Value-Based Selection of Requirements Engineering Tool Support

Matthias Heindl, Franz Reinisch
Siemens Program and Systems Engineering, Gudrunstraße 11, A-1100, Vienna, Austria
matthias.a.heindl@siemens.com
franz.reinisch@siemens.com

Stefan Biffl
Inst. of Software Tech., Vienna Univ. of Tech., A-1040, Vienna, Austria
Stefan.Biffl@tuwien.ac.at

Alex Egyed
Teknowledge Corporation, 4640 Admiralty Way Marina Del Rey, CA, USA
aegyed@acm.org

Abstract

In large software and systems engineering companies like Siemens PSE there are several requirements tools in use. There is no "one tool fits all projects/departments" solution for requirements engineering due to the variety of departments and project types. Thus, project managers and requirements engineers usually face the problem of selecting the most suitable tool sets available for their given situation.

In this paper, we report on a value-based requirements tool selection approach developed at PSE that helps to find the optimal tool support. This selection is based on rating the value contribution of suitably-defined tool features for the given project context. We conducted an initial feasibility study with a number of typical requirements tools at PSE. Main results were: The approach is straight-forward and seems to be a good means for project managers to compare requirements tools and select the most valuable tool for a particular project.

1. Introduction

Siemens Program and System Engineering (PSE) is an independent research and development entity within the Siemens group with more than 6000 employees. PSE's software (and also hardware) development projects cover many application domains: telecommunication and information technologies, automation and control, power, transportation, medical solutions, components, and space technology. Therefore, PSE projects span a wide range of types, differing in project size, duration, degree of distribution, application domains, tooling and process models.

Handling requirements in such projects is complex due to the following factors:

- Interdependencies of requirements to various other artefacts like design specifications, source code, test cases;
- Requirements management spans the whole product lifecycle and has to interact with other disciplines (from design to maintenance);
- All project members are involved in handling requirements.

Project managers and requirements engineers, as well as other project participants, need appropriate requirements tool support to manage this complexity. There is a wide range of requirements tools available with very different capabilities, concepts, and terminologies. Therefore, a challenge for the project manager is to evaluate and select the most appropriate tool. In this context, the Support Center for Configuration Management (SCCM) at PSE developed a comprehensive and easy-to-use requirements tool selection approach that aims to help project managers in a value-based way to select requirements tools. The SCCM is a unit in the PSE providing (amongst other services) consulting for employees and enabling experience exchange among them on key topics like configuration and requirements management.

In this paper, we present an approach that is based on a generic requirement engineering (RE) process description that considers RE as an intrinsic part of the whole development processes (rather than an isolated discipline). From this process description we derived a well-structured requirements tool feature catalogue. Additional input came from literature and empirical information from experienced tool users at PSE. This feature catalogue serves two purposes: (1) it allows a standardized rating of tool capabilities in order to compare different tools (tool rating model), and (2) it can be used to create a value-based profile of needed tool capabilities for a given project situation (tool value model).

The feature catalogue can conveniently be represented as feature tree, consisting more than 80 features. Top nodes in the tree (derived from the RE process description) are similar to the structure of the require-

Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA'06)
0-7695-2594-6/06 $20.00 © 2006 IEEE

ments analysis part of the guide for the software engineering body of knowledge (SWEBOK) [2]. Compared to already existing feature catalogues like INCOSE [5], new essential requirements tool features were identified. Furthermore, essential requirements, originating in our view on RE as tightly integrated with other involved disciplines (from design to maintenance), were given respect. With this approach, we conducted an initial tool evaluation with 5 commercial and 2 in-house requirements tools.

The remainder of this paper is as follows: Section 2 outlines related work on value-based tool decisions. Section 3 sketches the research questions and proposes the value-based tool selection approach. Section 4 then contains a discussion on mapping between requirements tools and project types. Finally, section 5 and 6 present a conclusion and further work.

2. Related work

Existing tool selection approaches, as well as value-based software engineering, and requirements engineering are parts of related work as explained in the following subsections.

2.1. (COTS) Tool selection

As requirements tools are commercial-off-the-shelf (COTS) products, we shortly outline related work for tool respectively COTS selection within this subsection.

