberger [46], Rothenberger [45] and Selby [50]. The identified reuse factors are elaborated on in greater detail per BTOPP category.

2.2.1 Business factors

The literature review resulted in the identification of the business factor 'domain focus'. The domain focus is an indicator for the level of commonalities among products. The development of applications for a narrow and focused domain will likely result in high levels of commonalities among created solutions, while the development of solutions for a broad and unfocussed domain will likely result in low levels of commonalities. The software reuse levels are therefore influenced by the strategic choice of an organization to focus on a certain domain. Griss also recognizes the domain focus as a form of a need to commit to software reuse [19]. He notes that when there is no need to commit to reuse employees may behave according to the Not Invented Here (NIH) syndrome [19]. This syndrome assumes that developers prefer to build their own assets instead of reusing the assets created by someone else.

2.2.2 Organizational factors

We have identified three organizational factors: 'top management support and instrumental mechanisms', 'organizational structure and reuse roles' and 'communication channels and organizational support'.

Top management support and instrumental mechanisms are taken as a factor for indicating the commitment of top management. The support provided by top management can be both passive and active. They can be reinforced through several instrumental mechanisms. Instrumental mechanisms mentioned in literature include the use of reuse champions, sample solutions, rewards and incentives, reuse education and training [15, 31].

The organizational structure and related reuse roles are often approached as a formal separation between producers of reusable assets and consumers of reusable assets [8, 14, 37] The producers of reusable assets are those who create the assets and the consumers are those who use the assets in building new solutions. The way how these reuse roles and others are installed within the organizational are related to the reuse levels.

The identification of communications channels and how they are supported is identified as a potential reuse factor. This reuse factor is embodied in a systematic software reuse process and includes the communication of change requests [14, 23]. The factor is emphasized by agile development methodologies as they tend to rely on informal processes and face-to-face communication [5].

2.2.3 Process factors

The process factors can be directly related to a systematic reuse process. The literature review resulted into six process factors: 'reuse planning', 'reuse measurement and cost justification', 'requirements management', 'quality management', 'supplier management' and 'configuration and change management'.

Reuse planning describes whether reuse events are planned in the beginning of a software development project or they pop-in during the development process. Reuse planning is an indicator for a systematic reuse process [2, 30]. Domain analysis may be used for systematically scanning the domain for potential reuse opportunities [41].

The use of reuse measurement and costs models is another reuse factor indicating a systematic reuse process. The measurement of

reuse activities may be enforced by top management for performing cost-benefit analysis [24]. Investments in reuse measurement and costs models may be substantial and are not always desired [33]. The factor is nevertheless taken as a reuse factor as it contributes to a systematic reuse process.

Requirements management over multiple projects can provide a solid basis for identifying commonalities, which is an indicator of a systematic software reuse process [26]. Based on the commonalities potential reusable assets can be identified and produced.

Quality management is arguably one of the most important reuse factors. Rosenbaum describes quality management as mandatory for a successful reuse program [44]. Quality management can be installed through the use of quality models or ranking systems [21, 41]. The way how quality management is addressed and its importance differs over the various reuse levels. Supplier management addresses issues related to the acquisition of externally acquired assets (such as from black-box component markets and open source projects), asset certification and legal aspects [15].

The last process factor is configuration and change management required for the managed of reusable assets. Without configuration and change management, reusable assets are expected to start 'having a life of their own'. In literature configuration and change management is addressed by assigning dedicated roles [38] or assuming that components are carefully tested before they are populated into a repository [26]. In practice additional mechanisms are likely to be present varying over different reuse levels.

2.2.4 People factors

The literature review identified two people factors which are 'producer skills and experience' and 'consumer skills and experience'.

A producer is a person actively working and the production of reusable assets. Producer skills and experience include both technical aspects such as programming languages skills, but also organization aspects such as domain specific knowledge. The latter one is required to recognize reuse patterns and to separate project specific functionality from functionality reusable over multiple projects [46].

