Knowledge Modeling for the Design of a KBS in the Functional Size Measurement Domain

Jean-Marc DESHARNAIS[1], Alain ABRAN[2], André MAYERS[3], Luigi BUGLIONE[4] and Valery BEVO[4]

[1] *Software Engineering Laboratory in Applied Metrics*,

Desharnais.jean-marc@uqam.ca,

[2] *Département de génie électrique - École de Technologie Supérieure (ETS)*

aabran@ele.etsmtl.ca

[3] *Département de Mathématique et d'Informatique - Université de Sherbrooke*

andre.mayers@DMI.USherb.CA

[4] *Software Engineering Management Research Laboratory - Université du Québec à Montréal*

luigi.buglione@computer.org, valerie.bevo@lrgl.uqam.ca

Abstract - This document presents the high-level design of a knowledge-based system to assist measurers in applying a functional measurement method consistently and systematically to often quite complex software applications which, moreover, may be from various application domains. The knowledge model underlying the proposed system is built on the key concepts of the software development process itself, as well as on the key concepts of a specific measurement method. The concepts describing the development process originate from the ontology of the SWEBOK [5] project and those related to the functional measurement method from the ontology of the COSMIC-FFP[6] method. The task originates from our modeling of the types of knowledge embedded in the measurement process.

1. INTRODUCTION

The application of a software functional measurement method is an intellectual process carried out on a set of complex abstract artifacts. This process includes both a mapping phase between the measurement model and a model of the software to be measured, and a measurement phase for the instantiation of the measurement rules to the derived model of the software. The application of a software functional size measurement method in general, and of the COSMIC-FFP method in particular, for a given set of software artifacts, is equivalent to solving a specific "problem", a measurement problem, from the measurer's point of view. The main parameters of the problem are the artifacts of the software (consisting of some outputs of the software development process) and the measurement method concepts and rules. The problem parameters need to be clearly identified and adequately interpreted. Then and only then can the problem be solved using appropriate rules. Figure 1 presents the measurer's cognitive path for solving the problem [1]:

- In the understanding phase, the measurer has first to gain an adequate understanding of both types of parameters of the problem at hand.
- In the interpretation phase, the measurer must identify, from among the artifacts, what is relevant to the measurement according to measurement method concepts.

- In the next or 'using' phase, the measurer must establish a link between an ontology related to the software development process and an ontology related to the measurement method. According to Grüber, "*an ontology is an explicit specification of a conceptualisation*" [2].
- In the last phase, to solve the measurement problem, the measurer must rely on his or her implicit knowledge of the software development process and on his or her knowledge of the various measurement tasks that must be performed to solve the problem.

It is challenging for any measurer to apply a functional measurement method consistently and systematically to often quite complex software applications and which, moreover, may be from various application domains. To support the measurers, we propose here a knowledge-based system (KBS) to improve the quality of the measurement results, and, of course, the performance of the measurers in terms of ease of use and consistency of the measurement results. Such a KBS can naturally facilitate, and improve, the teaching of such a measurement method. This paper discusses the knowledge modeling methodology used to map the software development process concepts to the functional measurement method tasks, and to embed it into a KBS.

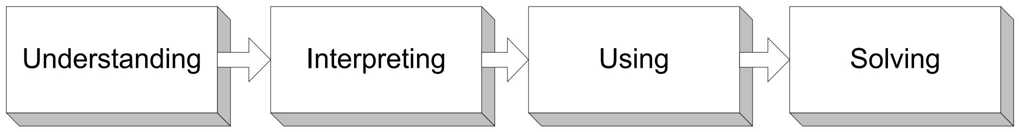

Figure 1: Measurer's cognitive path

2. OVERVIEW OF KNOWLEDGE MODELING

To understand the design of this KBS, it will be useful to describe all these types of knowledge more explicitly. In this section, we take a look at the relationships between the different types of knowledge identified in the literature and the measurer's path for solving the problem. According to van Heijst [3], there are at least five different types of knowledge to be taken into account when constructing a KBS: tasks, problem-solving methods, inferences, ontologies and domain knowledge. For each knowledge type, we provide examples of the elements we have integrated into the design of the KBS for software functional size measurement in order to take into account all these types of knowledge.

2.1 Tasks

Tasks constitute what must be achieved during problem-solving. For the measurer, the goal is to identify the occurrences of the different concepts of the measurement method based on the accessible artifacts from the software documentation (or from the developer's comments). Table 1 presents the set of tasks we identified as required to support the measurer, and which were integrated into the KBS.

2.2 Problem-solving

According to Heijst [3], "problem-solving methods are ways to achieve the goals described in tasks". In our KBS, we included the possibility of accessing a list of keywords, and their definitions, to facilitate the use of keywords for finding a topological concept (e.g. subtask of the task "entering a keyword").