In the 1990s Le Blanc and Korn [6] proposed a structured approach for the evaluation and selection of CASE (computer aided systems engineering) tools. The base of the approach was a list of criteria like "common repository", "standard user interface", "hardware/software portability", and "network support".

Over time other more comprehensive feature lists emerged, e.g., from the International Council on Systems Engineering (INCOSE) [5]. The features of our feature tree were derived from a RE process and later on balanced with the INCOSE catalogue. They are structured to the requirements engineering activities (see below) in a feature tree to support clarity, and is extended by some new features identified by practitioners and experienced tool users at PSE.

Alves and Finkelstein [1] provide a goal-driven requirements engineering perspective for COTS decision-making. They describe a rather high-level COTS selection process consisting of steps like acquiring goals, understand COTS, match goals and COTS, and finally selecting COTS.

We consider our approach to be more low-level (feature level) and therefore more practical to use. We describe the concrete means to compare tools and support the tool decision.

The VOLERE method [10] proposes to use predefined checklists to ask the right questions at the right time. For example, the method encourages the requirements engineer to ask the most important questions like "How to measure this requirement?" at the right time.

Ncube and Maiden [7] report lessons learned when selecting a complex commercial off-the-shelf (COTS) software system. He discusses problems of and solutions regarding requirements acquisition and product selection. For example, requirements that enable effective discrimination between products should be described in more detail, requirements must be as measurable as possible to enable product selection, and stakeholder representatives should be present during product evaluation.

With their PORE approach [8], they propose an easily understandable approach for guiding parallel requirements acquisition and COTS software selection, consisting of a) acquire information from stakeholders, b) analyse acquired information, c) make decisions about product-requirement compliance, and d) select one or more candidate COTS software. Our approach follows the same high-level process, but the low-level PORE realization seemed to effort-consuming for our situation.

Ochs et al. in [9] propose a COTS assessment and selection approach, the CAP method, which is based on a CAP evaluation taxonomy containing categories of measures, e.g., usability, integration cost. VERPRO [12] extends this taxonomy by incorporating business factor measures influencing the COTS selection decision. Both approaches, CAP and VERPRO are measurement-based, whereas our approach seems to be more feature-based, that is the functionalities are.

2.2. Value-based software engineering

Value-based software engineering (VBSE) [3] brings value considerations to the foreground to support software engineering decisions at all levels in order to meet or reconcile explicit objectives of the involved stakeholders, from marketing staff and business analysts to developers, architects, and quality experts, and from process and measurement experts to project managers and executives. In VBSE, decisions are not made in a setting blind to value perspectives, whether common or differing, of these project participants.

In contrast to existing tool selection approaches, where value considerations were, if at all, considered only implicitly, our tool selection approach contains a value

model that enables explicit integration of stakeholder value propositions into the decision process with the goal to optimize the decision-making.

2.3. Requirements engineering (RE)

Since our value-based RE tool selection approach is based on a comprehensive tool feature catalogue which is structured to the requirements engineering main activities, we shortly outline the main parts of a requirements engineering process in this subsection [11].

Requirements Elicitation. Requirements Elicitation involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints. The goal is to gather raw requirements.

Requirements Analysis and Negotiation. Requirements analysis and negotiation is an activity which aims to discover problems and conflicts with the requirements and reach agreement on changes to satisfy all system stakeholders (people that are affected by the proposed system). The final goal is to reach a common understanding of the requirements between all project participants.

Requirements Documentation and Validation. The defined requirements, written down in a software requirements specification, are validated against criteria like correctness, completeness, consistency, verifiability, unambiguity, traceability, etc.

Requirements Management. Requirements Management consists of managing changes of requirements (keeping them consistent), e.g., by ensuring requirements traceability (identification of interdependencies between requirements, other requirements, and artifacts).

Requirements management can be seen as integrated part of change and configuration management.

2.4. Configuration Management

Software Configuration Management is the process of identifying and defining components in a system, controlling the release and change throughout the lifecycle, recording and reporting the status of components and change requests, and verifying the completeness and correctness of systems components (IEEE standard 729).