A consumer is a person who is using existing reusable assets to create new solutions. Consumer skills and experience relate to all the knowledge and practices required at the consumer side including knowledge about the reusable assets itself and best practices for integrating reusable assets.

The people factors are operationalized in the reuse maturity model component through the use of the People Capability Maturity Model (P-CMM) [48].

2.2.5 Technology factors

Lastly, the literature review identified three technology factors including 'repository support', 'CASE tool support' and 'communication tool support'.

Repository supports refers to the use of a repository for storing and retrieving reusable assets. This reuse factor includes the use of search and retrieval techniques [4] and configuration management tools. A simple configuration management tool may serve as the basis for providing and managing reusable assets [33].

Computer Aided Software Engineering (CASE) tool support refers to the use of integrated programming tools [3]. Effective software reuse may make use of tools where the repository is integrated in the programming tools, reducing reuse barriers as much as possible. By further integrating CASE tool support the idea of an application generator may become feasible [26].

Communication tool support is used to describe the active support of communication among producers and consumers through the use of tools. The reuse factor is included into the management tool as it is also identified as a possible way to scale agile methodologies for more complex environments.

2.3 Reuse practices

The combination of a reuse factor and a maturity level is operationalized through the use of reuse practices [36]. A reuse practice describes a set of practices which a reuse factor has to meet before it reaches a certain maturity stage. The reuse practices form the basis for the assessment method component discussed in the following two sections. Due to space limitations in this paper, the practices are not further elaborated on. The original set of reuse practices and related information can be found in the master thesis of Spoelstra [51].

2.4 Factor scoring and factor relevance

The assessment method component consists of the measurement of two variables. The first is the scoring of reuse factors, which is done according to the three dimensions defined by Daskalantonakis: 'approach', 'deployment' and 'results' [10]. The combination of these three dimensions leads to a fixed score on a scale of 0-10. By dividing the score by 2 the related reuse level can be found. An odd score can be used to indicate that the current reuse level shows characteristics of two adjacent reuse levels, but has not met all the characteristics of the upper level.

The second element of the assessment method is the relevance variable. The relevance variable is added to the assessment method to overcome the limitation of not being able to tailor the management tool to the demands of the individual organization. Furthermore the relevance variable is used as an indicator of the need to scale a certain reuse factor. The use of a relevance variable favors the continuous approach of the CMMI-DEV model [9]. By using the relevance variable in multiple case studies in similar research settings it is expected that reuse patterns can be discovered. The use of reuse patterns favors the staged approach of the CMMI-DEV model. Investigating reuse patterns was out of scope for this research.

3. VALIDATION RESULTS

The management tool has been validated based on three validity factors, which are: 'completeness', 'internal consistency' and 'applicability'. The identified validity factors can be compared with the work of Lagerström et al. [27]. An expert panel has been selected within the case organization for performing the validation process. During the validation process the expert panel was asked to apply the management tool and evaluate the tool during and after the application process. The first section of this chapter introduces the case organization and the expert panel. After that the application results are presented per BTOPP category. An overview of the results is also presented in Table 1. In the last section the expert evaluation results are presented.

Table 1: Application results

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

3.1 Case description and expert panel

The case organization is a medium sized Dutch software development organization. This organization utilizes agile development methodologies [5] to be able to quickly respond to fast changing dynamic markets. Within the organization each of 9 members and covers four business units operating on the same knowledge base. The roles covered by the expert panel are: developer, analyst and project manager. Each role is covered multiple times.

3.2 Business factor results

All the business units of the case organization have defined their own market niche. In some cases there may be overlap between the defined market niches offering opportunities for collaboration between business units. Some business units have matured their solutions more for their market niche than others, but all of them have relevant domain experience. Because the domain focus is for a large part defined it is also architected to a certain extend. A true product line architecture as defined by Bosch [8] is however not used, as the customer demands are leading and not the existing architecture. The domain factor was considered as very relevant.

