A Web-Based Software Engineering Measurement Expert System

Qing He, Yingxu Wang, Behrouz H. Far, and Shuangshuang Zhang

Theoretical and Empirical Software Engineering Research Centre
Dept. of Electrical and Computer Engineering
University of Calgary
2500 University Drive, NW, Calgary, Alberta, Canada T2N 1N4
Tel: (403) 220 6141, Fax: (403) 282 6855
Email: {heq,wangyx}@enel.ucalgary.ca

Abstract

The software measurement expert system tool (SEMEST) is a rule-based and web-enabled application for multiple software engineering measurement and analysis. SEMEST is designed on the basis of the software Engineering measurement system (SEMS) that provides a comprehensive set of software measures and metrics in a formal and consistent framework. The Module View Controller (MVC) design model is adopted for implementing SEMEST. As a multiple-layer web-based system with an expert inference engine and a knowledge base, SEMEST supports goal-, process- and category-oriented measurement and analysis in software engineering.

Keywords: *Software engineering, measurement, metrics, expert systems, knowledge base, web application*

1. INTRODUCTION

With the increasing importance of quantitative software engineering, a plenty of software measures and metrics has been proposed [1, 3-10]. A software engineering measurement expert system tool (SEMEST) is developed to assist users to apply quantitative measurement and control of software engineering projects. SEMEST is designed on the basis of a software engineering measurement system (SEMS) [8, 10].

By identifying software measurements in formal and rigorous way, SEMS provides a comprehensive and consistent measurement framework. SEMEST defines the relationships between measurement goals, processes, technology keywords, and metrics. The existence of these links makes a measurement program more understandable, modifiable, and reusable. The association between goal/ process/keywords and a large set of metrics is represented in a knowledge base and manipulated by an expert inference engine of SEMEST.

In this paper, we first briefly describe the SEMS framework. Then, the design and implementation of SEMEST are described, focusing on the construction of the software engineering measurement knowledge base and the design of the inference engine of SEMEST.

CCECE 2003 – CCGEI 2003, Montreal, May/mai 2003
0-7803-7781-8/03/$17.00 © 2003 IEEE

2. THE SOFTWARE ENGINEERING MEASUREMENT SYSTEM (SEMS)

SEMS models a comprehensive set of software engineering metrics with more than 300 meta and derived measures [8, 10]. Major characteristics of SEMS are summarized as follows:

- SEMS models all software engineering measures and metrics formally and rigorously. Each measure adopted in SEMS is defined by a formula.
- SEMS provides a wide rang of metrics covering software product measurement, software engineering process measurement, and software engineering project predictive and estimative measurement.
- SEMS creates a consistent and unified framework for software engineering measurement in a tree-like hierarchy. This forms a rich knowledge base for the implementation of the SEMEST expert tool to satisfy needs to practical measurement programs.

SEMS divides software measurements into two categories: the meta and the derived measures. The meta measures are primary measures that can be independently conducted. The derived measures are indirect measures that relates to one or more meta measures defined by a given function. In SEMS, all measures and their relations have been described by algebra formulae.

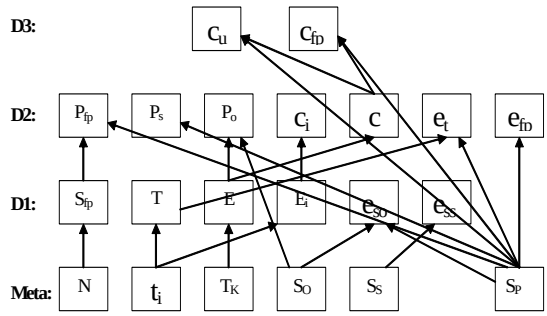

Fig. 1. The structure of SEMS

A part of SEMS' structure is shown in Fig. 1,where meta measures such as physical source code S_p, process time t_i, source memory size S_o, and etc. are the fundamental measures. Based on the combination of them, derived measures are obtained as shown at layers *D1*, *D2* and *D3*. Among them, for example, total development effort *E*, source code space efficiency e_{ss} , total cost C, and unit code cost C_u are at different levels depending on their relationships with other meta or derived measures. For example, based on the derived measure *E* that is obtained by work time T_k and number of programmers, the total development cost *C* and source code productivity *Ps* can be derived at higher levels. Based on the SEMS hierarchical framework, effort needed for data collection will be dramatically reduced since the data required are only restricted to the meta measures. SEMS provides an algebraic system that enables the development of new metrics in the future, useful for improved understanding of software engineering processes.

