QUALITY MEASUREMENT MODEL FOR REQUIREMENTS ENGINEERING FLOSS TOOLS

María Pérez, Edumilis Méndez, Kenyer Domínguez and Luis E. Mendoza
Processes and Systems Department, LISI, Simón Bolívar University, PO Box 89000, Caracas 1080-A, Venezuela

Keywords: Software Quality, Requirements Engineering, FLOSS, Quality Model.

Abstract: The goal of delivering a suitable or quality software product increases the properly definition of system requirements. Requirements Engineering (RE) is the process of discovering, refining, modeling and specifying software requirements. In addition to the trend of using Free/Libre Open Source Software (FLOSS) tools, we should consider their strengths and weaknesses towards in the light of a suitable RE. This article is aimed at proposing a quality measurement model for RE FLOSS tools and supporting their selection process. Characteristics selected for its evaluation include Functionality, Maintainability and Usability. This model was applied to four FLOSS tool and assessed for completeness, accuracy and relevance to establish which FLOSS tools support RE, either totally or partially, thus making it useful for Small and Medium-sized Enterprises.

1 INTRODUCTION

Selecting FLOSS tools that support for Requirements Engineering (RE) is challenging, as tools must support all RE stages and validate that the features characterizing this type of software are fully met. Such premise has led us to evaluate the tools quality by means of a quality model that allows determining the fulfillment of requirements.

For this research, we used and instantiated the systemic quality model (MOSCA) (Mendoza et. al, 2005). This model is based on ISO 9126 (ISO/IEC 9126, 2001), the Dromey Quality Model (Dromey, 1995), and the Goal-Question-Metrics (GQM) paradigm (Basili et. al, 2001). The instantiation proposed herein includes software attributes, such as Functionality, Usability and Maintainability, and establishes 129 new metrics for a total of 210 metrics, to evaluate FLOSS tools supporting RE. Four tools were selected: Open Source Requirement Management Tool (OSRMT), StarUML, Use Case Maker (UCM) and OpenOME; the four being open tools with Free Software licenses.

This article consists of 7 sections. First and second sections present the introduction and the methodology applied in this research, respectively. Third section describes MOSCA. Fourth section introduces the quality model proposed for FLOSS tools supporting RE. Fifth section describes the model application and results obtained. Lastly, sixth section presents our conclusions and recommendations.

2 METHODOLOGY

For this work, we used the Systemic Methodological Framework for Information Systems research (Pérez et. al, 2004), based on DESMET (Kitchenham, 1996) methodology and the research-action method (Baskerville, 1999). The action-research method is developed in five phases: diagnosing, action planning, taking action, evaluating, and specifying learning (Baskerville, 1999), whereas DESMET methodology is used to supplement the model evaluation. This methodology suggests 9 evaluation methods, among which the case study feature analysis (Kitchenham, 1996) was applied herein. In addition, the GQM approach was included, in order to evaluate software in a quality improvement context (Basili et. al, 2001).

3 SYSTEMIC QUALITY MODEL

The purpose of the systemic quality model is to measure systemic quality at a software developing organization (Rincón et. al, 2005) and assess Information Systems quality by integrating the

Pérez M., Méndez E., Dominguez K. and Mendoza L. (2010).
QUALITY MEASUREMENT MODEL FOR REQUIREMENTS ENGINEERING FLOSS TOOLS .
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 249-254
DOI: 10.5220/0002905402490254

Table 1: Functionality characteristics.

Characteristic	Definition and Rationale
Suitability (FUN 1)	The capability of a software product to provide an adequate group of functions according to user specific tasks and objectives. In this research, tool functions should be adapted to satisfy RE needs. The tool must cover all functional needs.
Interoperability (FUN 3)	The capability of a software product to interact with one or more specified systems. The RE tool can use or provide functionalities from/to other systems. Also, it must relate to subsequent stages in the Software Engineering process.
Correctness (FUN 5)	It is divided into three categories related to computing, completeness and consistency capabilities. The violation of any of such properties may generate software without the functionality level expected. Tools supporting RE must generate complete and consistent requirements.

Table 2: Sub-characteristics proposed for Functionality of FLOSS tools supporting RE.

