Software Maintenance expert system (*SM^{xpert}*)
Measuring the use of the knowledge base

Alain April, Jean-Marc Desharnais, Ph.D.
École de Technologie Supérieure, 1100 Notre-Dame West, Montreal, Canada
Email: alain.april@etsmtl.ca , jean-marc.desharnais@etsmtl.ca

Keywords: Knowledge-based system, software maintenance, maturity model, ontology, decision support.

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 knowledge-based system which helps in locating best practices in a software maintenance capability maturity model (***S3^m***). It presents an XML-based usage of the knowledge base to measure the concepts most often employed by software maintainers. The contributions of this paper are: 1) to instrument the maturity model with a support tool to aid software maintenance practitioners in locating specific best practices; and 2) to describe an XML-based measurement approach to locate the concepts most often accessed by users.

1. INTRODUCTION

Knowledge transfer of a large number of the best practices described in a maturity model has proved difficult (Abran et al., 2004). This is especially true during the training of an assessor or a new participant in a process improvement activity. It is also challenging to identify where the effort of defining the knowledge base should be placed. We set out to develop an XML profiling tool for use in identifying the concepts most often employed by software maintainers, in order to help focus the knowledge base development effort.

The ***S3^m*** maturity model contains a large number of concepts and information structured in many successive levels (April et al., 2004b, 2002; April et al., 2004a). The first is called the process domains level, and reflects the main process knowledge areas of a maturity model. In the ***S3^m***, there are 4 process domains (*process management*, *maintenance request management*, *software evolution engineering* and *support to software engineering evolution*). Each process domain is broken down into one or more key process areas (KPAs). These KPAs logically group together items which conceptually belong together. A KPA is further divided into *roadmaps* with one or more *best practices*, spanning five ***S3^m*** maturity levels. Thus, the complete ***S3^m*** consists of 4 domains, 18 KPAs, 74 roadmaps and 443 best practices.

It would be beneficial to have a knowledge-based system (KBS) to help access this complex structure and the large amount of information it contains. A potential solution to this problem would be to develop a knowledge-based system for the ***S3^m***. A proposed modeling of a software maintenance KBS based on the van Heijst methodology (van Heijst et al., 1997), which consists of constructing a task model, selecting or building an ontology (Uschold and Jasper, 2001), mapping the ontology onto the knowledge roles in the task model and instantiating the application ontology with this specific domain knowledge, was presented in (April 2005). 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 knowledge. For van Heijst, domain knowledge refers to a collection of statements about the domain (van Heijst et al., 1997). The domain of this specific research is software maintenance, and it is divided into 4 process domains. Examples of statements are presented in section 3. At a high level, the ontology refers to a part of the software maintenance ontology proposed by (Kitchenham et al., 1999) presented in section 4. The inferences, problem-solving methods and tasks are described at length in section 5. The tool environment and conclusion, as well as future work, are presented in sections 6 and 7. Section 2 begins by presenting the goals of the ***S3^m*** architecture.

2. GOALS OF THE *S3^m* ARCHITECTURE

The ***S3^m*** was designed as a customer-focused benchmark for either:

- auditing the software maintenance capability of a service supplier or outsourcer; or
- supporting the process improvement activities of software maintenance organizations.

To address the concerns specific to the maintainer, a distinct maintenance body of knowledge is required. The ***S3^m*** is also designed to complement the maturity model developed by the SEI at Carnegie Mellon University in Pittsburgh (CMMi, 2002) by focusing mainly on practices specific to software maintenance. The architecture of the model locates the most fundamental practices at a lower level of maturity, while the most advanced practices are located at a higher level of maturity. An organization will typically mature from the lower to the higher maturity level as its practices improve. Lower-level practices must be implemented and sustained for higher-level practices to be achieved.

5. TASK ANALYSIS

According to (van Heijst et al., 1997), the first activity in the construction of a KBS is the definition of task analysis. Task analysis begins, at a high level, with the definition of an index of terms. This index includes words commonly used in software engineering (see Figure 2). 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 KBS user 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 (see Table 2) which will help identify a best practice related to the *S3^m*. 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. A close look at Figure 1 reveals that the 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*). In Table 2, step 11, 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 (see Table 2). 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 being used to help "diagnose" the current maintenance practice and map it to the maintenance model.

