

# QUALITY MODEL FOR THE SELECTION OF FLOSS-BASED ISSUE TRACKING SYSTEM

Eduardo Raffoul, Kenyer Domínguez, María Pérez, Luis E. Mendoza, Anna C. Grimán  
Laboratorio de Investigación en Sistemas de Información (LISI),  
Departamento de Procesos y Sistemas, Universidad Simón Bolívar,  
Apartado 89000, Baruta, Caracas 1080-A, Venezuela.  
eraffoul@gmail.com, {kdoming, movalles, lmendoza, agriman}@usb.ve

## ABSTRACT

The complexity of the issue tracking systems (ITS) which meet the requirements of the Infrastructure Technology Information Library (ITIL) encumbers their selection. In addition, we have to consider some other variables, such as the wide range of tools, their functionality level and their costs. Regarding the cost of the ITS, nowadays the use of ITS based on Free/Libre Open Source Software (FLOSS) is an increasing trend. Therefore, the purpose of this article is to present a model to evaluate the quality characteristics of the FLOSS-ITS according to ITIL recommendations. This quality model is aimed at selecting the best of the available tools. The set of characteristics evaluated is presented based on the product perspective of the Software Quality Systemic Model (MOSCA). This perspective is inspired on the ISO/IEC 9126 standard and Dromey's quality model. Also it was applied to a case study -a venezuelan company-, which is willing to exploit and extend the capabilities of this type of tools to its customers. The case study allowed us to establish the FLOSS-ITS's quality requirements: functionality, reliability and usability. Lastly, validation of the model through its application to three FLOSS-ITS tools is presented.

## KEYWORDS

Software Quality Model, ISO 9126, Issue Tracking System, Freeware, ITIL.

## 1. Introduction

Free Software products are those ensuring four kinds of freedoms for users [9]: (1) Freedom to run the program at any place, for any purpose; (2) Freedom to study how the program works and adapt it to their needs, which requires access to the source code; (3) Freedom to redistribute copies, allowing for cooperation; and (4) Freedom to improve the program and release such improvements to the public in order to benefit the community. The most common license to distribute this type of products is the GNU/GPL [28] (General Public License), created by the Free Software Foundation. Another related movement focuses on providing licenses, which among others benefits, grant access to the source code: Open Source Initiative [16]. As a result

thereof, the term FLOSS was coined to join both movements' efforts. These movements have had an increased participation in corporate environments. Help Desk (HD) systems, particularly Issue Tracking Systems (ITS) do not escape from this reality.

The HD concept is closely related to the processes of each organization, since it is a vital tool for the centralization and follow-up of tasks [1]. Basically, the main purpose of a HD is to serve as a common ground to manage the requirements and events taking place in an organization [26], either externally with its customers or internally, with its employees. Likewise, ITIL defines different related processes that go hand in hand with the HD concept, such as Incident Management, Problem Management, Change Management and Configuration Management, included within the Service Support area. Additionally, Service Level Management [6, 26] is included in Service Delivery best practices. Each process requires the assistance of information systems efficiently managing information used.

ITS are the tools used in HD. Their main function is to receive "cases" in a central database, which are recorded for subsequent resolution and/or attention. This process can be followed up over the issue's life cycle. However, depending on the ITIL recommendations and the specific requirements of each particular business, they may get to be very complex, including features such as interactive voice response systems, exchange switchboard interaction and companies' assets database management [10]. Additionally, there is a great diversity of ITS in the market, which compete with each other by offering similar functionalities.

Given the diversity of technology proposals in this area and the complexity of their evaluation and/or selection, organizations require a tool that allows systematically determining the satisfaction of functional and nonfunctional requirements of these systems, thus facilitating the selection of the best group of alternatives.

This article consists of seven sections. Section 2 includes the background; section 3 documents the methodology followed; section 4 presents the quality model proposal; section 5 describes the case study; the results obtained are analyzed in the section 6, and,

lastly, section 7 presents conclusions and recommendations

## 2. Quality Systemic Model - MOSCA

According to Pressman [23], software quality is the concordance of functional and performance requirements explicitly established with explicitly documented development standards and implicit features expected for any professionally-developed software. The lack of faults, suitability, security, reliability and the formulation of specifications are elements involved in the software quality concept. For purposes of defining software quality, software product quality must be differentiated from its manufacturing process quality [20]. However, goals established for the final product determine the objectives of the development process, since the quality of the former element depends, among other, on the quality of the latter.