3. THE DESIGN OF SEMEST

3.1 The Architecture of SEMEST

SEMEST is a web-based software engineering measurement system designed to assist less experience project managers to carry out a measurement program more accurately and efficiently. The expert system functions with an expert inference engine and a comprehensive knowledge base of software engineering measurement. An overview of the SEMEST framework is shown in Fig. 2. By surveying industrial experience and interviewing experts in software measurement area, SEMEST models 10 goals, 7 processes and 24 technical keywords in software engineering. Users only need to choose from options of business goals, processes, or technical keywords, a full list of measures and metrics will be recommended by the expert system. After users determine the final list of measurement, required data to be collected and entered will be promoted, measures will be automatically calculated, and results will be illustrated.

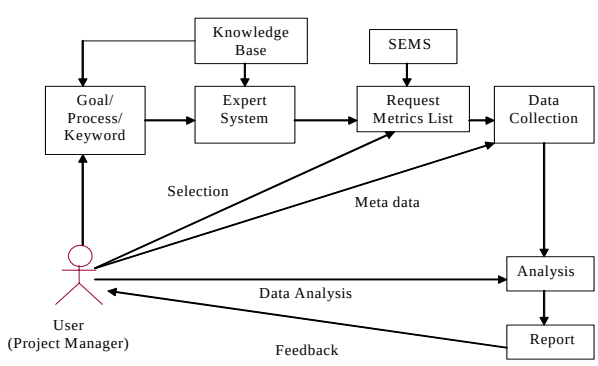

Fig. 2. The framework of SEMEST

3.2 The Knowledge Base of Software Engineering Measurement

Knowledge base is deemed as a database with structured knowledge about a field of expertise. An expert system that uses the knowledge base is able to solve problems by using the knowledge derived from experts in the field [2, 8]. SEMEST encompasses a comprehensive knowledge base for software engineering measurement, built on the basis of SEMS.

3.2.1 Structure of the Knowledge Base. The SEMEST inference engine is rule-based that operates on the measurement knowledge-base. The rule-based inference engine supports *goal-*, *process-*, and *keyword-driven* measurement rules as defined below:

Goal-driven rules: $R_G: G \times W \rightarrow M$ (1.1)
Process-driven rules: $R_P: P \times W \rightarrow M$ (1.2)
Keywords-driven rules: $R_K: K \times W \rightarrow M$ (1.3)

where *G*, *P*, and *K* represent the sets of goals, processes, and keywords respectively, *W* is a set of weighted thresholds, and *M* a set of measures.

In the above rules, a set of measures, *M*, is defined as a triple, i.e.:

$$M = \langle Cat\#, ID, W \rangle \quad (2)$$

where *Cat#* is a category number of a specific measure, *ID* the name of the measure, and *W* the weight defined as a function between goal/process/keyword and related measures as defined in the knowledge-base, i.e.:

$$W = f(G, M) \mid f(P, M) \mid f(K, M) \quad (3)$$

3.2.2 Knowledge acquisition and inference algorithm. Knowledge acquisition of SEMEST starts by a survey of experience and general goals in measurement programs in the software industry. Then, the SEMS measures and metrics are organized into hierarchical categories that help experts to concentrate on a coherent part of metrics at a given time. Several experts were asked how necessary to use a given measure corresponding to a certain project goal. A weight between 0 to 10 is assigned to every measure in SEMS (0 represents absolutely not relevant, and 10 absolutely relevant). Since experts' evaluations may be varying upon certain goal on the basis of their own experience, weighted averages are taken for all measures in the knowledge base.

After the establishment of the knowledge base, feedback and modification are taken to refine the knowledge base. The knowledge base of SEMEST consists of two components: goal-metrics mapping and process-metrics mapping. There are more than 5,000 expert weights are elicited to build the knowledge base. An example of a part of the knowledge base is provided in Table 1.

The selection of goals/processes/keywords is based on a confidential level of k%, which shows how confident is

for a specific measure related to a given goal, process, or keyword. The inference algorithm of SEMEST can be defined by a selection of relevant measures for a given goal by the following rules:

$$\text{If } W(G,W) \geq k\%,\ 0 \leq k \leq 100 \text{ then metrics sets are selected} \quad (4)$$

Tailoring of the record for the recommended measures can be done by users to meet specific needs.