Sub-characteristic	Description
Diagrams (FUN 1.1)	The capability of the tool to represent RE related models through diagrams associated with a modeling language, such as UML.
Documentation (FUN 1.2)	The capability of the tool to provide mechanisms necessary to generate related documentation.
Classification (FUN 1.3)	The capability of the tool to classify requirements associated with a software project.
Phase support (FUN 1.4)	The capability of the tool to provide support to the different RE phases.
Consistent (FUN 5.1)	The capability of the tool to verify that dependency relations among requirements, diagrams generated from others, change traceability, among other characteristics, be consistent.

quality models of the product and development process (Alfonso et al, 2008). MOSCA consists of four levels, namely:

- **Level 0. Dimensions.** Includes the internal and contextual aspects of the process, and the internal ad contextual aspects of the product.
- **Level 1: Categories.** Consists of eleven (11) categories, six (6) belonging to the product and five (5) to the development process.
- **Level 2: Characteristics.** Each category has a set of characteristics that define key areas to achieve, assure and control product and process quality.
- **Level 3: Metrics.** For each characteristic, there is a series of metrics to measure systemic quality; total metrics are 715.

MOSCA evaluates the software product in accordance with international standards, given that the aforementioned categories agree with the characteristics of ISO 9126 (ISO/IEC 9126, 2001), established to assure the quality of the software product to be evaluated.

Mendoza et. al (2005) introduced an algorithm to evaluate software quality using MOSCA, which will be instantiated in the context of SQM for the purpose of this research.

4 QUALITY MODEL PROPOSAL

The Systemic quality model (MOSCA) established the selection of three out of the 6 Product Perspective categories, one being Functionality. The other two categories selected are Usability, because the tool enables requirements management in a simple and easy manner and provides an interface that is appealing to users; and Maintainability, because this a FLOSS tool and should allow obtaining documentation to get access to information required for making changes and maintenance. Based on prior works of Pessagno et. al (2008) and Alfonso et. al (2008), a sub-group of characteristics was selected for each category; also, a sub-set of metrics was selected for this sub-group, and in certain cases, it was necessary to add new metrics. Find below an explanation to each category and criteria used for selecting these characteristics.

4.1 Functionality

Functionality is the capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions (ISO/IEC 9126, 2001). It is essential that a RE tool meet the functional requirements expected from a software product. The characteristics selected for this category are described in Table 1.

To fulfill the requisites necessary to represent RE phases, new sub-characteristics were added, and are presented in Table 2.

Table 3: Usability characteristics.

Characteristic	Definition and Rationale
Understandability (USA 1)	The capability of a software product to enable the user software understanding and use. The RE supporting tool must provide the required use easiness to the team in charge of eliciting and managing requirements.
Learnability (USA 2)	The capability of a software product to enable the user learning and performance of basic operations. The RE tool must provide the user with use forms and enable it to learn how to operate them.
Graphic interface (USA 3)	The capability of a software product to be appealing to users. The RE tool interface must be comfortable enough to manage requirements in a user-friendly environment.
Operability (USA 4)	The capability of a software product to enable the user operation and control thereof. The user should count on all facilities to operate and control the RE tool.
Self-descriptive (USA 11)	A structural form is deemed self descriptive when its purpose is evidenced in the name of the modules and when labels have meanings that refer to the application context. If the RE tool is self-descriptive, the tool-user interaction will be easily achieved.

Table 4: Sub-characteristics proposed for Usability of FLOSS tools supporting RE.

Sub-characteristics	Description
Ergonomics (USA 1.1)	The capability of a tool interface to enable tool-user interaction.
Error control (USA 4.1)	The capability of a tool to enable user recovery from system errors.
Documentation (USA 4.2)	The tool's documented functionalities.

Table 5: Maintainability characteristics.

Characteristic	Definition and Rationale
Analyzability (MAB 1)	The capability of a software product to be diagnosed for software error or failure. The RE tool should be easy to diagnosed, as this will determine the selection of parts susceptible of upgrades.
Changeability (MAB 2)	The capability of a software product to enable the implementation or improvement of new/existing functionalities. Changeability is an essential characteristic of the RE tool subject to modifications; it allows adding modifications suggested by developers.
Stability (MAB 3)	The capability of a software product to avoid unexpected effects upon functionality modifications. The RE tool subject to modifications must keep its stability upon change implementation.
Coupling (MAB 5)	The lowest possible coupling level is pursued as it enables software changes. The RE tool is required to count on simple module interconnection to make code modification much easier.
Cohesive (MAB 6)	It is desirable that all elements be closely linked to each other and contributes to meet the objective. The RE tool should have all its elements duly linked to achieve simple and desirable operation.
Software maturity attributes (MAB 8)	The group of features associated to age and use of the tool. The RE tool must ensure that all modifications made in the future will not affect its quality.