3.3 Organizational factor results

Software reuse activities are stimulated and encouraged within all business units. As one of the experts noted: '*software reuse is considered an integrated part of the selling strategy*'. Top management assigned dedicated resources for the production of reusable assets. Due to resource constraints the roles dedicated to business units focuses on separate market segments (e.g., financial sector, healthcare, etc). Because each business unit is operating in a separate market segment, its domain is defined to a certain extend and natural levels of reuse exist. The expert panel consist

software reuse are merely part time roles and only the most important reusable assets are maintained actively. The other reusable assets are often created within the normal project scope and have to be extracted from there in order to be reused. The communication channels are installed and information is shared across multiple business units. The extent of sharing is however informal and relies greatly on individual efforts. Some experts noted that they would like to see more information from other business units, but at the same time such information is likely to be irrelevant as the most important aspects are shared. Furthermore the experts emphasized that sharing information is a desired situation and it should not be formalized. Simply asking around is considered as being a good practice. Additional documentation in general could further improve the information flow though. The three reuse factors we have identified were considered by the expert panel as relevant factors for software reuse.

3.4 Process factor results

The processes regarding software reuse can grouped around two sets of components. The first set of components consists of those that are centrally and more formally managed. The assessment results for these components were remarkably higher than for the

other set of components. The other set of components is managed more informally often within a business unit or a specific project. The components have to be extracted from these projects first before they can be reused. For both component sets the customer demands are directive. The customer is effecting a pull mechanism on the development of components. The components are not developed for being pushed into the market. In some rare cases a separate project is set up to create reusable assets, this can be due to converging versions of reusable assets which have to be merged again or because the expected paybacks of a reusable asset are so obvious that the costs can be easily justified. Apart from these extraordinary situations reuse events are usually planned during the design phase of a project or emerge during the development phase. In the latter case experienced developers recognize and exploit reuse opportunities. The chance of successfully creating reusable assets is highest at the beginning of a project, because priorities shift near the end of the project and negatively influence the resources available for creating reusable assets.

The experts noted that reuse events and costs can be roughly estimated based on the experience of a developer and relevant domain knowledge. Formal reuse measurement and costs models are not present at the case organization.

Requirements are analyzed at customer level and are only to a limited extent useable over multiple projects. It is mainly the individual who recognizes and exploits reuse opportunities. The idea of requirements management across multiple projects has been proposed by the business, but so far its applicability appears to be limited.

Quality models appeared seem to be less relevant as the organization relies on the expertise of developers. An expert noted that: *'when a component 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. This line of thought indicates a possible relation between the amount of times a component is reused and its quality. Formal quality models may be more applicable in larger organizations.

The experts were in general not positive about externally acquired assets from black-box component markets. The functionality required was often slightly different from the functionality provided by such components. Open source projects such as .Spring and NHibernate are successfully integrated and have proven their value. Formal supplier management appeared to be not applicable for open source projects.

Configuration and change management seems to be difficult at the case organization. Individual projects wish to reuse a component in a slightly different that the way they are offered, leading to the possible existence of multiple versions. These versions have to be merged back at a certain point in time, requiring substantial efforts. Configuration and change management was installed through a revision management system and through frequent meetings regarding the most important assets. This point is expected to require additional attention in the future.

Our observation is that process factors scored consistently lower during the evaluation process compared to the other reuse factors. The experts also added lower relevance variables indicating that major improvement are not desired. The discussion with the experts confirmed this statement. The essence seems to be that reuse events should be planned in the beginning of a project as much as possible, outside the project scope the efforts are difficult to justify.

3.5 People factor results

A formal role distinction between producers and consumers of reusable assets was less applicable at the case organization, as roles are interchangeable. This is also a characteristic of agile development environments [6]. The experts noted that producers of reusable assets are usually experienced developers in line with the findings of Fafchamps [14]. Best practices have been defined and the producers sometimes act as a coach towards the consumers of reusable assets, helping them learn the required knowledge and skills. When a consumer wants to change a component the change request must be placed at the producer of this asset. If the change request is accepted the change requester has to populate the change into the software asset. When a consumer decides the skip this step a separate component version is created leading to possible merge conflicts in the future. The assessment of the people factors appeared to be difficult as the People Capability Maturity Model seems to be more applicable for larger organizations. Smaller organization do address the development of skills and experience, but arguable do not do this structurally. Additional research is required to improve the assessment of the people factors.

