Design of a diagnostic tool to improve the quality of the functional measurement

Jean-Marc DESHARNAIS[1], Tim Küssing[2], Alain ABRAN[3] André MAYERS[4]
[1] Software Engineering Laboratory in Applied Metrics,
Desharnais.jean-marc@uqam.ca,
[2] Department of Data and Information Technology
University of Applied Science, Nuremberg (Germany)
tim.kuessing@gmx.de
[3] Département de génie électrique
École de Technologie Supérieure (ETS)
aabran@ele.etsmtl.ca
[4] Département de Mathématique et d'Informatique
Université de Sherbrooke
andre.mayers@DMI.USherb.CA

Accepted for publication at IWSM 2002 – Magdeburg (Germany)

Abstract - This document presents the design of a diagnostic tool to assist measurers in applying consistently and systematically a functional measurement method. The design of the diagnostic tool is based on the UML (Unified Mark-up Language) method [7] and a specific application of van Heijst knowledge modeling method [3]. The result is a hybrid diagnostic tool using CBR and rule based techniques.

1. INTRODUCTION

The application of a software functional measurement method is an intellectual process carried out on a complex abstract artifacts: this process includes both a mapping phase between the measurement model and a model of the software, and a measurement phase for the instantiation of the measurement rules to the derived model of the software to be measured. The application of a software functional size measurement method in general, and 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 (these artifacts are some outputs of the software development process) and the measurement method concepts and rules. To produce a quality measure i.e. to insure the accuracy[1] of measurement and the repeatability[2] of the results of measurements, the parameters of the “problem” need to be clearly identified, adequately interpreted, then and only then the “problem” can be solved using appropriate rules. Figure 1 presents the measurer’s cognitive path for solving the “problem” [1, 10]:

- In the understanding phase, the measurer has to gain an adequate understanding of both types of parameters of the problem on hand.
- In the interpretation phase, the measurer must identify the artefacts which are relevant to the measurement according to measurement method concepts.
- In the 'using' phase, the measurer must next establish a link between an ontology related to the software development process and ontology related to the measurement method. According to Grüber, “ontology is an explicit specification of a conceptualization” [2].
- In the last phase, solving his measurement “problem”, the measurer must rely on his implicit knowledge about the software development process and on his knowledge about the different measurement tasks he must perform to solve his “problem”.

It is challenging for any measurer to apply consistently and systematically a functional measurement method to software applications that can be quite complex and/or from various application domains. To support the measurers, we propose here a diagnostic tool 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. This paper discusses the knowledge modelling methodology use to design the framework to map the software development process concepts to the functional measurement method tasks, and to embed it into a diagnostic tool.

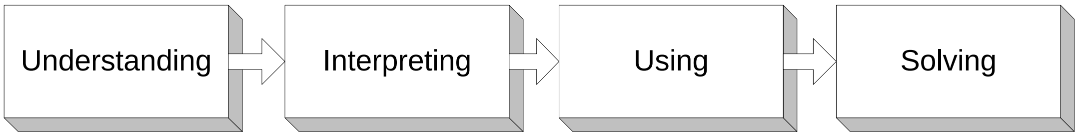

Figure 1: Measurer cognitive path

---

[1] Accuracy of measurement: closeness of the agreement between the result of a measurement and a true value of the measurand (qualitative) [9].

[2] Repeatability of results of measurements: closeness of the agreement between the results of successive measurements of the same measures carried out under the same conditions of measurements (quantitative) [9].

2. CLASS DIAGRAM, USE CASES AND SCENARIOS

A class is “a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment” [7]. There are a number of classes in our class diagram (figure 2) (“diagram that shows a collection of declarative - static- model elements, such as classes, types, and their contents and relationships” [7]:

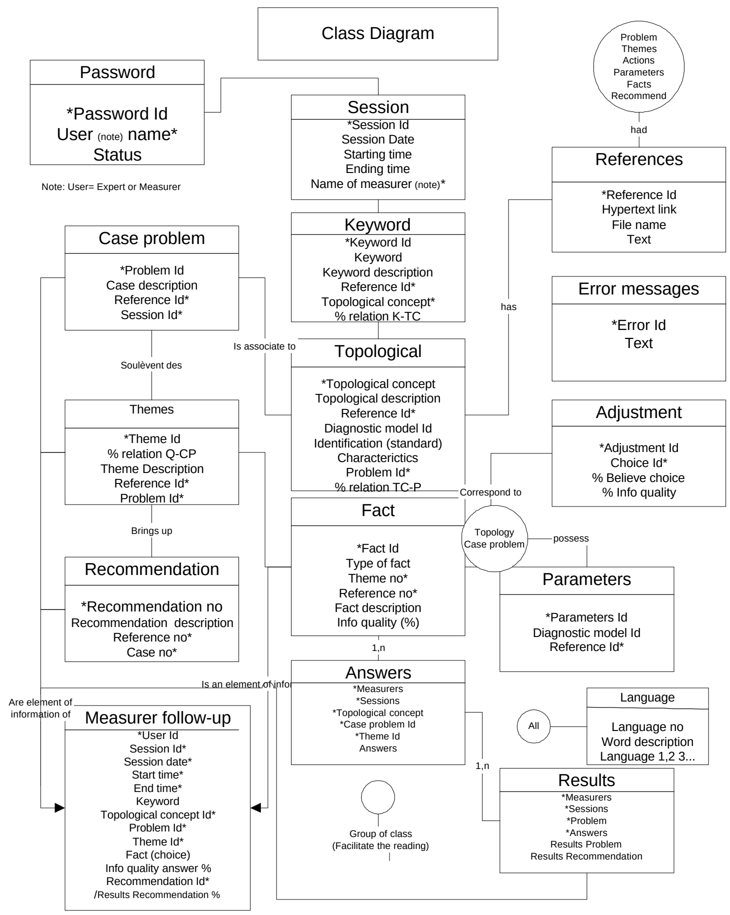

Figure 2 Class Diagram

A use case is “the specification of a sequence of actions, including variants that a system (or other entity) can perform, interacting with actors of the system. Our use case diagram (diagram that shows the relationships among actors and use cases within a system) is the following:

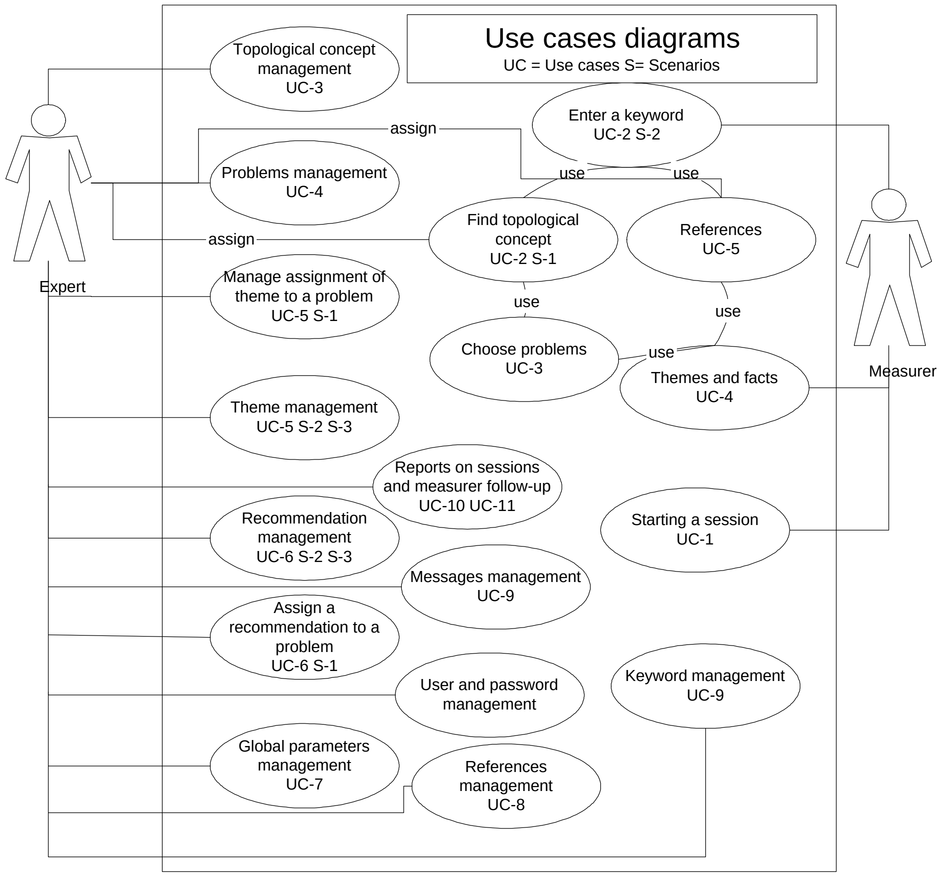

Figure 3 : Uses case diagram

There are two actors (or agent): the measurer and the expert. An actor is "a coherent set of roles that users of use cases play when interacting with these use cases. An actor has one role for each use case with which it communicates" [7].