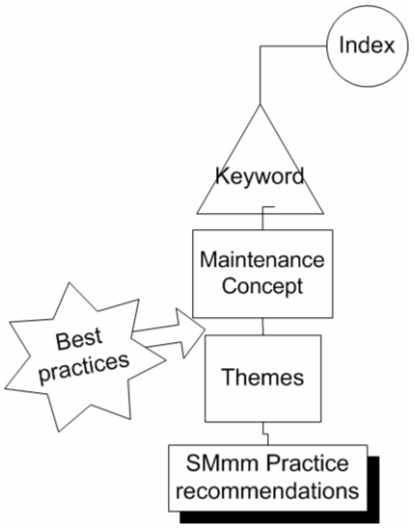
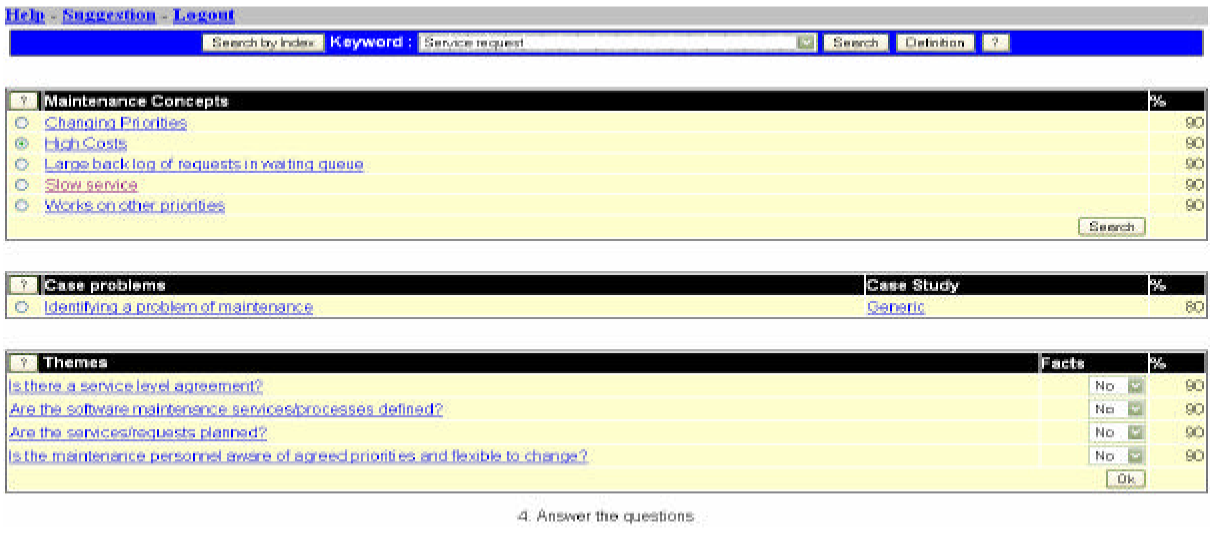


Figure 2: High-level view of ***SM^xpert***

Appendix A shows how the KBS helps the user answer the following question: How do we accept or reject a new maintenance request?

6. MEASURING USE OF CONCEPTS

The ***SM**^xpert^* KBS was built using Java script and XML, and supports the ***S3**^m^*. The architecture, design and implementation details of this KBS are similar to those of the COSMIC KBS (Desharnais, 2003), which was developed as a diagnostic tool to help IT personnel in the estimation of functional size. The design of the KBS is based on the use of both the case-based and ruled-based approaches (Desharnais et al., 2002). Figure 3 shows an example of the user layout presented to students when they experiment with the KBS. 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 ***SM**^xpert^*, while the expert interface gives the expert the option of adding new keywords, concepts, cases, themes and recommendations.


Help - Suggestion - Logout

Search by Index | Keyword : Service request | Search | Definition | ?

?	Maintenance Concepts	%
○	Changing Priorities	90
●	High Costs	90
○	Large back log of requests in waiting queue	90
○	Slow service	90
○	Works on other priorities	90

Search

?	Case problems	Case Study	%
○	Identifying a problem of maintenance	Generic	80

Themes	Facts	%
Is there a service level agreement?	No	90
Are the software maintenance services/processes defined?	No	90
Are the services/requests planned?	No	90
Is the maintenance personnel aware of agreed priorities and flexible to change?	No	90