3.6 Technology factor results

The first set of components is stored within a central revision management system. The other component set is usually stored within a project specific repository. These components have to be extracted from a project first before they can be reused, they are however set up in such a way that this can be done relatively quickly. Documentation such as functional designs and test cases is stored separated from the assets if available at all.

CASE tool support is integrated as far as desired, namely, for interface components. The experts do not see potential in further investing in CASE tool support for software reuse. In a narrower domain it may however offer additional benefits.

Communication tool support is installed through the use of mailing lists and a bug tracking system. Both tools are not specific for software reuse and are used in a broader sense. The experts emphasized that communication channels should remain informal. Wikipedia pages may be used for complementary documentation.

The technology factors were also considered relevant factors. The experts did note that the technology factors are rather supportive. Basic technology issues have to be addressed and can be developed along the demands of the organization.

3.7 Expert evaluation

This section presents the expert opinions based on the evaluation during and after the application process. The proposed reuse levels are considered useful and relevant. During the discussion presented in previous sections of this chapter, several points regarding the reuse factors already became clear. Supplier management and CASE tool support are considered less relevant, but may be more applicable in other organizations. Furthermore, the experts emphasized the importance of the selling strategy related to the domain focus. It is reasonable to assume that the selling strategy is the basis for the domain focus, it was however not taken explicitly in the reuse factor itself. Due to interchangeable roles the experts had problems with making a

distinction between the people factors, but at the same time it proved to be valuable for discussing relevant aspects from different viewpoints. Based on the expert evaluation no direct changes were proposed to the proposed set of reuse factors, they are considered complete without superfluous elements. The reuse practices were not evaluated in detail, but the application process did provided valuable insights. A high standard deviation among the results can possibly serve as an indicator for assessment difficulties. The relatively high standard deviation for the people factors confirmed such difficulties. The used practices for the people factors are derived from the P-CMM [48], which are likely to be more applicable in larger organizations. The experts did note that these practices contained relevant elements, but the exact matching of the elements was a combination spread out over multiple reuse levels. Improving the practices through the use of additional case studies likely leads to a higher applicability of the management tool in general. The experts also suggested improving the assessment method through the use of an additional variable. This variable should measure the desired scalability of a reuse factor explicitly. The relevance variable indirectly includes the scalability as well, but the way it includes the scalability did not completely satisfy the experts.

4. DISCUSSION AND CONCLUSION

This paper proposes a conceptual management tool for addressing software reuse issues. The management tool extends existing literature and provides valuable insights in the adoption of software reuse in agile development organizations. Although the tool was intended to be specific for agile development organizations it was set up in a more generic way, because agile development methodologies have proven to be applicable in more complex environments. The combination of the CMMI-DEV model with agile methodologies seems to be contradictive, but recent literature investigates the combination of both resulting in several success stories by balancing both aspects [7]. In all cases it remains up to the individual organization to choose which levels of the management tool are desirable for the business environment. Furthermore, the case study resulted in insights regarding the validity of the management tool. CASE tool support and supplier management appeared to be less applicable for the case, but are expected to be more important for other organizations. The assessment of the people factors should be further improved through the use of additional case studies. The use of a single case study is considered a major limitation of this research. Another limitation is that the performed case study is based on a great amount of quantitative data. Despite the use of a voice recorder, transcripts and multiple data sources the results are coloured by individual perception. Interestingly, the results discovered during the case study are in line with the expectations of agile development organizations, which maintain less formalized processes and focus on people aspects. The case study confirms that such organizations are indeed well capable of identifying and seizing reuse opportunities as stated in the introduction. Future research can use the assessment results as a basis for identifying reuse patterns. A reuse pattern defines a combination of reuse scores and relevance variables linked to certain types of organizations. When several reuse patterns have been identified they can be used for new organizations as implementation guideline. The purpose of the management tool in such a case is then no longer focussed on evaluating software reuse, but also on providing prescriptive guidelines for implementing software reuse into organizations.