Table 1. Example of the SEMEST Knowledge Base

Cat.	Meta/Derived Measures	P1	P2	P3	P4	P5	P6	P7
1.1	5							
1.1.1.1	Physical source size (M)	1.7	2.2	2.1	8.2	4.3	5.7	5.7
1.1.2.1	Source memory size (M)	1	1.7	1.5	5.6	2.7	4.2	4.1
1.1.2.2	Object memory size (M)	1	1.5	1.2	5.5	2.7	4.1	4
1.1.3.1	Cognitive functional size (D)	1	2.5	4	6.2	4.8	5.1	5.5
1.1.3.2	Function-point size (D)	1.7	1.7	2.7	5.2	3.7	3.7	3.2
1.2	5							
1.2.0.1	Code functional efficiency (D)	2.7	2	3.7	7.7	3.7	3.5	2.2
1.2.0.2	Code function point efficiency (D)	2.6	1.8	3.3	6.3	3.6	3.1	2
1.2.0.3	Code time efficiency (D)	3	1.7	3	6.6	3.5	3.2	2
1.2.0.4	Source space efficiency (D)	2.8	1.6	2.7	5.8	2.6	2.3	1.7
1.2.0.5	Object space efficiency (D)	3.1	1.8	2.8	5.3	3.1	2.6	2

3.3 The Web Patterns of SEMEST

SEMEST is a web-based tool to help multiple users to share measurement services. We adopt the MVC 2 design model to implement the web application of SEMEST, where *M* refers to model that maintains data throughout the application, *V* means view that extracts data from the model and present data to the user, and *C* represents the controller via it users' requests are handled and actions are delegated to the appropriate process(es). Fig. 3 shows the structure of the MVC2 model of SEMEST.

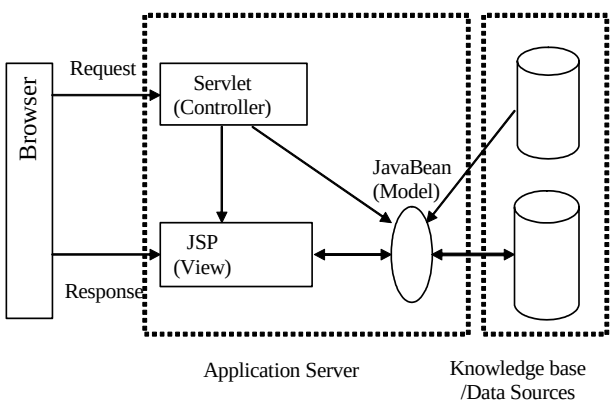

Fig. 3. The MVC2 design model of SEMEST

Java is adopted to implement the MVC2 design model. User requests are handled by Java servlets. Java bean is used for maintaining both request information and data/knowledge bases. JSP is used to communicate with users via the Internet. There are two databases in SEMEST: one is for the knowledge base and another is for measurement records. By using the measurement record database both meta and derived measures can be manipulated for multiple users and projects.

4. THE IMPLEMENTATION OF SEMEST

The development of SEMEST is based on the J2EE structure. TOMCAT is used as the web server. Users can browse the SEMEST web site for measurement services or learning detailed measurement technologies. SEMEST provides explicit link between metrics and project goals through the expert inference engine. This tool can significantly reduce complexity and effort in project measurement and lower the requirements for measurement knowledge and skills.

4.1 Project Management

SEMEST is implemented to support project management of measurement recommendations and results. The project management subsystem provides users for measurement services as a specific project. This subsystem has the following functions:

- *Add new project*: To register users to SEMEST for a new project.
- *Delete project*: To delete an existing project and clean up related data and records.
- *Project record*: To display and modify project logs.
- *Retrieve project*: To retrieve measurement results for a project.
- *Import data*: To upload measurement data in an EXCEL file.
- *Export data*: To export data from a project record to an EXCEL file.

4.2 Measurement Recommendation

The measurement recommendation subsystem provides a list of measures for a given goal, process, or keyword by the expert engine of SEMEST. Measures are displayed according to their weights related to user requirements.

The recommended measures in the list are a subset of the SEMEST knowledge base. Two ways are provided to sort the recommended measures: by category or by weights of the measures. The user can tailor the list according to project, resource, and environment factors. A resulted set of measures checked by the user will be used to prompt detailed methods and data collection guidelines. If there are inter-relationship between the measures in the final list, SEMEST can automatically analyze them, and data required for mutually related meta measures are only need to be entered once.