State-of the-Art Configuration Management (CM) can be described as a discipline which helps to implement a desired process in real project life. CM provides the process infrastructure for one or many projects (e.g., version management, requirements management, change management, access control, workflow, collaboration).

3. Value-Based Tool Selection Approach

The main parts of the approach are: (1) the tool rating model that was created by developing a generic requirements engineering process (step 1), deriving a tool feature tree from the process (step 2), and finally an evaluation of requirements tools (step 3), and (2) the value rating model, that allows project managers to rate the value contribution of requirements tool features for their project.

3.1. Research Questions

The research questions we address in this paper are:

- What are the tool needs from a practitioner's point of view?

We developed a practice-oriented tool feature tree with experienced tool users at PSE.

- How can we integrate stakeholder value propositions into the tool selection decision in a simple and practical way?

We describe a tool rating and value rating model as means to support a value-based tool selection.

- Which requirements tools (and features) have which value for certain types of projects?

We discuss the relationship between tool features and their value for certain project types.

3.2. Developing the Tool Rating Model

The approach to develop a rating model was thought as follows: First, the need for tool support of requirements engineering activities should be modelled without respect to the current abilities of commercial requirements tools.

The modelled need resulted in a tool feature tree that was then used as checklist to evaluate how existing requirements tools meet the needs for requirements tool support. The following paragraphs describe the approach in more detail.

Step 1. Development of a requirements engineering process description. This process describes the main activities in requirements engineering, e.g., requirements elicitation, requirements validation, requirements management. Well knowing that the requirements tools to be evaluated focus on requirements management and do not support requirements elicitation (most requirements tools used in PSE are requirements management tools), the evaluators still modeled the need for requirements elicitation tool support to raise the full scope of requirements engineering tool support needs.

Step2. Development of a tool feature tree based on the RE process description. For each requirements engineering activity the evaluators derived desirable

tool features to support the activity. Additional input for this step came from INCOSE's tool feature list, and from experienced tool users at Siemens PSE (see elicited new tool features in table 1). Features are described in an understandable and unambiguous manner (uses cases or scenarios were attached where appropriate).

Table 1. Illustrating subset of new tool feature needs elicited at PSE survey and interviews.

Tool feature	Description
Definition of a workflow for requirements	A workflow (states, roles, state transitions) is configurable for requirements, like it is usual for other artifacts in configuration management.
Automated generation of bi-directionality of traces	When the user creates a trace between artifact A and artifact B, it automatically establishes a backward trace from B to C without interaction of the user.
Definition of user-specific trace types	An authorized user can define trace types and assign names to these trace types .
Suspect traces	When a requirement changes, the tool automatically highlights all traces related to this requirement for checking and updating traces.
Long-term archiving functionality	All data in the tool can be archived in a format accessible without the tool in order to re-setup the environment if necessary.

Different from other tool catalogues, e.g., [4], the result was a tool feature tree structured similarly to the elements of the "software requirements analysis" part of the guide to the software engineering body of knowledge (SEWBOK) [2]. The leaves of the feature tree consist of more than 80 tool features. A cutout of the feature tree is depicted in figure 1. The software requirements analysis module of the guide to the SWEBOK contains elements like requirements engineering process, requirements elicitation, requirements analysis, requirements validation, and requirements management. For the latter element, figure 1 depicts a sub-tree containing the requirements tracing-related tool features.

The nodes below "requirements tracing" are a classification layer that supports clarity and togetherness of tool features. For example, the features under "types of traces" represent a tool's ability to create traces between requirements and design, source code (at different levels like class or method level). "Support of understandability" refers to the ability to name traces. "Trace generation" contains features that allow manual or automated trace generation. "Automated generation of bi-directionality for traces" means, that a tool is able to automatically establish a trace from B to A when the user creates a trace from A to B.