Taking into account product quality [17,18], where product quality is assessed regarding its functionality and process quality [20], which evaluates its development process, the Information Systems Research Department (LISI) at Simón Bolívar University (Venezuela) developed the Software Quality Systemic Model (MOSCA). This model relies on total systemic quality concepts [22]. The Human Group Quality [22], which assesses the quality of the community or group of people involved in the software development process, was recently added. However, this work only considered the first sub-model, since the main purpose is to systematically evaluate existing FLOSS-ITS. It should be mentioned that MOSCA might be adapted to specify the ITS quality in a context other than ITIL or FLOSS, by selecting different categories and features in line with the case's quality requirements. This is mainly due to the model's systemic approach that allows adapting assessment to the environment where it is applied, considering quality required by users. For the Product sub-model, MOSCA took the best of Dromey [5] and ISO/IEC 9126 [11], encompassing the following levels:

- Level 0. Dimensions. It includes the internal and contextual aspects of Product measuring quality within the context of the ratio between the result obtained and the desired result. The result of combining this definition with Product is that contextual and internal aspects of the Product measure quality within the context in which software operates. In this case, it is especially important to determine software quality features observed in running time. Additionally, only manuals provided by the licensees and software demo versions to be evaluated are available.

- Level 1. Categories. It is comprised by three categories. This level includes functionality category, while the other two categories should be selected among the other five proposed by MOSCA [18].

- Level 2. Characteristics. This in-depth level

correspond to a group of characteristics defining key areas to be satisfied in order to achieve, secure and control quality of the Product and all categories.

- Level 3. Sub-characteristics. Depending on the features specific for each assessment objective, sub-characteristics that allow for a better assessment of each feature may be included. The formulation of sub-features will always be related to the characteristic from which it stems.

- Level 4. Metrics. For each characteristic and sub-characteristic, there is a series of metrics related to software qualities and attributes to be evaluated.

In addition, [18] introduced an algorithm to evaluate software quality using MOSCA, which is summarized as follows:

- 1) Estimating quality of product functionality. Product functionality must be measured first for all categories. If 75% of the characteristics required for this category are satisfied, then the second activity is addressed.

- 2) Instantiation of the product sub-model. In this activity, the customer must select two categories from the five remaining categories of the product sub-model, including those considered as software musts and those it wishes to evaluate. Then, each of the categories selected by the customer must be evaluated. At this point, it should be noted that the algorithm recommends working with a maximum of three product features (including functionality). If more than three product features are selected, these may conflict with each other.

- 3) Quality measurement for each category. For the two categories previously selected, the following should be made:

- Apply metrics proposed in the product sub-model for selected categories.

- Verify that 75% of the metrics reach optimum values (3 or higher) for each feature.

- Evaluate the category. For fulfilling a category, at least 75% of its features must be highly satisfied, thus guaranteeing coherence and consistency regarding the model acceptability levels.

- 4) Measure product quality from categories evaluated. At this point, it should be remembered that when functionality is not satisfied, the algorithm ceases to work, and software product quality is deemed null. If a software product meets its original purpose (functionality), it is deemed as having a basic quality. If it satisfies only one of the categories selected, besides functionality, it is deemed as having reached an intermediate quality level; but if it satisfies all selected categories, it is deemed as having reached an advanced quality level.

## 3. Methodological Framework

This research used the Methodological Framework for Research of Information Systems (Pérez et al., 2004). The adaptation of the Methodological Framework for this work consists of ten steps: 1) Documentary and bibliographical research; 2) Background Analysis; 3)

Formulation of Objectives and Scope of Research; 4) Adaptation of Methodological Framework; 5) Proposal of a strategy to support quality reuse; 6) Analysis of Context; 7) Application of DESMET Methodology (Kitchenham, 1996), the most appropriate method to evaluate strategy was the analysis-survey feature; 8) Evaluation of proposed strategy; 9) Analysis of results; 10) Conclusions and Recommendations.

