SOFTWARE MAINTENANCE KNOWLEDGE-BASED SYSTEM (S3MXPERT): WEB-BASED SOFTWARE MAINTENANCE EXPERT TRAINING

Alain April, Jean-Marc Desharnais
École de Technologie Supérieure – Université du Québec
1100 Notre-Dame West, Montréal, Québec H3B 4L1
Canada
Alain.april@etsmtl.ca, Jean-Marc.Desharnais@etsmtl.ca

ABSTRACT

Maintaining and supporting the software of an organization is not an easy task, and software maintainers do not currently have access to tools to evaluate strategies for improving the specific activities of software maintenance. This article presents a web-based training approach using a knowledge-based system which helps in locating best practices in a software maintenance capability maturity model ($S^3m$). The contributions of this paper are: 1) To define a task analysis, 2) to instrument the maturity model with a support tool to aid software maintenance practitioners in locating specific best practices; and 3) to describe the web-based training approach and system overview used by the research team.

KEY WORDS
Expert system, decision support, software maintenance maturity model, web-based training.

1. Introduction

Knowledge transfer of a large number of best practices, described in a maturity model, has proved difficult [1]. This is especially true during the training stage for an assessor or a new participant in a process improvement activity. It is also challenging to quickly refer to, or access, a specific practice, or subset of practices, when trying to answer questions during or after a maturity evaluation.

It would be beneficial to have a knowledge-based system (KBS) to help access this complex structure and its large amount of information. A potential solution to this problem is to develop a knowledge-based system that the underlying process model used by the maturity model.

This paper presents the proposed modeling of a software maintenance KBS based on the van Heijst methodology [2], which consists of constructing a task model, selecting or building on an existing ontology [3], mapping the ontology onto the knowledge roles in the task model and instantiating the application ontology with this specific domain knowledge. According to van Heijst, there are at least five different types of knowledge to be taken into account when constructing such a system: tasks, problem-solving methods, inferences, the ontology and the domain knowledge1. For van Heijst, domain knowledge refers to a collection of statements about the domain[2]. The process domain of this specific web-based expert training system is software maintenance. Examples of the maturity model knowledge based statements are presented in section 2. At a high level, the ontology refers to a part of the software maintenance ontology proposed by [4] presented in section 3. The inferences, problem-solving methods and tasks are described at length in section 4 showing the task analysis for the proposed knowledge-based system.

The tool environment (showing a user interface layout of the knowledge-based system) followed by a conclusion, as well as future work, are presented in sections 5 and 6.

2. $S^3m$ and Knowledge Statements

Software maintainers experience a number of problems. These have been documented and an attempt made to rank them in order of importance. One of the first reported investigations was conducted by Lientz and Swanson [6].

They identified six problems related to users of the applications, to managerial constraints and to the quality of software documentation. Other surveys have found that a large percentage of the software maintenance problems reported are related to the software product itself.

This survey identified complex and old source code which was badly documented and structured in a complex way. More recent surveys conducted among attendees at successive software maintenance conferences [7] ranked

1 van Heijst uses the different types of knowledge in a more generic way than we do in this document, and these have been adapted for us by Desharnais, J.-M., Application de la mesure fonctionnelle COSMIC-FFP: une approche cognitive, UQAM, 2004 Montréal
perceived problems in the following order of importance (see Table 1).

These are also examples of knowledge statements about the domain of software maintenance. Key to helping software maintainers would be to provide them with ways of resolving their problems by leading them to documented best practices.

Table 1: Top maintenance problems [7]

<table>
<thead>
<tr>
<th>Rank</th>
<th>Maintenance problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Managing fast-changing priorities</td>
</tr>
<tr>
<td>2</td>
<td>Inadequate testing techniques</td>
</tr>
<tr>
<td>3</td>
<td>Difficulty in measuring performance</td>
</tr>
<tr>
<td>4</td>
<td>Missing or incomplete software documentation</td>
</tr>
<tr>
<td>5</td>
<td>Adapting to rapid changes in user organizations</td>
</tr>
<tr>
<td>6</td>
<td>A large number of user requests in waiting</td>
</tr>
<tr>
<td>7</td>
<td>Difficulty in measuring/demonstrating the maintenance team’s contribution</td>
</tr>
<tr>
<td>8</td>
<td>Low morale due to lack of recognition</td>
</tr>
<tr>
<td>9</td>
<td>Not many professionals in the field, especially experienced ones</td>
</tr>
<tr>
<td>10</td>
<td>Little methodology, few standards, procedures or tools specific to maintenance</td>
</tr>
<tr>
<td>11</td>
<td>Source code complex and unstructured</td>
</tr>
<tr>
<td>12</td>
<td>Integration, overlap and incompatibility of systems</td>
</tr>
<tr>
<td>13</td>
<td>Little training available to personnel</td>
</tr>
<tr>
<td>14</td>
<td>No strategic plans for maintenance</td>
</tr>
<tr>
<td>15</td>
<td>Difficulty in meeting user expectations</td>
</tr>
<tr>
<td>16</td>
<td>Lack of understanding and support from IT managers</td>
</tr>
<tr>
<td>17</td>
<td>Maintenance software running on obsolete systems and technologies</td>
</tr>
<tr>
<td>18</td>
<td>Little will for reengineering applications</td>
</tr>
<tr>
<td>19</td>
<td>Loss of expertise when employees leave</td>
</tr>
</tbody>
</table>

There is a growing number of sources where software maintainers can look for best practices, a major challenge being to encourage these sources to use the same terminology, process models and international standards.