2.3 Inferences

Inferences describe the primitive reasoning steps in the problem-solving process. The inferences form a functional model which is sometimes called the inference model or inference structure. For the measurer, inferences are the basic reasoning steps related, for example, to the search for the topological concepts and to ordering them based on the keyword.

2.4 Ontology

An ontology describes the structure and the vocabulary of the static domain knowledge. For the measurer, the relevant ontologies are these related to the software development process and those related to the measurement method to be applied. For instance, a well-known ontology of the software development process is the waterfall model. The waterfall model of the software life-cycle was constructed in such a way that "*the successful completion of one of the life-cycle phases in the waterfall chart corresponds to the achievement of the counterpart goal in the sequence of program engineering goals for the software process*" [4, pp. 36-37]. In practice, various authors have presented their own ontologies of the software development process. Furthermore, each such ontology of the software development process is instantiated differently in different organizations.

For the construction of a KBS, it is therefore useful to select the development process ontology that has the broadest base possible, and the strongest consensual level. The Guide to the Software Engineering Body of Knowledge (SWEBOK) is currently the most relevant ontology for this purpose: it has been built "*to provide a consensually-validated characterization of the bounds of the software engineering discipline and to provide a topical access to the Body of Knowledge supporting that discipline* [5, p. i]. We have therefore elected to use the structure and the vocabulary adopted in the SWEBOK document to define the domain knowledge of the software development process, as well as the new COSMIC-FFP measurement method [6] which is currently undergoing ISO standardization.

2.5 Domain knowledge

Domain knowledge refers to a collection of *statements* about a domain; in our context, the relevant domain knowledge is the integration of the software measurement domain and the software engineering development process model. These are the *statements* of the experts in the domain of functional measurement for mapping the artifacts of the software to be measured and the measurement rules. For example, the measurer will make the following statement to map the functional process concept in COSMIC-FFP [6] and the process model in SWEBOK [5]: "It is possible to find a functional process in a process model". Such statements identify a topological concept which will be used to identify a case problem using another statement. Finally, a statement will link the case problem and the themes at the lowest level. It is possible with this method to build a knowledge base with cases and diagnostic rules used by the measurement experts to measure the software.

3. MODELING PROCESS

Van Heijst [3] suggests the following approach for building a knowledge model:

- Construct a task model for the KBS;
- Select and configure appropriate ontologies, and, if necessary, refine them;

- Map the application ontology onto the knowledge roles (Figure 1) in the task model;
- Instantiate the application ontology with the domain knowledge.

3.1 Constructing a task model for the KBS

According to van Heijst, the first activity in the construction of a KBS is task analysis. This concept of the task analysis has already been established for our KBS and is presented in Table 1.

Table 1: Task description for the KBS for functional size measurement

No.	TASK	DESCRIPTION
1.	**Entering a keyword**	The measurer will enter a keyword that will help the KBS find the topological concepts related to the case problem
2.	**Searching for a topological concept**	The KBS will present the topological concepts to the measurer
3.	**Giving priority to topological concepts**	The KBS will present the topological concepts in order of priority to the measurer
4.	**Choosing a topological concept**	The measurer chooses one or multiple topological concepts
5.	**Finding a case problem**	The KBS will find the case problems related to the topological concepts chosen by the measurer
6.	**Giving priority to the case problems**	The KBS will present the case problems in order of priority to the measurer
7.	**Choosing case problems**	The measurer will choose the case problems corresponding with his/her interpretation of the problem
8.	**Displaying themes**	The KBS will show all the themes related to the case problems to the measurer
9.	**Responding to the themes**	The measurer will find facts for each theme
10.	**Rating the facts**	An algorithm will rate the facts chosen
11.	**Displaying the results**	The percentage will be presented to the measurer
12.	**Assessing the results**	The KBS will assess the results based on the heuristics
13.	**Recommendation/explanation**	The KBS will recommend either a solution to each case problem, another case problem and/or an explanation for the case problem not being solved
14.	**Displaying other case problems**	The KBS will suggest one or more new case problems to the user
15.	**Displaying an explanation**	The KBS will explain the solution, if necessary
16.	**Acceptability**	The measurer will decide whether or not the recommendation is acceptable
17.	**Choosing case problems (new)**	The measurer will choose another case problem, either one already suggested by the KBS or his or her own.

3.2 Selecting and configuring the KBS ontology

In this KBS for functional size measurement, the measurement rules are already included in the ontology of the COSMIC-FFP method [6], and the development artefacts to be measured are in the SWEBOK ontology. The following concepts from the ontology of the COSMIC-FFP method are to be used: layer, user, boundary, data group, triggering event, process, sub-process (entry, exit, read and write). The number of concepts in SWEBOK [5] is, of course, too large to be listed here. These concepts can, however, be identified throughout the main development phases: software requirements, software design, software construction, software testing and software maintenance.