Ok

4. Answer the questions

Figure 3: ***SM**^xpert^* user interface layout

Each time a user uses the knowledge base, an XML trace is kept of what was accessed during the transaction. Figure 4 shows the access trace left by the student user of the screen in Figure 3, who is looking for information about the "High Costs" of the service requests.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Sessions>
 <Session>
  <KW idRef="K1">
    <TC name="Read">
      <CP keyword="Service Request">
        <TH Concept="Hi costs" />
        <TH Concept="No" Question="Is there a
                         service level agreement" />
       <TH Fact="No" Question="Are the software
              maintenance service/processes defined" />
         <TH Fact="No" Question="Are the
                         services/requests planned" />
        <TH Fact="No" Question="Is the maintenance
                personnel aware of agreed priorities" />
          <REC cf="-100" />
            <Doc       href="/usr/local/jakarta-tomcat-
4.1.30/webapps/SMXpertNew/KB/CP/W/ecp_R_cha
nge_customer_retrieve_customer_data_W.xml;CF=-
100" />
            <Doc       href="/usr/local/jakarta-tomcat-
4.1.30/webapps/SMXpertNew/KB/TC/etc_R.xml;TH=
TH3" />
        </CP>
      </TC>
    </KW>
   <Doc href="KB/TC/etc_DG_MIS.xml" />
```

</Session> </Sessions>

The trace identifies the keyword and the concepts used. After numerous usages of the knowledge base, the following profile has been observed (see Figure 5):

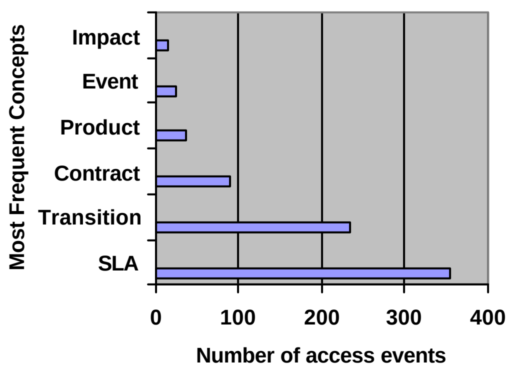


Figure 5: ***SM***^{xpert} concept usage

7. CONCLUSION AND FUTURE WORK

Measuring the concepts most often selected by knowledge base users is helpful in identifying where our effort should be spent in detailing both the support tools and the maturity model. Our hypothesis is that the concepts most frequently queried provide hints as to what are the greatest concerns of users. Our experience shows simply the feasibility of the approach, and the results should not be taken to constitute those of an industrial experiment. 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 measurement of KBS usage is useful for identifying the most important maintenance preoccupations of today's maintainers in industry.

REFERENCES

Abran, A., Moore, J. W., Bourque, P., Dupuis, R. and Tripp, L.,*Guide for the Software Engineering Body of Knowledge (SWEBOK)*, Ironman version, IEEE Computer Society Press: Los Alamitos CA,2004; 6-1-6-15, Montréal, http://www.swebok.org.

April, A., Abran, A. and Dumke, R. *SM*^{cmm} *Model to Evaluate and Improve the Quality of the Software Maintenance Process: Improvements, traceability and conformity to standards*, CSMR 2004 8th European Conference on Software Maintenance and Reengineering, (2004a) Tampere (Finland)

April, A., Abran A. and Dumke, R. *Assessment of Software Maintenance Capability: A model and its Design Process*, IASTED 2004, Conference on Software Engineering (2004b), Innsbruck (Austria)

CMMi (Ed.) (2002) *Capability Maturity Model Integration for Software Engineering (CMMi), Version 1.1,* CMU/SEI-2002-TR-028, ESC-TR-2002-028, Carnegie Mellon University.

Dekleva, S. M. *Delphi Study of Software Maintenance Problems*, International Conference on Software Maintenance (CSM 1992) (1992) IEEE Computer Society Press: Los Alamitos CA

Desharnais, J.-M., "Application de la mesure fonctionnelle COSMIC-FFP: une approche cognitive," in *Informatique, Thèse de doctorat en informatique cognitive*. Montreal: UQAM, 2003, pp. 187.

Desharnais, J.-M., A. Abran, A. Mayers, et T. Küssing, "Design of a diagnostic tool to improve the quality of functional measurement," document présenté à Proceedings of the 12th International Workshop on Software Measurement, 2002.

Desharnais, J.-M., Abran, A., Mayers, A., Buglione, L. and Bevo, V. *Knowledge Modeling for the Design of a KBS in the Functional Size Measurement Domain*, KES 2002, IOS Press, Crema, Italy

Desharnais, J. M., Abran, A., Mayers, A., Vilz, J. and Gruselin, F. (2004), *Verification and validation of a knowledge-based system*, KI, Special Issue on Software Engineering for Knowledge-based Systems, Germany, **3**.

Dias, M. G., Anquetil, N. and Oliveira, K. M. (2003), *Organizing the Knowledge Used in Software Maintenance*, Journal of Universal Computer Science, **9, 7** 64-658.

Durkin, J. (1994) *Expert system: Design and Development,* Prentice Hall, New York.