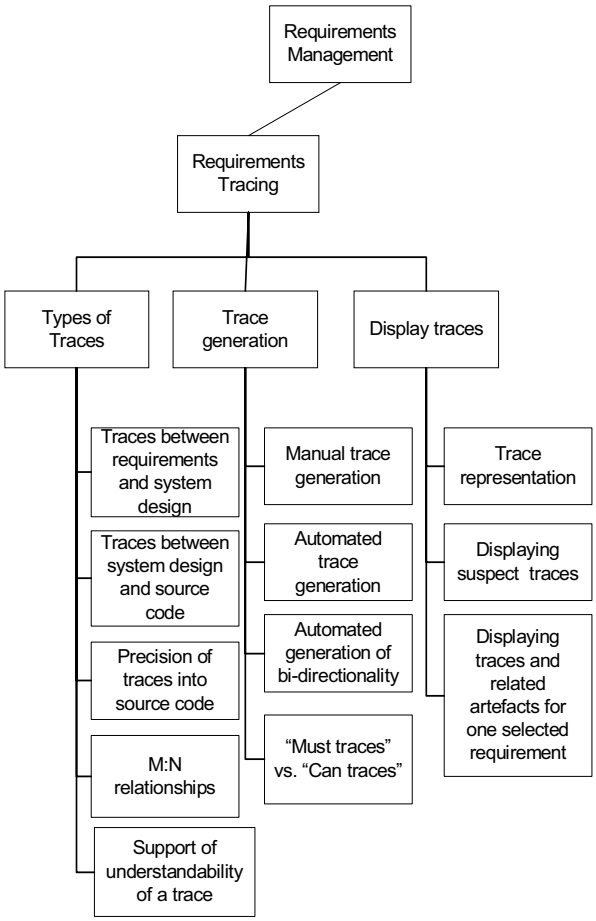

Figure 1. Cutout from the feature tree

"Display traces" contains features that support different ways of displaying traces: "trace representation" contains graphical, matrix, or tabular trace representation; "suspect traces" means that the tool highlights traces (suspect traces) that have to be checked, because one artifact attached to these traces has changed.

Besides requirements management, the feature tree also contains features for requirements elicitation, validation, documentation (e.g., import and export features), requirements engineering process configuration, and configuration management, as well as usability features.

The advantages of the tool feature tree in comparison to other types of tool feature catalogues are:

- The feature tree addresses the whole requirements engineering process, thus, not only on requirements management;
- The tree is structured similarly to the SWEBOK to provide a clear framework for software engineers,
- The tree contains new features that came from experienced tool users at PSE (see table 1 for examples).

The feature tree was found to be mostly stable; however with changing software engineering and requirements engineering processes, some parts of the feature tree are expected to evolve.

Step 3. The Support Center "Configuration Management" at PSE performed initial tool evaluations of 5 commercial and 2 in-house requirements tools.

Table 2. Example rating of tool features.

Tool feature	Rating Tool X	Comment
Nameable trace types	0	Only pre-defined trace types provided.
Automated generation of bi-directional traces	4	When a trace from A to B is captured, the tool automatically creates an inverse trace from B to A.
Tracing into code	2	Possible only to code files, e.g., java files, no traces to method level possible.

The elements of the feature tree (leaves) were used as a checklist for the evaluation and for a given tool each feature was rated on a scale from 0 (feature not provided by the tool) to 4 (feature fully provided by the tool). Additionally, comments of how each feature was implemented in a tool were captured for each feature. The rating of an example tool is illustrated in table 2.
The results of our rating model were validated with experienced tool users at PSE.

As mentioned above, the evaluators in the Support Center evaluated seven tools. The tools were selected, because either they were already in use in the PSE, or there were at least negotiations conducted with tool suppliers. The set of evaluated tools contained some of the most popular requirements management tools, as well as PSE in-house solutions.
The rating model is scalable by rating new requirements tools with the feature tree as checklist.

The resulting artifacts of the rating model were: a feature tree outlining desirable functionality for requirements-related activities, and for each evaluated tool a rating form (excel) describing if and how the tool provides a feature in order to provide means for comparison of existing tools.

3.3 Tool Value Model

As feature catalogues are mostly value-neutral, this subsection describes, how project managers and leaders of organizational units can use the feature tree described above to value the tool features and thereby find the most suitable tool that best addresses the project- or unit-specific stakeholder value propositions.

First of all, the rating model above provides a simple means for project managers to get an overview about features of existing tools. It is a good means to point out the importance of certain features which project managers were unaware of.
Further, project managers have two options to rate the value of a given tool: a combined rating and a rating by value classes, as described in the following paragraphs.