5. ACKNOWLEDGMENTS

This research would not have been possible without the support of the anonymous experts and the resources provided by the case organization. Furthermore we want to thank Jasper Laagland and Johan te Winkel for their support and contributions made to the content of this paper. The work presented in this paper is also partly supported by the Agile Service Development project, which is part of the Service Innovation & ICT program (www.si-i.nl), sponsored by the Dutch Ministry of Economic Affairs.

6. REFERENCES

[1] Alavi, M. and Leidner, D.E. Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. *MIS Quarterly*, *25* (1). 107-136.

[2] Banker, R.D. and Kauffman, R.J. Reuse and productivity in integrated computer-aided software engineering: an empirical study. *MIS Quarterly*, *15* (3). 375-401.

[3] Banker, R.D., Kauffman, R.J., Wright, C. and Zweig, D. Automating output size and reuse metrics in a repository-based computer-aided software engineering (CASE) environment. *IEEE Transactions on Software Engineering*, *20* (3). 169-187.

[4] Banker, R.D., Kauffman, R.J. and Zweig, D. Repository evaluation of software reuse. *IEEE Transactions on Software Engineering*, *19* (4). 379-389.

[5] Beck, K. Agile Manifesto, http:/agilemanifesto.org/.

[6] Boehm, B. Get ready for agile methods, with care. *Computer*, *35* (1). 64-69.

[7] Boehm, B. and Turner, R., Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. in *Proceedings - International Conference on Software Engineering*, (2004), 718-719.

[8] Bosch, J. Organizing for Software Product Lines. in *Software Architectures for Product Families*, 2000, 117-134.

[9] CMMI-ProductTeam. CMMI-DEV, V1.2 - Improving processes for better Products, Software Engineering Institute, 2006.

[10] Daskalantonakis, M.K. Achieving Higher SEI Levels. *IEEE Software*, *11* (4). 17-24.

[11] Davis, M.J., 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), M 1-6.

[12] Davis, T. The reuse capability model: A basis for improving an organization's reuse capability *proceedings of the 2nd International Workshop on Software Reuse*, Lucca, Italy, 1993, 126-133.

[13] Desouza, K.C., Awazu, Y. and Tiwana, A. Four dynamics for bringing use back into software reuse. *Communications of the ACM*, *49* (1). 96-100.

[14] Fafchamps, D. Organizational Factors and Reuse. *IEEE Softw.*, *11* (5). 31-41.

[15] Frakes, W.B. and Fox, C.J. 16 Questions About Software Reuse. *Communications of the ACM*, *38* (6). 75-&.

[16] Frakes, W.B. and Fox, C.J. Quality improvement using a software reuse failure modes model. *IEEE Transactions on Software Engineering*, *22* (4). 274-279.

[17] Garcia, V.C., Lucredio, D., Alvaro, A., De Almeida, E.S., De Mattos Fortes, R.P. and De Lemos Meira, S.R. Towards a maturity model for a reuse incremental adoption *SBCARS 2007 - Brazilian Symposium on Software Components, Architectures and Reuse*, Brazil, 2007, 61-74.

[18] Griss, M. Systematic Software Reuse: Architecture, Process and Organization are Crucial, Fusion Newsletter, HP Laboratories, Oct 1996, 1996.

[19] Griss, M.L. Software reuse: from library to factory. *IBM Syst. J.*, *32* (4). 548-566.

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

[21] Haefliger, S., Von Krogh, G. and Spaeth, S. Code reuse in open source software. *Management Science*, *54* (1). 180-193.