The practices used by maintainers need to show them how to meet their daily service goals. While these practices are most often described within their corresponding operational and support processes, and consist of numerous procedures, a very large number of problem-solving practices could be presented in a KBS which would answer their many questions about those problems.

Examples are presented in section 5. When using the software maintenance ontology in the KBS, it was necessary to consider the structure of the maturity model relationship between the many process domains, roadmaps and practices. This problem is addressed next.

3. Ontology of the Software Maintenance Body of Knowledge

We elected to implement only a subset of the ontology developed by Kitchenham et al. [5] and Ruiz et al. [7] for the initial trial of this research project. Figure 1 describes the different maintenance concepts considered surrounding a software maintenance activity.

Software maintenance is highly event-driven, which means that some maintenance activities are unscheduled and can interrupt ongoing work. This subset of the ontology represents many, but not all, the concepts involved in responding to the questions related to the first problem identified by Dekleva: ‘Managing fast-changing priorities.’

Maintainers agree that this is the most important problem they face. How can they handle the fast-changing priorities of the customer? Solutions to this problem are likely to be found by using many paths through the maintenance concepts of the ontology.

Navigation through these concepts should lead to associated concepts which are conceptually linked and likely to contribute to a solution, like the need for better event management, change control, maintenance planning, Service Level Agreements, maintenance manager negotiation, training, procedures, and so forth.

Many more concepts must be involved to contribute to all aspects of the solution, but our purpose is to show the utility of a KBS in the software maintenance domain, and it therefore starts with a constrained number of concepts. Maturity models typically include the detailed best practices that could be of help in solving this type of problem.

The main issue is that the best practice locations and their interrelationships are hidden in the layered architecture of the maturity model, specifically in its process domains, KPAs and roadmaps.

It is therefore necessary to find a way to link this layered architecture with the maintenance concepts of the ontology and proceed to analyze the tasks required to build a KBS to support the maintainers in their quest for solutions. The next section presents the web-based training system task analysis that supports the knowledge based system.
Figure 1: Part of the software maintenance ontology of (Kitchenham et al., 1999)

4. Task Analysis

According to [2], the first activity in the construction of a web-based KBS is the definition of task analysis. Task analysis begins, at a high level, with a definition of an index of terms. This index includes words commonly used in software engineering (see Figure 3). From this index, a subset of more restrictive words is identified. This subset is a list of keywords recognized specifically in software maintenance.

Each keyword is then connected to one or more maintenance concepts. A maintenance concept, in software maintenance, is a concept found in the Software Maintenance Body of Knowledge and ontology (see Figure 2). Using the software maintenance ontology, every software maintenance problem identified by Dekleva has been linked to themes (questions) which help the user of the KBS to navigate to the part of the maturity model that will propose recommendations in the form of best practices.

Expanding the 5 high-level tasks in Figure 2, we propose 15 detailed tasks which will help identify a best practice related to the $S^{me}$. The link between the maintenance concepts and the maturity model is made in the themes concept. Themes are questions which have been developed to hop from node to node in the ontology. Themes concept can send the user to another theme, to another maintenance concept (up arrow), or, finally, to a recommendation of the maturity model (down arrow). A number of themes, in the form of questions, are presented to the user to guide him through the network of maintenance concepts. For every best practice, there are a number of themes (or choices) from which the user can select (also called facts) which will lead to a specific recommendation.

There are also a number of sub-tasks related to the maintenance processes and the maintenance best practices. This step-by-step process corresponds to the establishment of a diagnosis on the basis of the identification of symptoms. It indicates probabilities of occurrence of a specific software maintenance problem. No symptom is sufficient by itself to confirm the existence of a specific problem. This is why we should use the word “diagnosis”. The task model is used to help “diagnose” the current maintenance practice and map it to the maintenance model.
5. Tool Environment

The S3Mxpert web-based training system was built using Java script and XML, and supports the S3m. The architecture, design and implementation details. The design of the web-based training system is based on using both the case-based and ruled-based approaches [8]. The S3Mxpert KBS was developed by two Master’s degree students from the University of Namur, Belgium, during a research exchange program with our university [9].

There is still a great deal of work required to populate the knowledge base for all the S3m practices to allow users to obtain answers to all the software maintenance problems identified by Dekleva. Figure 3 shows an example of the web-based training system user interface layout. In this case, the user requests a recommendation in a case where the service request is very costly. A number of questions (themes) are asked by the system. According to the answers, there will be a specific recommendation which could either suggest further research or provide an opinion.

There are also interfaces for both the administrator and the expert. The administrator interface manages access to S3Mxpert, while the expert interface gives the expert the option of adding new keywords, concepts, cases, themes and recommendations.

![Figure 2: High-level view of S3Mxpert](Image)

![Figure 3: S3Mxpert user interface layout](Image)
6. Tool Environment

Identifying the best practices in a maturity model is a difficult task, considering their number and the multiple appropriate answers associated with each of them. Our hypothesis is that a KBS could help in finding an appropriate recommendation.

The next step in this research project is to populate the KBS, validate the results with experts in the domain and determine whether or not the KBS is a useful support tool for training on the content of the maturity model. It will be also necessary to improve the interface, mainly for the sake of the expert.

7. Conclusion

Identifying the best practices in a maturity model is a difficult task, considering their number and the multiple appropriate answers associated with each of them. Our hypothesis is that a KBS could help in finding an appropriate recommendation.

The next step in this research project is to populate the KBS, validate the results with experts in the domain and determine whether or not the KBS is a useful support tool for training on the content of the maturity model. It will be also necessary to improve the interface, mainly for the sake of the expert.

References