Combined value rating. The combined rating allows a project manager to rate the value of features with the following scale: A(3 points), B(2 points), C(1 point), D(0 point). An example is depicted in table 3.
In the example, the project manager rated "nameable trace types" with D (0 points) because in his project he does not want to create individual trace types; he rather wants to use existing trace types. "Automated generation of bi-directional traces" is very valuable (A), because the project is about a very large system and lots of traces have to be generated. Therefore, the project manager supposes this feature to save efforts for requirments tracing. "Tracing into code" is demanded by internal instructions and is also rated very important. Highlighting traces that have to be checked due to the change of an attached artefact was considered as medium important.

Adding up the points of this rating, results in a total score of 8 points. Subsequent normalization in the example shows that feature 1 provides 0% of the tool's value for the project manager, features 2 and 3 each provide 37.5% value, and feature 4 provides 25% of the value. Starting from these results, the project manager, assisted by Support Center members, can map these values to the tool feature ratings and identify the best-fitting tool.
A weak point of this kind of combined rating is that A, B, C, and Ds are lumped together in terms that one A is worth 3 Cs, and one B is worth 2 Cs. Therefore, another option for value rating is the rating by value classes.

Table 3. Examples for combined value rating model.

Nr	Tool feature	Value rating
1	Nameable trace types	D
2	Automated generation of bi-directional traces	A
3	Tracing into code	A
4	Displaying suspect traces	B

Rating by value classes. Instead of using a quantitative approach for the value rating, rating by value classes enable a qualitative way of reasoning.
The value classes are not assigned with numerical values, but with qualitative values, namely: A (crucial), B (important), C (nice to have), D (unimportant). Thereby, the interdependencies between the rating classes are cancelled, because you can no longer say that, e.g., A is three times more valuable than C.
Again, a project manager can now rate each feature of the feature tree with A, B, C, or D. Finally, the numbers of As, Bs, Cs, and Ds can be counted separately, and the tools' numbers can be compared to find the best-fitting tool: First the number of "A"-features are compared and then the tool with the highest number of "A"-features is picked. The "B"-features will be checked only in the case there is an equal number of "A"-features.
Besides this feature-based value rating, which represents a bottom-up rating, the feature tree further allows a top down rating, where the high-level nodes, e.g., the requirements engineering main activities are rated first to eliminate unimportant sections of the feature tree, and then the leaves of the remaining tree are rated as described above. This saves effort, because parts of the feature tree can be blended out because of their unimportance, which is another benefit of the feature tree structured according to the requirements engineering activities. In addition to value rating of already evaluated requirements tools, a project manager has also the option to rate his usual requirements engineering tool support using the tool feature list. This gives him the opportunity to compare the proposed tools to his usual tool and allows weighing the effort for adopting a new tool against the tool support improvement.

4. Discussion: Value of tools/features for Project Types

A coarse analysis of the project types at Siemens PSE showed us the big variety of project types ranging from small, collocated projects with a handful of members, agile projects, to large, highly-distributed projects with dozens of project members.
These projects do not only differ in organizational aspects, but also concerning topics. Besides many others, there are traditional software development projects, covering the whole software development lifecycle, as well as pure verification projects, where the main focus is on system testing and ensuring traceability between requirements and test cases. While the value of a traceability feature that enables traces between requirements and test cases may be only of average importance for the traditional software development project, it has a great value for the verification project.
This leads to the assumption that the value of tool features can not be stated independent from the project context and the stakeholder value propositions within this project. Based on this assumption, our value-based tool selection approach provides a means to identify the stakeholder value propositions by developing a value model and thereby find the optimal tool support.