A scenario is "a specific sequence of actions that illustrates behaviors". There are many scenarios in our software. This is an example describing the registration of a measurer in a session.

Use case 1: Measurer registration to a session
Scenario 1: Session registration
<u>Description</u> A screen allowing to enter the identification of the measurer and the password Primary education actor: Measurer Secondary actor No Pre condition No <u>Short description</u> The measurer enter his name (recognized by the software) and his password. The identification of the session is created automatically by the software. <u>Exception</u> If the name and the password do not correspond to the content of the class password, there is an error message Post-condition (rules of termination) Access to the software <u>Classes used:</u> session, measurer <u>Data exchanged:</u> Identification of the measurer, password, identification of the session User interface: see table 1
Calculation Yes: No: <u>X</u>

Table 1 Example of a scenario

3. KNOWLEDGE MODELING

To ensure consistency of the measurement results (and of the teaching of such a measurement method) it is useful to describe in a more explicit manner both types of knowledge. 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 diagnostic tool: tasks, problem-solving methods, inferences, ontologies and domain knowledge. For each type, we provide in [8] examples of the elements we integrated into the design of the tool for software functional size measurement.

4. TASK MODEL FOR ESTABLISHING A DIAGNOSTIC TOOL

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

- Construct a task model for the diagnostic tool;
- 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 domain knowledge.

For now we will concentrate on the construction of the task model and then show in the next section, as an example, how the user interface looks like. Selection, mapping and instantiation are presented in [8].

No.	TASK	DESCRIPTION
1.	**Entering a keyword**	The measurer will enter a keyword that will help the tool find the topological concepts related to the case problem
2.	**Searching a topological concept**	The tool will present the topological concepts to the measurer
3.	**Giving priority to topological concepts**	The tool 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 tool will find the case problems related to the topological concepts chosen by the measurer
6.	**Giving priority to the case problems**	The tool 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 tool will show all the themes related to the case problems to the measurer
9.	**Answering the themes**	The measurer will find facts for each theme
10.	**Rating the facts**	An algorithm will rate the fact chosen
11.	**Displaying the results**	The percentage will be presented to the measurer
12.	**Assessing the results**	The tool will assess the results based on the heuristics
13.	**Recommending**	The tool will recommend either a solution to each case problem, another case problem and/or an explanation to the case problem not solved
14.	**Displaying others case problems**	The tool will suggest one or more new case problems to the user
15.	**Displaying an explanation**	The tool will give an explanation about the solution if necessary

[3] In the context of our project, the way we used van Heijst approach is not as generic as the way he propose it.

16.	**Acceptable**	The measurer will decide if the recommendation is acceptable
17.	**Choosing case problems (new)**	The measurer will choose another case problem, either one already suggested by the tool or his own.

Table 2: Task Model

The link between each task is the following:

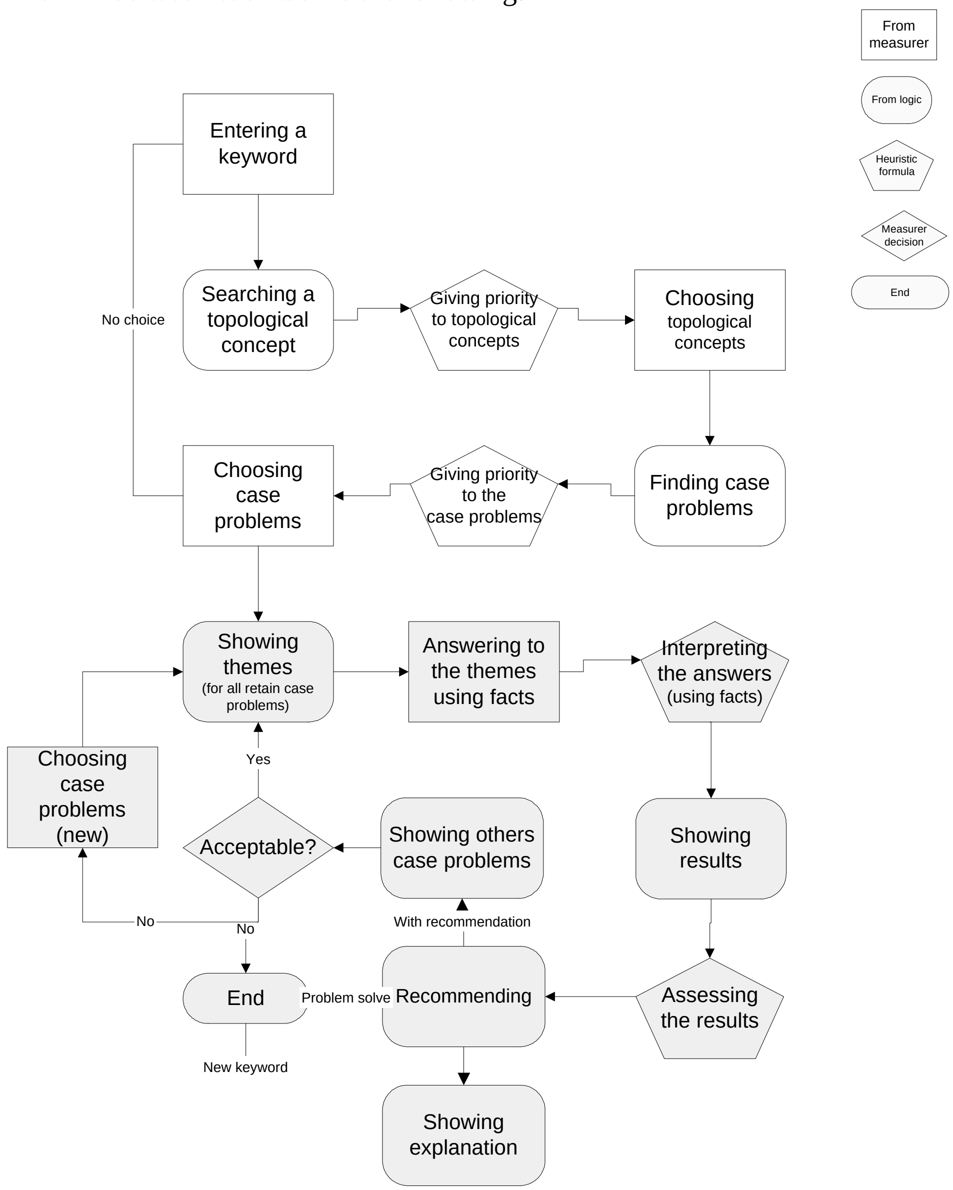

Figure 4 Task model

Figure 4 shows the dynamic of the role of the measurer when going through each task. Each square box shows where the measurer needs to intervene (entering a keyword, choosing topological concepts, choosing case problems, answering to the themes using facts). The first part is more CBR type, while the second part is rule based. There are heuristics formulas represented by a pentagon (giving priority to topological concept, giving priority to the case problems, interpreting the answers, assessing the results). Some of them used certainty theory formulas proposed in MYCIN.

5. MEASURER INTERFACE

The measurer, using the interface, is following the task model.