#### 4. ITS Quality Model Proposal

The first step to formulate the model was to define the functionality characteristics of the ITS as provided by the methodology proposed in [21], supported by bibliographical references. For this purpose, a comprehensive review of publications related to ITS and their application to HD processes environment was performed, thus finally selecting functionality, reliability and usability. Rationale for this selection is described as follows:

- The functionality category was selected because software to be evaluated must provide functions that meet specific or implicit needs corresponding to ITS. According to MOSCA, this aspect is mandatory [20].
- Reliability corresponds to software's need to provide HD personnel and customers with reliable results and information for their work processes, since HD users and customers need to communicate with Help Desk assistants on a daily basis, thus allowing to monitor users' environment in connection with technical issues that might arise and satisfaction with proposed solutions, and facilitating use of this information in other business areas, even in those not related to IT [16].
- The usability feature was selected since the ITS will generally be operated by a large number of Help Desk agents, who do not always have the same training level, and in certain cases by HD customers [10]. This research was focused on product quality estimation, since the most important aspect here is to support the selection of existing FLOSS-ITS. Then, the proposed model encompasses the functionality, reliability and usability categories described in Tables 1, 2, and 3. The FUN1 characteristic is that where particular sub-characteristics have been proposed for FLOSS-ITS by maintaining sub-characteristics of the other characteristics as proposed in the original model. This is because the Suitability characteristic specifies the functional requirements expected by the software for which the model is being developed.

**Table 1. Functionality characteristics and sub-characteristics selected for FLOSS-ITS evaluation**

Characteristic	Sub-characteristic
FUN1. Suitability.	Scalability, Knowledge Management, Incidents Management, Issues Control, Errors Control, Configurations Management, SLA, Reports.

FUN2. Accuracy	The ability of a system to provide accurate results at the appropriate scalability level.
FUN3. Interoperability	Dependencies, Databases Interactions, Other Devices Interactions.
FUN4. Security	It refers to the system's ability to restrict unauthorized accesses to the system.
FUN5. Correctness	It refers to the system's ability to offer correct data and results to users, in line with database and users' requirements.
FUN6. Structurated	A structural property has to be structured if it follows the rules of the structured programming.
FUN7. Encapsulated	Variables, constants and types must be used within the context they were originally defined in (Dromey, 1995, 1996)
FUN8. Specified	Functionality is described with pre and post-conditions (Dromey, 1995, 1996)

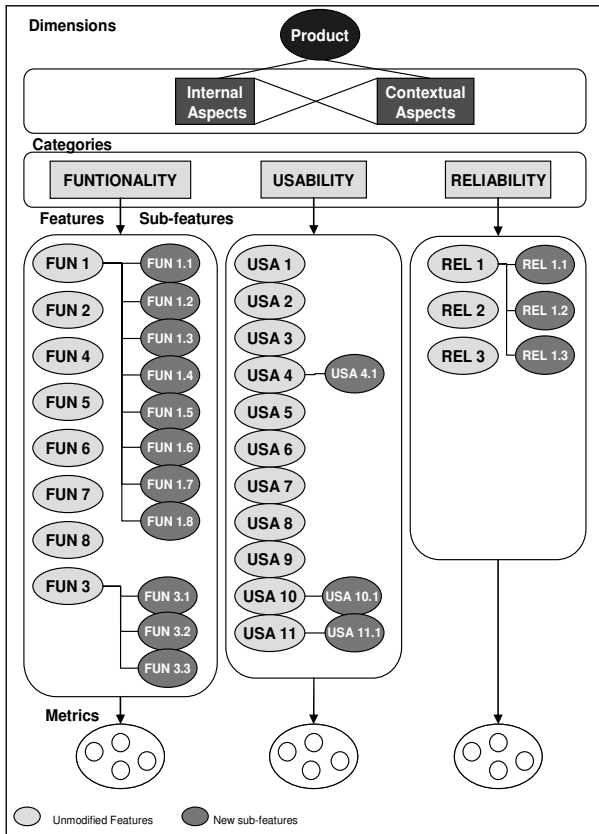
**Table 2. Reliability characteristics and sub-characteristics selected for FLOSS-ITS evaluation.**

Characteristic	Definition
REL1. Maturity	These are the measurements associated to age and use of the target system.
REL2. Fault Tolerance	Ability of the product to manage and avoid faults resulting from product daily operations.
REL3. Recovery	Ability of the product to recover from crashes and to subsequently restart operations.