Kitchenham, B. et al. (1999), *Towards an Ontology of Software Maintenance,* J. Softw. Maint:Res. Parct., **11(6):**365-389.

Lientz, B. and Swanson, E. (1981), *Problems in Application Software Maintenance,* Communications of the ACM, **24, 11,** 763-769.

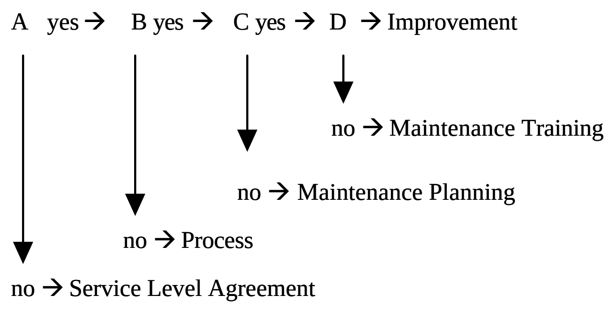
Ruiz, F., Vizcaino, A., Piattini, M. and Garcia, F. (2004) *International Journal of Software Engineering and Knowledge Engineering,* **14, 3** 323-349.

Uschold, M. and Jasper, R. (2001), *An ontology for the management of software maintenance projects*, In Industrial Knowledge Management: a micro-level approach,Bedford (UK), pp. 549-563.

van Heijst, G., Schreiber, A. T. and Wielinga, A., Using *Explicit Ontologies in KBS Development*, 2003 University of Amsterdam, Department of Social Science Informatics, Amsterdam, 1997

Vizcaíno, A., Favela, J. and Piattini, M. *A multi-agent system for knowledge management in software maintenance*, KES 2003 (2003), Springer Verlag, Oxford, UK.

Appendix A: Task description of the KBS using Dekleva's first problem

No.	TASK	EXAMPLE
1.	**Accessing the index**	The user enters a word that will identify a suggested keyword. As an example, the user enters: *Change in Priority*
2.	**Choosing a resulting keyword**	The user will enter a keyword that will help the KBS find the most closely related KPA and roadmap concepts. The system presents the following keywords: Change Management, Change Control, Staff Rotation, Event and Service Request, Service Level Agreement. The user chooses: *Event and Service Request*
3.	**Searching for a related software maintenance concept**	The KBS presents the maintenance concepts (which are related to the KPA and roadmap) to the user.
4.	**Giving priority to concepts**	The KBS will present the concepts in order of priority to the user. A percentage is related to each concept. The expert has previously established this percentage. As an example: *1) Event, 2) Process, 3) SLA, 4) Resource, 5) Change Control, and 6) Maintenance Manager.*
5.	**Choosing a maintenance topological concept**	The user chooses one or multiple maintenance concepts, *Event* in our example
6.	**Displaying themes**	With *Event*, there are 5 themes presented to the user in the forum of questions: A) Is there a Service Level Agreement ? B) Are the software maintenance services/processes defined ? C) Are the services/requests planned ? D) Are the maintenance personnel aware of agreed priorities and amenable to change?
7.	**Choosing the status of each theme**	The user will find facts for each practice (theme). He can answer yes or no to any of the themes.
8.	**Rating the status (facts)**	An algorithm based on Bayesian Theory (Uschold and Jasper, 2001) is used to calculate the rate (MYCIN approach). The algorithm rates the facts chosen.
9.	**Displaying the results**	The resulting percentage relating to the best request management is shown to the user.
10.	**Assessing the results**	The formula is based on Bayesian Theory, as explained by (Durkin, 1994). Case 1 – CF(CP) = CF(Theme1) = q_choice_perc*P_Q_perc Case 2 – CF(CP) = CF(Theme1) *CF(Theme2) Case 3 – CF1(Theme) = CFcombine[CF(Theme1, CF(Theme2)] CF(CP) = CFcombine[CF1(Theme), CF(Theme3)] Etc.
11.	**Recommendation/explanation**	 A yes → B yes → C yes → D → Improvement no → Maintenance Training no → Maintenance Planning no → Process no → Service Level Agreement The KBS will recommend the following solution (simplified for this paper):
12.	**Displaying other best practices**	Another part of the recommendation will show a different option, like: route request to account manager, interrupt work and insert in list of work, insert minor enhancement in list of work.
13.	**Displaying an explanation**	There is also the possibility of an explanation. In our case, the explanation takes up one page and could not be presented here due to lack of space (April et al., 2004b)
14.	**Acceptability**	Depending on the case that the user has to solve, the recommendation/explanation will be accepted or rejected. In our case, the user accepted the recommendation because it was not necessary to refer the change request to another group based on the criteria.
15.	**Choosing best practices (new)**	The process could start again. In this example, the user decided to stop because he considered he had enough information about the case. In a more complex situation, more choices could be necessary.