Furthermore, our feature tree does not only represent features that are implemented in available requirements tools, but also features that are desireable for practitioners. For example, practitioners wished to have forums withing their requirements tools to discuss and negotiate requirements in distributed projects. Until now, neither of the evaluated tools provided these features. Thus, the feature tree, if maintained properly in the future, could also be a means for tool vendors to identify needs for new features.
In this context, it is also a means for the Support Center to explain features to a project manager and to point out the value a feature might have for him, which he was unaware until then.
Another discovery we made during developing the tool selection process was that the proportion between the number of tool features of a tool and the total value of this tool seems not to be linear. We suppose it to be S-shaped, as exemplarily depicted in figure 2. This figure is an estimate for the value of some examplary tool features for a large-scale, highly-distributed project, and is just for illustration. It needs further work to more precisely analyse the proportion between features and value.
The segments of the S-curve are "infrastructure", containing necessary tool features that do not provide a specific value. The middle part is the "high-payoff" segment containing features that in context of the given project type provide a strong increase of value (steep value ascent). The last segement is the "nice-to-have" part that contains features that do not extremely increase the total value of the tool.

In our example in figure 2, an "adaptable requirements engineering process " feature means that a state model for requirements can be configured in a tool, containing permissions which tool users are allowed to change the state of a requirement. Possible states could be: elicited, validated, to be checked, etc. In figure 2, this feature provides only a low value for the project (see the low ascent of the curve), because it provides a good infrastructure, but could also be realized outside the tool by carefully defining a process. "Requirements versioning" is also something like a "basis requirement" that does not provide a specific value to the given project type.

The features in the high-payoff segment all provide an increased value for the given project type. Structuring and classification of requirements, e.g., by a requirements tree structure or a file explorer structure, helps to keep the overview, which is very valuable due to the high amount of requirements usually existing in large-scale projects. Although one would expect that this is a feature that is provided by all existing tools, there are some that do not provide this feature. Thus, the latter would not be a good solution for the large-scale project type.

Due to the high number of requirements and artefacts emerging during the project, some kind of automated traceability support, e.g., automated generation of bidirectional traces, are very valuable because they help to save a lot of effort. Since all traces have to be birectional, in order to follow the life of a requirement in a forward and backward direction, such a feature halves the effort for bidirectional traceability, because only a trace in one direction has to be created.

Multi-User support in highly-distributed projects, where one team is e.g. located in China, one in Austria, and one in Slovakia, is extremely valuable in order to support coordinated collaboration, file access, and well-defined user permissions.

Furthermore, multi-project capability can be an inevitable requirement for organization-wide used solutions.

The last section of the S-curve in our example contains features that do not provide a big additional value, illustrated by the flat ascent of the curve. Such features are all kind of add-ons, e.g., additional report functionalities or WYSIWYG-editors.

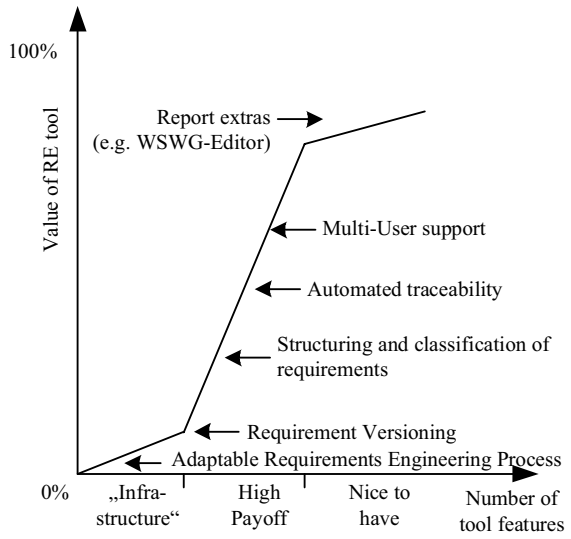

Figure 2. S-curve to illustrate the proportion between features and value.

A further goal will be to more precisely analyze the project types and its characteristics and to map tool features to these project types in S-curve manner in order to reflect the value of certain features for certain project types (see further work). Further work also contains ways how to evaluate the value of features for project types.

5. Conclusion

There is a big variety of projects at Siemens Program and Systems Engineering (PSE) and project managers or leaders of organizational units face the challenge to select their requirements tool that best fulfills the project- or department-specific (one solution for many projects) needs. This is a challenge because the number of available requirements tools, commercial as well as open source tools, is very high, and the comparison of these tools is very expensive and time-consuming.

For that reason, the Support Center Configuration Management, a unit of the PSE competence basis department, developed an approach to provide decision support for project managers in requirements tool selection decisions.