**Table 3. Usability characteristics and sub-characteristics selected for FLOSS-ITS evaluation.**

Characteristic	Definition
USA1. Understandability	The ability of a software product to be easily understood by users in terms of general operations.
USA2. Learnability	Quality and effectiveness of the supporting material aimed at describing product use.
USA3. Graphic interface	Quality characteristics of the graphic interface that allow for user-product interaction.
USA4. Operability	It defines properties of the operation scheme by product users.
USA6. Complete	A product is deemed complete when its structural forms comprise all elements; e.g. Modules, Instructions, Objects, etc.
USA7. Consistent	A product is deemed consistent when its structural forms maintain their properties.
USA8. Effective	A product is deemed effective when its structural forms count on all necessary elements to be defined and implemented.
USA9. Specified	Functionality is described with pre and post conditions (Dromey, 1995, 1996)
USA10. Documented	It refers to the existence of manuals for understanding the product, its functionalities and implementation. It includes code comments for proper understanding.
USA11. Self-descriptive	It refers to easy understanding of the product's code.

Instantiation of FLOSS-ITS MOSCA is shown in Figure 1. The proposed model includes 124 metrics, 60 out of



**Figure 1. MOSCA Instantiation for FLOSS-ITS evaluation.**

which were applied designed for this research, 50 corresponding to functionality, 4 to reliability, and 6 to usability, which are complemented with MOSCA original characteristics, of which 23 correspond to functionality, 8 to reliability and 33 to usability. It can be seen in Figure 1 that the majority of new metrics are in FUN 1 because that is where the instantiation includes typical ITS characteristics.

The model proposed to evaluate the FLOSS-ITS is comprised of 73 metrics for functionality, 12 for reliability, and 39 for usability. Table 4 shows a sample of metrics proposed in this model.

**Table 4. Extract of metrics proposed.**

Sub-charac.	Metrics
FUN1.1	Does it allow recording roles or user groups?
FUN1.2	Is it Computer-Telephone integrated?
FUN1.3	Does it allow assigning a status to incidents?
FUN1.4	Does it allow classifying issues?
FUN1.5	Are solutions given to errors recorded?
FUN1.6	Does the system have a knowledge database?
	Is it possible to link known errors to issues?
FUN1.7	Is the system capable to interact with configurable SLA?
FUN1.8	How complete is the reporting system?

FUN3.1	Does it allow for any knowledge indexation technique?
REL1.1	How many versions of the product have been released so far?
	How many months have elapsed since the first version was released?
REL1.2	How many system successful implantations have been made?
	How many downloads have been made from the product's official website?
USA10.1	Is the functionality of instructions commented in the code?
	Is there access to the developer's manual?
	Is there any type of documentation based on visual modeling? e.g. UML
	Is there access to any type of API?
USA11.1	How complex it is to interpret the code for reviewing and/or auditing purposes?

## 5. Case Study

In order to validate the proposed model, the FLOSS-ITS was evaluated in a Venezuelan company (named Sync Consulting Co.) an organization involved in providing solutions for computing platforms interconnection. This evaluation was performed on three FLOSS-ITS (see Table 5) and was conducted by:

- Sponsors: This evaluation was sponsored by Sync Consulting Co.

- Evaluator: It is the person responsible for performing the review; Eduardo Raffoul in this case, jointly with a Systems Engineer acting as advisor on behalf of the sponsor. Their responsibilities included the preparation of an evaluation plan, identification of the object to be evaluated, identification and definition of features to be evaluated in the MOSCA adaptation to the ITS, organization of the object to be evaluated, gathering and analysis of results obtained, preparation of final report and recommendations.

- Researchers: Kenyer Domínguez, María Pérez, Luis Mendoza and Anna Grimán, representing the Information Systems Research Department (LISI), acting as advisors by virtue of their roles as responsible for MOSCA's development and enhancement.