3.3 Mapping the task model onto the KBS ontology

The next step consists in mapping the COSMIC-FFP ontology onto the SWEBOK development process ontology. Our KBS ontology defines the concepts relevant to COSMIC-FFP and SWEBOK. According to van Heijst, "*when performing a generic task, instances of a particular concept typically fulfill particular roles in problem solving.*" [3]. The measurer plays a direct role in the various tasks (Table 1). First, the measurer should understand from the artifact the type of problem he or she needs to interpret and enter a keyword (e.g. process) (task 1). Then, topological concepts will be suggested by the KBS (task 2). For example, a combination of the concept "process model" (SWEBOK) and the concept (functional process) will suggest the topological concept "process modeling". The measurer will decide which one to use. This topological concept is related to a number of case problems. One of the case problems will be "identification of a process". The measurer will then solve the case problems by choosing the facts that will validate or invalidate the existence of a functional process (task 3). The final role of the measurer will be to accept or reject the recommendation proposed by the KBS (task 4).

Table 2 Definition of process model (SWEBOK)

Process model [5, p.2-8, section 2.1.1]

Definition: Provide an understanding that the requirements engineering process:

- is not a discrete front-end activity of the software life-cycle, but rather a process that is initiated at the beginning of a project and continues to be refined throughout the life-cycle of the software process;
- must identify requirements as configuration items, and manage them under the same configuration regime as other products of the development process;
- will need to be tailored to the organization and project context.

Table 3 Definition of functional process (COSMIC-FFP)

Functional process: A functional process is a unique set of data movements (entry, exit, read, write) implementing a cohesive and logically indivisible set of Functional User Requirements. It is triggered directly by an Event (-type) (or indirectly via an 'actor') and is complete when it has executed all that is required to be done in response to the triggering Event (-type) (COSMIC-FFP Measurement Manual, page 8)

Table 4: Example for an identification of a case problem

Case problem	Themes	Facts
Identification of a functional process	Number of triggering events	a) At least one triggering event
		b) There is one and only one triggering event
		c) There is no triggering event
	Nature of the triggering event	a) Trigger by a human user
		b) Trigger from another piece of software
		c) Trigger by a clock
		d) Trigger from another layer

The case problem originates from a topological concept. The themes and the facts help to solve the problem using the topological concept, which are the final two steps of the measurer's path (Figure 1).

3.4 Instantiating the KBS ontology

While our KBS ontology defines which concepts are used in the domain (software development/functional measurement), the measurement expert, using his or her knowledge, describes the instances of these concepts. According to van Heijst, "*besides the reusability aspect, one of the main arguments for distinguishing between the ontology and the application knowledge is that the domain knowledge must by definition be defined by the expert.*" [3]. The domain knowledge is related to the keywords for finding a topological concept, to the topological concept for finding case problems, and to the themes created by the expert for solving case problems.

4. CONCLUSIONS

This paper has presented the design of a knowledge-based system (KBS) to assist measurers in applying consistently and systematically a functional measurement method to software applications. The knowledge model underlying the proposed KBS is based on the key concepts of the software development process itself, as well as on the key concepts of a specific measurement method. It also takes into account the two types of knowledge required by the measurer and the modeling methodology used to integrate them into the design of the KBS. We have also presented some of the concepts used to populate a KBS, focusing specifically on the COSMIC-FFP method, as well as on the IEEE Software Engineering Body of Knowledge (SWEBOK) as the reference for the ontology (vocabulary and structure) of the Software Life Cycle phases and related concepts. The paper has also presented the set task of the KBS for functional size measurement. The design of such a KBS has already been completed, and its construction is in progress; an operational prototype is already operational and populated with more than 100 case problems. Experimentation with experts and novices will start in the summer of 2002.

5. REFERENCES

[1] Desharnais J.-M., Abran A., Applying a Functional Measurement Method: Cognitive Issues, International Workshop on Software Measurement - IWSM 2001, in Current Trends in Software Measurement, Shaker Verlag, Aachen (Germany), 2002, pp. 26-50.

[2] Grüber T.R., A translation approach to portable ontology specifications, Knowledge Acquisition, 5:199-220, 1993.

[3] Van Heijst G., Schreiber A.Th. & Wielinga B.J., Using Explicit Ontologies in KBS Development, 1997.

[4] Boehm, B., Software Engineering Economics, Prentice Hall, 1981.

[5] Abran, A., Moore, J., Bourque, P., Dupuis, R. L. Tripp, L., Guide to the Software Engineering Body of Knowledge – SWEBOK, Trial Version, IEEE-Computer Society Press, Dec. 2001, URL: http://www.swebok.org

[6] Abran, A, Desharnais, J.-M., Oligny, S., St-Pierre, D.,Symons, C., COSMIC-FFP Measurement Manual, version 2.1, May 2001, URL: http://www.lrgl.uqam.ca/cosmic-ffp