The approach contains:

- A requirements engineering process description that provides an introduction and structuring of requirements engineering activities (requirements elicitation, analysis, documentation, validation, prioritization, management)
- Based on this process description, evaluators derived a practice-oriented feature tree, which is structured after the guide to the software engineering body of knowledge. The structure provides clarity and a good overview.

- The evaluators used this feature tree as a checklist to evaluate seven currently available requirements tools. The result is a rating model by which the strength and weaknesses of each tool can be easily compared.
- Furthermore, a value model allows project managers to value the importance of the features for their projects and thereby identify the most suitable requirements tool support that best addresses the project manager's value propositions.

The resulting artefacts are a good means for the Support Center to provide decision support to project managers and other tool decision makers. The rating model can be easily extended by rating new tools. It helps to point the value of certain features out that the project manager was not aware before. The long-term goal could be to reduce the number of used requirements engineering tools, and thereby to provide a higher level of support for a given tool. Furthermore, costs for tool support and maintenance could be reduced.
The initial feasibility study showed that the proportion between the number of tool features and value is not linear, but s-shaped. Furthermore, different tool features seem to provide different values to different kinds of projects, which will be analyzed in further work.

6. Further Work

Further work will be the extension of the existing rating model (the currently evaluated tools) with other requirements tools, open source as well as other commercial requirements tools. This should ease the comparison of tools on feature basis.
Another idea is to make the value rating approach applicable not only for one stakeholder, but for multiple stakeholders.
Furthermore, we aim to concretize the classification of project types in order to improve the mapping between tools and tool features to project types and to further analyze the proportion between tool features and value for these defined project types.
In a last step, we want to develop some kind of tool support that: (a) allows evaluators to rate new tools and to store and update their ratings for given tools, and (b) supports project managers, assisted by Support Center representatives, in more efficiently finding the most suitable requirements tool. It is also planned to develop a requirements management experience base with the tool rating results as input.

References

[1] Carina Alves, Anthony Finkelstein, Workshop on software engineering decision support: components: Challenges in COTS decision-making: a goal-driven requirements engineering perspective, Proc. 14th int. conf. on Software engineering and knowledge engineering SEKE '02, July 2002

[2] P. Bourque, R. Dupuis, A Abran, The Guide to the Software Engineerin Body of Knowledge, IEEE Software, Nov/Dec. 1999

[3] Biffl S., Aurum A., Boehm B., Erdogmus H., Grünbacher P. (eds.) (2005), Value-Based Software Engineering, Springer Verlag, ISBN:3-540-25993-7.

[4] Hoffmann, M.; Kuhn, N.; Weber, M.; Bittner, M.; Requirements for requirements management tools, Requirements Engineering Conference, 2004. Proc. 12th IEEE International, 2004 Page(s):301 – 308

[5] International Council on Systems Engineering, http://www.paper-review.com/tools/rms/read.php

[6] Louis A. Le Blanc, Willard M. Korn, A structured approach to the evaluation and selection of CASE tools, Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing, March 1992

[7] Neil A. M. Maiden, Cornelius Ncube, Andrew Moor, Lessons learned during requirements acquisition for COTS systems, Communications of the ACM, Vol.40 Issue 12, Dec. 1997

[8] Ncube, C.; Maiden, N.A.M.; Guiding parallel requirements acquisition and COTS software selection, Proc. IEEE International Symposium on Requirements Engineering 7-11 June 1999 Page(s):133 - 140

[9] Ochs, M.; Pfahl, D.; Chrobok-Diening, G.; Nothhelfer-Kolb, B, A method for efficient measurement-based COTS assessment and selection method description and evaluation results, Proc. 7th Int. Software Metrics Symposium, 2001. METRICS 2001., 4-6 April 2001 Page(s):285 – 296

[10] Robertson S. The VOLERE process model. Internal Document, Atlantic Systems Guild, London, 1996,

[11] Sommerville, Sawyer: Requirements Engineering, John Wiley & Sons, 1997

[12] Yeoh, H.C.; Miller, J.; COTS acquisition process: incorporating business factors in COTS vendor evaluation taxonomy, Software Metrics, 2004. Proceedings. 10th International Symposium on 14-16 Sept. 2004 Page(s):84 - 95