**Table 5. Description of FLOSS-ITS to be evaluated.**

ITS	DESCRIPTION
JTrac [13]	JTrac is a Java-based program. It is an open code ITS with a web-technology configurable license. Its first stable version was released on May 6, 2006. The project's current stable version is 2.0.
RT [24]	Request Tracker (RT) is a project of Best Practical Solutions LLC. Distributed under GPL license, RT is a ITS in PERL language, based on web technology. Developers have released two RT-based products, namely RT Faq Manager (RTFM) and RT for Incident Response (RTIR), which provide RT with ITIL features beyond simple ticket management. The first RT version was

	released on October 13, 1999; the stable version currently in use is 3.6.4. The first stable version of RTFM was released on August 26, 2003, 1.1.5 being the current stable version. The first version of RTIR was released on September 25, 2003, and current stable version is 2.0.4.
OTRS::ITSM [19]	Open source Ticket Request System (OTRS) is a software developed and maintained by OTRS Inc. OTRS was developed in Perl, and it is distributed under GPL license. It operates under a Web platform. Developers have released a plug-in named OTRS::ITSM, which provides the system with functionalities meeting ITIL, such as incident management, knowledge management and configuration management. The first stable version of OTRS was released on September 1, 2003, 2.2.2 being the current stable version. The first version of OTRS::ITSM was released on February 7, 2007, and current stable version is 1.0.2.

## 5. Results Analysis

Once measurements were made based on functionality, usability and reliability features in the three FLOSS-ITS, the following results were obtained:

### - Functionality:

Figure 2 shows that the OTRS::ITSM product met 6 of the 8 features for functionality, thus satisfying minimum 75% of associated metrics. Unmet features are FUN3 (Interoperability) and FUN8 (Specified). The former is due to the product's lack of interaction with communication elements, such as phone integration systems; the latter is due to the product's lack of elements to describe code structures pre-conditions or post-conditions. The other two products, RT and JTrac, only met 50% of functionality features (4 out of 8). Both products did not meet FUN1 (Suitability), FUN3 (Interoperability), FUN4 (Security) and FUN8 (Specified). However, RT including the features of Incident Management and FAQ Manager met more metrics for FUN1 than JTrac. It should be mentioned that OTRS::ITSM does not show any disadvantage with respect to the other products.

### -Reliability:

According to Figure 3, the three products met 100% of reliability features, since tests applied on products, as well as statistic data obtained, showed that they met at

least 75% of the features for this category, i.e. these three elements reflect a 100% compliance with REL2 (Fault Tolerance). OTRS and RT met 100% of maturity metrics, way ahead of JTrac with 75%, mainly due to the fact it is a recent product within the market. These three elements share equal conditions regarding REL3 (Recovery).

### - Usability:

As can be seen in Figure 4, OTRS::ITSM and RT products are on the top of the usability category, with 6 out of 11 features covering at least 75% of metrics. OTRS::ITSM met USA2 (Learnability), USA3 (Graphic Interface), USA6 (Completeness), USA7 (Consistent) and USA8 (Effective), all of them with 100%, and USA4 (Operability) with 75%. Additionally, RT met USA2 (Learnability), USA6 (Complete), USA7 (Consistent) and USA8 (Effective), all with 100%, USA3 (Graphic Interface) with 85% and USA4 (Operability) with 85%. As observed herein, OTRS::ITSM exceeds RT regarding USA3 (Graphic Interface), but RT is over this position regarding metrics related to USA4 (Operability). JTrac is last in this category, just meeting four features, USA6 (Complete), USA7 (Consistent), USA8 (Effective) and USA11 (Self-descriptive), the latter being the only feature where it beats the other two products. After analyzing results of the evaluations, it was concluded that OTRS::ITSM is the product with best capabilities, both functional and nonfunctional, especially when compared to RT and JTrac, by globally beating both of them in all features, except for Usability, which is higher in JTrac.