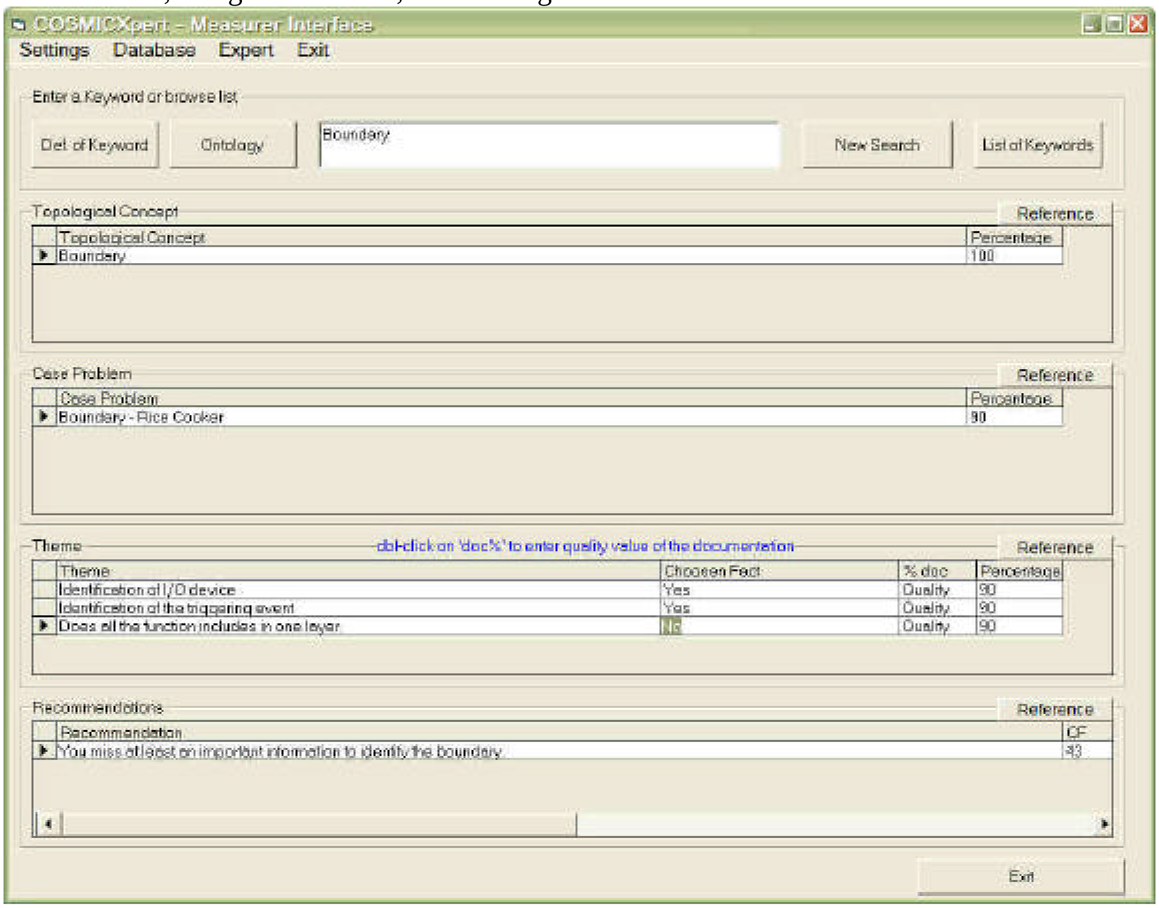

Figure 5 Measurer interface

The measurer selects a list of keyword (right at top) and then the interface populates topological concept, case problem and theme. The measurer could choose which topological concept he/she wants to keep. If so, this will change the list of case problems. Again the measurer can choose case problem to keep. This will affect the themes. The measurer will then choose the facts appropriated to each theme. This will lead to a calculation to provide recommendations with a percentage of probability. The inference used is based on certainty theory. We do not have enough space to explain it in this article. There are also sub tasks not describe. For example, it is possible for the measurer to have a definition of the key concepts using the ontology button. It is also possible to obtain a definition of each keyword. Also, the interface can be interchangeable, using different languages.

6. 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 ontologies 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, JM, Oligny, S., St-Pierre, D.,Symons, C.,COSMIC-FFP Measurement Manual version 2.1, May 2001, URL: http://www.lrgl.uqam.ca/cosmic-ffp

[7] Rational Software Corporation, UML Semantics Appendix M1-UML Glossary, version 1.0 , 13 January 1997

[8] Desharnais J.M., Abran, Mayers A., Buglione L., Bevo V., Knowledge Modeling for the Design of a KBS in the Functional Size Measurement Domain, KES 2002, Italy, 7 pages.

[9] Vocabulaire international des termes fondamentaux et généraux de métrologie Accord international sur la terminologie, résultant de la collaboration entre des experts nommés par le BIPM, la CEI, la FICC, l'ISO, l'OIML, l'UICPA, et l'UIPPA, 60 p., bilingue1993.

[10] Desharnais J-M, Abran A., Mayers A., Buglione L., Bevo V., A Knowledge-Based System in Functional Size Measurement, Departemental document, University of Quebec in Montreal, 2002, 15 p.

On the WEB:
http://smi-web.stanford.edu/projects/history.html#MYCIN