4.2 Usability

Usability is the capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions (ISO/IEC 9126, 2001), The RE tool should enable requirements management in a simple and easy manner and provide an interface that is appealing to users. Table 3 shows the characteristics selected for this category.

Table 4 shows the sub-characteristics added to the original model and their corresponding description.

4.3 Maintainability

Maintainability is the capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications (ISO/IEC 9126, 2001), Because the tool subject to modification is a FLOSS tool, it should enable the obtaining of documentation to get access to the information required to make changes and maintenance. The characteristics selected for this

Table 6: Sub-characteristics proposed for Maintainability of FLOSS tools supporting RE.

Sub-characteristic	Description
Code readability (MAB 1.1)	The capability of the source code to be read by any developer, even if not belonging to the project team.
Modification (MAB 1.2)	The capability of the source code to be modified.
License (MAB 2.1)	It refers to the tool license properties. A FLOSS tool license should count on 4 user freedoms, freedom to have access to the source code, freedom to modify the code, freedom to copy the code, and freedom to distribute the code.
Services (MAB 3.1)	The support, consulting, and training services provided by the tool development community.
Adoption (MAB 8.1)	The level of tool acceptance at the market among individuals and companies.

Table 7: The results obtained from the evaluation.

Category	Feature	Sub-feature	StarUML	OSRMT	UCM	OpenOME
Functionality	Suitability	Suitability	60%	100%	60%	60%
		Diagrams	40%	0%	0%	80%
		Documentation	50%	0%	18.75%	11.11%
		Classification	40.90%	69.69%	62.50%	47.82%
		Phase support	43.75%	30.76%	28.57%	26.66%
	Interoperability	Interoperability	16.66%%	0%	37.50%	20%
	Correctness	Consistent	37.50%	71.42%	57.14%	14.28%
Functionality percentage			**0%**	**14.29%**	**0%**	**14.29%**
Usability	Understandability	Understandability	100%	83.33%	100%	100%
		Ergonomics	100%	100%	100%	100%
	Learnability	Learnability	50%	0%	100%	0%
	Graphic interface	Graphic interface	100%	100%	100%	100%
	Operability	Operability	80%	90%	80%	100%
		Error control	100%	100%	0%	100%
		Documentation	100%	100%	20%	100%
	Self-descriptive	Self-descriptive	100%	100%	100%	100%
Usability percentage			**87.50%**	**87.50%**	**75.00%**	**87.50%**
Maintainability	Analyzability	Analyzability	100%	100%	100%	100%
		Code readability	100%	100%	100%	100%
		Modification	100%	100%	100%	100%
	Changeability	Changeability	100%	100%	100%	100%
		License	100%	100%	83.33%	100%
	Stability	Stability	100%	100%	100%	100%
		Services	50%	50%	50%	100%
	Coupling	Coupling	100%	100%	100%	100%
	Cohesive	Cohesive	100%	100%	100%	100%
	Software maturity attributes	Software maturity attributes	66.66%	50%	83.33%	66.66%
		Adoption	100%	80%	60%	80%
Maintainability percentage			**81.82%**	**81.82%**	**81.82%**	**90.91%**
Quality level			**Null**	**Null**	**Null**	**Null**

category are described in Table 5.Table 6 shows all sub-characteristics added to this category. MOSCA instantiation for FLOSS tools supporting Requirements Engineering presents 210 metrics, of which 129 are new metrics (127 for Functionality, 1 for Usability, and 1 for Maintainability).

5 INSTATIATION APPLICATION AND RESULTS ANALYSIS

MOSCA instantiation was applied to four tools: Open Source Requirement Management Tool (OSRMT), StarUML, Use Case Maker (UCM) and OpenOME, the four being open tools with Free Software licenses. From these tools, StarUML is considered and Analysis and Design tool, but may also show other characteristics such as RE supply. Also, StarUML allows for automatic generation of the use case specification document.

OpenOME incorporates an improved version of OME (a modeling and analysis tool oriented towards goals and agents that provides connection between requirements/specifications development and

architectonic design, and i*modeling support, which is a framework suggesting an approach oriented towards RE goals and agents), with other RE supporting tools oriented to goals, agents, and aspects. The results obtained from the evaluation are detailed in the Table 7.

In sum, approximately 64 hours were incurred for instantiation application. The process was performed as follows: the tools' most recent versions were downloaded from the corresponding development community website and installed. Then, applications were run one by one and verified for metrics compliance. For evaluation of Maintainability metrics, we reviewed current information and documentation at the official tool website. Information required for software qualification could not be found for some metrics; therefore in these cases, a minimum punctuation was awarded (1).