## 6. Results Discussion

According to the bibliographical review on ITS and ITIL application for this type of systems, evaluators considered as relevant those metrics proposed for the functionality category, since they include the necessary features and functionalities to be satisfied and implemented in a good ITS implementing ITIL recommendations. Furthermore, we confirmed the

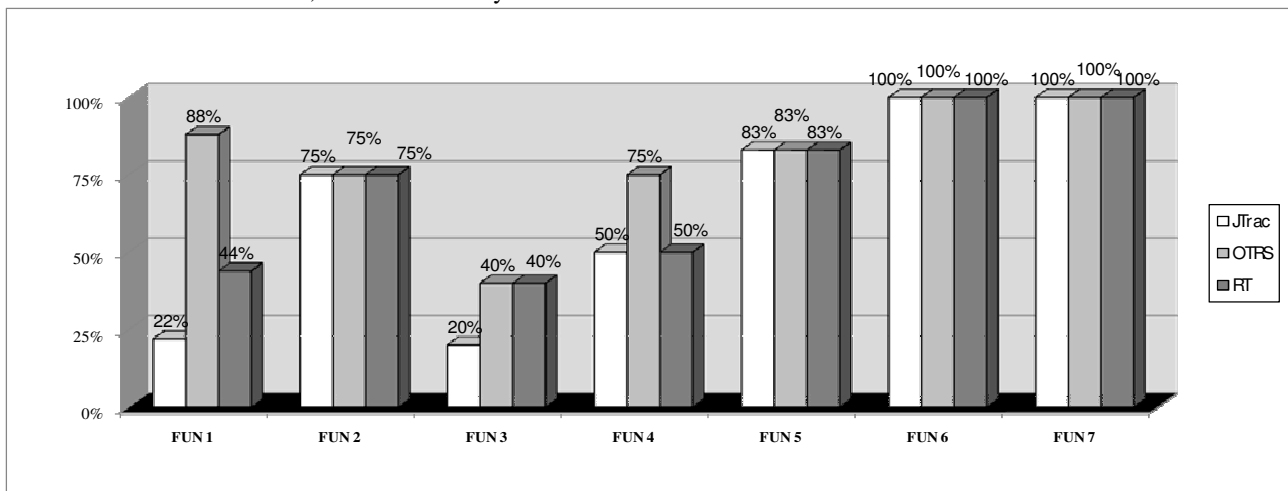
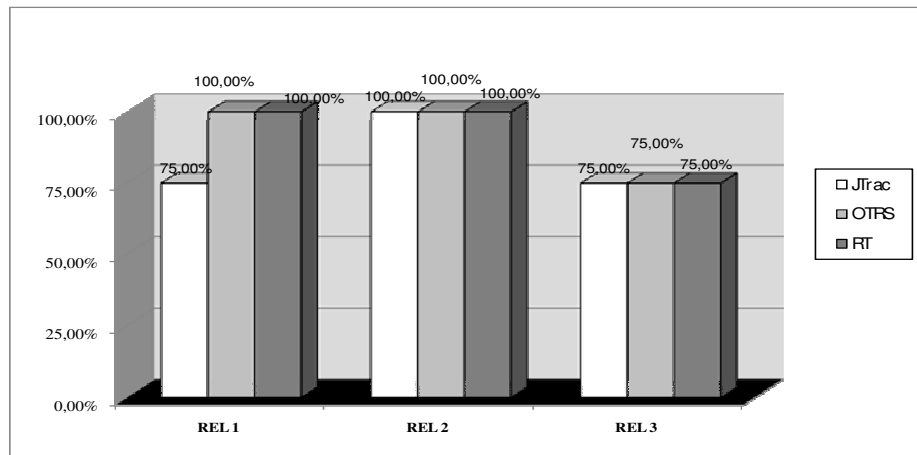
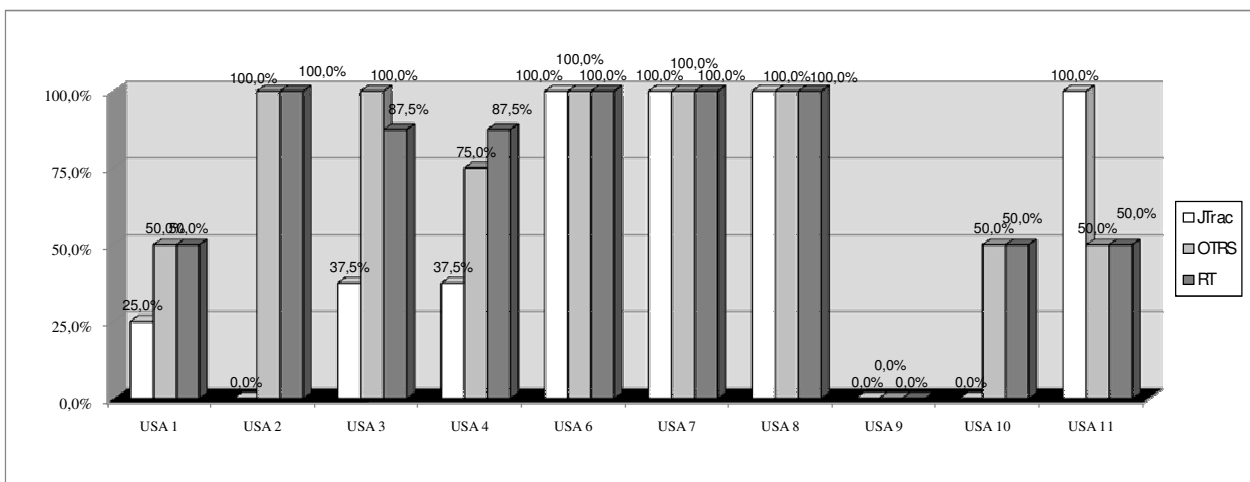