[22] Holibaugh, R., Cohen, S., Kang, K. and Peterson, S. Reuse: where to begin and why *Proceedings of the conference on Tri-Ada '89: Ada technology in context: application, development, and deployment*, ACM, Pittsburgh, Pennsylvania, United States, 1989.

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

[24] Joos, R. Software Reuse at Motorola. *IEEE Softw.*, *11* (5). 42-47.

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

[26] Krueger, C.W. Software reuse. *ACM Computing surveys*, *24* (2). 131-183.

[27] Lagerström, R., Johnson, P. and Ekstedt, M. Architecture analysis of enterprise systems modifiability: a metamodel for software change cost estimation. *Software Quality Journal*.

[28] Lee, N.Y. and Litecky, C.R. An empirical study of software reuse with special attention to ada. *IEEE Transactions on Software Engineering*, *23* (9). 537-549.

[29] Lim, W.C. Effects of reuse on quality, productivity, and economics. *IEEE Software*, *11* (5). 23-30.

[30] Lucrédio, D., dos Santos Brito, K., Alvaro, A., Garcia, V.C., de Almeida, E.S., de Mattos Fortes, R.P. and Meira, S.L. Software reuse: The Brazilian industry scenario. *Journal of Systems and Software*, *81* (6). 996-1013.

[31] Mili, H., Mili, F. and Mili, A. Reusing software: issues and research directions. *Software Engineering, IEEE Transactions on*, *21* (6). 528-562.

[32] Mohagheghi, P. and Conradi, R. An empirical investigation of software reuse benefits in a large telecom product. *ACM Transactions on Software Engineering and Methodology*, *17* (3).

[33] Morisio, M., Ezran, M. and Tully, C. Success and Failure Factors in Software Reuse. *IEEE Trans. Softw. Eng.*, *28* (4). 340-357.

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

[35] Nerur, S., Mahapatra, R. and Mangalaraj, G. Challenges of migrating to agile methodologies. *Commun. ACM*, *48* (5). 72-78.

[36] Niazi, M., Wilson, D. and Zowghi, D. A maturity model for the implementation of software process improvement: an empirical study. *J. Syst. Softw.*, *74* (2). 155-172.

[37] Poulin, J.S. Technical opinion: reuse: been there, done that. *Commun. ACM*, *42* (5). 98-100.

[38] Prieto-Díaz, R. Implemeting Facted Classification for Software Reuse. *Communications of the ACM*, *34* (5). 88-97.

[39] Prieto-Díaz, R. Making software reuse work: an implementation model. *SIGSOFT Softw. Eng. Notes*, *16* (3). 61-68.

[40] Purao, S. and Storey, V.C. Evaluating the adoption potential of design science efforts: The case of APSARA. *Decision Support Systems*, *44* (2). 369-381.

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

[42] Rine, D.C. and Nada, N. An empirical study of a software reuse reference model. *Information and Software Technology*, *42* (1). 47-65.

[43] Rine, D.C. and Sonnemann, R.M. Investments in reusable software. A study of software reuse investment success factors. *Journal of Systems and Software 41* (1). 17-32.

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

[45] Rothenberger, M.A. Project-level reuse factors: Drivers for variation within software development environments. *Decision Sciences*, *34* (1). 83-106.

[46] Rothenberger, M.A., Dooley, K.J., Kulkarni, U.R. and Nada, N. Strategies for software reuse: A principal component analysis of reuse practices. *IEEE Transactions on Software Engineering*, *29* (9). 825-837.

[47] Schwartz, R.B. and Russo, M.C. How to quickly find articles in the top IS journals. *Commun. ACM*, *47* (2). 98-101.

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

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

[50] Selby, R.W. Enabling reuse-based software development of large-scale systems. *IEEE Transactions on Software Engineering*, *31* (6). 495-510.

[51] Spoelstra, W.J.T. Reusing software assets in agile development organizations - a management tool, Master thesis, University of Twente, Enschede, 2010.

[52] Wartik, S. and Davis, T. A phased reuse adoption model. *Journal of Systems and Software*, *46* (1). 13-23.