For the Stability feature in the Maintainability category, where no security patches were founds, the formula was deemed a ratio (patches solved/patches found) expressed in percentage values; 100% was awarded.

The results obtained for the Functionality category: StarUML obtained 0%, OSRMT 14.29%, UCM 0%, and OpenOME 14.29%. According to the MOSCA algorithm, none met the level of acceptance required, which is 75%. Regarding results obtained for Usability, all four tools exceeded the 75% required. OpenOME, StarUML and OSRMT obtained the highest punctuation at 87.50%, followed by UCM with 75%. Lastly, for Maintainability, all 4 tools exceeded 75%, OpenOME with 90.91%, followed by StarUML, OSRMT and UCM with 81.82%.

Given that none of the 4 tools met the minimum satisfaction percentage required for Functionality, they qualified as null quality tools. Nevertheless, these four tools are above 75% for Usability and Maintainability. According to the MOSCA algorithm, when Functionality results do not reach 75%, the evaluation is suspended, but in this case, as we are evaluating FLOSS tools, there is the possibility of adding new functionalities to the tools, as opposed to proprietary software tools, which do not allow for modifications. Therefore, we are free to choose a tool and subject it to any improvement relating to the characteristics deemed appropriate for our research. The selected tool was UCM, a tool developed in C# language. It should be noted that results presentation goes beyond the scope of this article and will be addressed in future works.

6 CONCLUSIONS

This work proposes an instantiation of the MOSCA model to measure the quality of FLOSS-based software engineering tools supporting RE, which should be easy to use and modify. This model was applied to Open Source Requirement Management Tool (OSRMT), StarUML, Use Case Maker (UCM) and OpenOME, to prove the model usability and select the most suitable tool to be modified in the near future. In this case, the tool selected was Use Case Maker. MOSCA may be adapted to any RE tools with specific characteristics and may be used by Small and Medium-sized Enterprises (SMEs) for tool evaluation purposes.

ACKNOWLEDGEMENTS

This research has been financed by FONACIT Venezuela, Project G-2005000165. Special thanks to A. Sevilla.

REFERENCES

Alfonso, O., Domínguez, K, Rivas, L., Perez, M., Mendoza, L., & Ortega, M. (2008). Quality Measurement Model for Analysis and Design Tools based on FLOSS. *19th Australian Software Engineering Conference (ASWEC 2008). Libro: "Proceedings of the 19th Australian Software Engineering Conference (ASWEC 2008)". Vol. 1. pp. 258 - 267*

Perth, Australia. Basili, V., Caldiera, G. y Rombach, H., "The Goal Question Metric Approach", en: Marciniak, J. J. (ed.), Encyclopedia of Software Engineering, Wiley, pp. 528–532, 2001.

Baskerville, R., "Investigating Information Systems with Action Research", Communications of the Association for Information Systems, vol. 2, nº 19, pp. 1-32, 1999.

Dromey, G., "A Model for Software Product Quality", IEEE Transactions on Software Engineering, vol. 21, nº 2, pp. 146-162, 1995.

ISO/IEC 9126-1, Software Engineering. Product Quality. Part 1: Quality Model, ISO, 2001.

Kitchenham, B., "Evaluating Software Engineering Methods and Tools. Part 1: The Evaluation Context and Evaluation Methods", ACM Software Engineering Notes, vol. 21, nº 1, pp. 11- 14, 1996.

Mendoza, L., Pérez, M. y Grimán, A., "Prototipo de modelo sistémico de calidad (MOSCA) del software", Computación y Sistemas, vol. 8, nº 3, pp. 196-217, 2005.

Pessagno, L., Domínguez, K., Rivas, L., Pérez, M., Mendoza, L., & Mendez, E. (2008). Modelo de calidad para herramientas FLOSS que dan apoyo al modelado de procesos del negocio. X Jornadas sobre

Innovación y Calidad del Software (JICS), September. Madrid.

Pérez, M., Grimán, A., Mendoza, L. y Rojas, T., "A Systemic Methodological Framework for IS Research", Proceedings of the Tenth Americas Conference on Information Systems AMCIS. New York (USA), pp. 4.374-4.383, 2004.

Rincón, G., Mendoza, L. & Pérez, M. 2004. Guía para la Adaptación de un Modelo Genérico de Calidad de Software. IV Jornadas Iberoamericanas en Ingeniería de Software e Ingeniería del Conocimiento - JIISIC, Madrid, España.