Figure 2. Category Evaluation: Functionality.



**Figure 3. Category evaluation: Reliability**



**Figure 4. Category evaluation: Usability.**

importance of selecting Usability and developing metrics to evaluate this category, since it is a key aspect for Help Desk users and agents to easily and pragmatically use the ITS to be selected; hence, the suitability of this model for this type of products. In addition, adequacy of the model was verified through its application, since all metrics proposed entail a close relation with the object to be evaluated. For most cases, we were able to obtain information required to evaluate metrics and we managed to adapt the model obtained to any other particular need of the ITS application scope, including sub-features and additional metrics that complemented existent metrics. Lastly, the model's utility was determined by selecting a product in line with the requirements of Sync Consulting Co., based on the evaluation of three products through application of the proposed model.

## 7. Conclusions and Recommendations

This work is aimed at helping for organizations using or wishing to implement ITS, especially FLOSS-ITS, since no similar works have been found in the bibliography consulted. An adaptation of MOSCA was

proposed for evaluation of the FLOSS-ITS product, by assessing functionality, reliability and usability, and adapting original MOSCA metrics to ITS related metrics. This was based on the features proposed by ITIL for this type of systems and FS features, which lays the foundation for evaluation of this type of products, which might be even broadened with specific requirements of the HD application scope. Based on this model formulation, three ITS were evaluated to select the most suitable product in line with the model proposed for the purpose of extending its abilities over product's potential weaknesses, and to include the product within the business solutions offered by the Venezuelan company Sync Consulting, Co. ITS subjected to evaluation were JTrac, RT, including RTFM and RTIM extensions, and OTRS::ITSM. The product selected was OTRS::ITSM, which obtained the highest values for the three categories (Functionality, Usability and Reliability). Additionally, OTRS::ITSM includes other applications that give added-value to the product, such as applications to verify system use, calendar agendas for HD users, and mail interaction over IMAP protocol, among others. However, within the scope of ITS, OTRS::ITSM may be enhanced,

especially regarding Change Management processes and communication device interactions. We recommended the company to evaluate the possibility of developing extensions to enhance the product's ability in such areas.

## Acknowledgements

This research was financed by Universidad Simón Bolívar, through Project DID S1-IN-CAI-003-07.