4.3 Measurement results

The measurement results and analysis subsystem shows meta data input, calculates results of derived measures, and

interprets measurement results. Fig. 4 shows a snapshot of the measurement result and analysis subsystem in SEMEST.

By using the SEMEST expert system, users who may not be very familiar with measurement technologies can benefit from the system and to be able to conduct a measurement project. SEMEST is also capable of assisting users in collecting data. Meta data is necessarily collected by users, but the data for derived measures are calculated by the SEMEST tool and automatically stored in the project database. Because the number of derive measures is far more than that of the meta measures in SEMEST, the tool can contribute greatly to the efficiency of carrying out a measurement project.

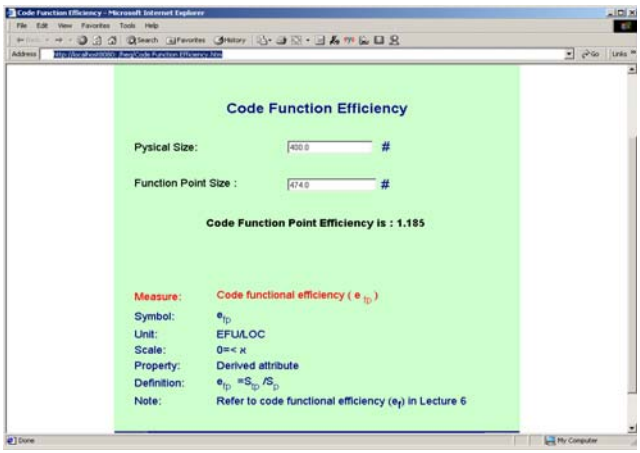

Fig. 4 The measurement results of SEMEST

5. CONCLUSIONS

This paper has presented a web-based Software Engineering Measurement Expert System Tool (SEMEST). SEMEST has been designed as a multi-layer web-based expert system with a rule-based inference engine and a knowledge-base. SEMEST provides multiple users for software measurement services via the Internet. It is understood that software engineering measurement is a systematic activity by using a number of coherent metrics rather than separated ones. The Software Engineering Measurement System (SEMS) has been developed that provides a comprehensive set of software engineering measurements and metrics for the SEMEST tool.

Refining software engineering goals to quantitative measurements has been enabled by the knowledge-based expert system in SEMEST. SEMEST has been designed to support goal-, process-, and category- oriented measurement in software engineering, and to do measurement analysis and benchmarking for software engineering projects. SEMEST has enabled the software industry to practice quantitative software engineering in a consistent and efficient way.

Future work of SEMEST will be related to adapt the expert system to a wide range of application domains and business contexts. Further investigation into the applications of SEMEST in the software industry and the updating of the SEMEST knowledge base will be focused.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the Natural Science and Engineering Council of Canada (NSERC) for its support to this work. We would like to thank the reviewers for their valuable comments and suggestions.

REFERENCES

[1] Basili, V.R., C. Caldiera, H.D. Rombach (1994), Goal Question Metric Paradigm, in J.J. Marciniak ed., Encyclopedia of Software Engineering, Vol. 1, John Wiley & Sons, pp. 528-532.

[2] Michel A. Carrico, John E. Girard, Jennifer P. Jones, Building Knowledge Systems, Developing and Managing Rule-Based Applications.

[3] Grady, R.B. (1992), *Practical Software Metrics for Project Management and Process Improvement*, Prentice-Hall, Inc., Englewood Cliffs, NJ.

[4] Lorenz, M. and J. Kidd (1993), *Object-Oriented Software Metrics: A Practical Guide*.

[5] Melton, A. *ed.*, (1996), *Software Measurement*, International Thomson Computer Press.

[6] Solingen, V.R. and E. Berghout (1999), *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*, McGraw-Hill Co., London.

[7] Wang, Y. and G. King (2000), *Software Engineering Processes: Principles and Applications*, CRC Press, USA, ISBN: 0-8493-2366-5, 752pp.

[8] Wang, Y. (2002), *Software Engineering Measurement and Analysis: An Applied Framework of Software Metrics*, CRC Press, USA, to appear.

[9] Wang, Y. (2002), *Component-Based Software Measurement*, Chapter 14, in F. Barbier ed. *Component-Based Software Engineering*, Kluwer Academic Publishers, UK, 2002, pp. 247-262.

[10] Wang, Y. (2003), *Measurement Theories for Software Engineering*, Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE'03), IEEE CS Press, Montreal, Canada, May.