## References

- [1] Auxilor, Inc. (N/K). Implementing Your Help Desk. A Practical Guide. Auxilor, Inc.
- [2] Basili, V. R., Caldiera, G., Rombach, H. D. (1994). Goal Question Metric Paradigm. In J. J. Marciniak (ed.), *Encyclopedia of Software Engineering*, John Wiley & Sons.
- [3] Baskerville, R. (1999). Investigating Information Systems with Action Research. *Communications of the Association for Information Systems*, October, 2, 19, 1-32.
- [4] Checkland, P. (1993). *Pensamiento de Sistemas, Práctica de Sistemas*. Grupo Noriega Editores.
- [5] Dromey, G., "Comering the Cimera", *IEEE Software*, 1996, 33-43.
- [6] Firestone, J. (2000). Knowledge Base Management Systems and The Knowledge Management Warehouse: A "Strawman". Executive Information Systems, Inc.
- [7] FlossMetrics Project. Retrieved: August 25, 2007. Web Site: <http://flossmetrics.org/>
- [8] Foo, S., Hui, S. & Leong P. (2002). Web-based intelligent helpdesk-support environment. *International Journal of Systems Science*. 33(6). 389-402.
- [9] Free Software Foundation Inc. (2004). El sistema operativo GNU - Libre, no gratuito. Retrieved, August 15, 2007, from *El Sistema Operativo GNU - Fundación para el software Libre*. Web Site: <http://www.gnu.org/home.es.html>
- [10] González, L., Giachetti, R. & Ramirez, G. (2004). Knowledge management-centric help desk: specification and performance evaluation. *Decision Support Systems*. 40(2). 389-405.
- [11] ISO/IEC 9126 1.2, (1998), *Information Technology-Software Product Quality, Part 1, Quality Model*, ISO/IEC JTC1/SC7/WG6.
- [12] ISO/IEC TR 15504-2, (1998), *Information Technology-Software Process Assesment, Part 2, A Reference Model Processes and Process Capability*, Canada, ISO/IEC JTC 1/SC 7.
- [13] JTrac (2006) JTrac. Web Site: <http://jtrac.info>
- [14] Kang, B., Yoshida, K., Motoda, H. & Compton, P. (1997). Help Desk System With Intelligent Interface. *Applied Artificial Intelligence*. 11. 611-631.
- [15] Kitchenham, B. (1996). Evaluating Software Engineering Methods and Tools. Part 1: The Evaluation Context and Evaluation Methods. *ACM SIGSOFT -*

*Software Engineering Notes*, 21, 1, 11-14.

- [16] Middleton, I. (1996). Key Factors in Help Desk Success (An analysis of areas critical to help desk development and functionality.). *British Library R&D Report 6247*, The British Library.[18]
- [17] Ortega, M., Perez, M. & Rojas, T. (2002). A Systemic Quality Model for Evaluating Software Products, 6th World Multiconference on Systemics, Cybernetics and Informatics, Orlando, Florida, Vol. I, 371-376.
- [18] Mendoza, L., Pérez, M., Grimán, A. and Rojas, T. (2002) Algoritmo para la Evaluación de la Calidad Sistémica del Software Memorias de las 2das. Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC 2002) Salvador, Brasil.
- [19] OTRS (2003) Open Ticket Request System, Web Site: <http://otrs.org>
- [20] Perez, M., Rojas, T., Mendoza, L. & Griman, A., (2001) Systemic Quality Model for System Development Process: Case Study, Seventh Americas Conference on Information Systems - AMCIS 2001, 1297-1304.
- [21] Perez, M., Grimán, A., Mendoza, L. & Rojas, T. (2004) A Systemic Methodological Framework for IS Research. *Proceeding of the Tenth Americas Conference on Information Systems (AMCIS 2004)*, August, USA, 4374-4383.
- [22] Perez, M., Grimán, A., Mendoza, L. & Rojas, T. (2006) Human Perspective in System Development Quality. 12th Americas Conference on Information Systems (AMCIS). Acapulco, México. August.
- [23] Pressman, R., (2002) *Ingeniería del Software: Un enfoque práctico*, 5th. edition, España: McGraw-Hill Interamericana.
- [24] RT (1996) RT: Request Tracker, Web Site: <http://www.bestpractical.com>
- [25] Software Quality Observatory for Open Source Software. Retrieved: August 25, 2007, Web Site: <http://www.sqo-oss.eu/>.
- [26] Wendle, K. (N/K). The HP OpenView approach to Help Desk and Problem Management. "Preparing Today for the Help Desk of Tomorrow". HP OpenView.
- [27] Mendoza, L., Pérez, M., Grimán, A. & Rojas, T. (2002) "Algoritmo para la Evaluación de la Calidad Sistémica del Software", 2das. Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC 2002), Salvador, Brasil. November. 1-11.
- [28] Free Software Foundation Inc. (2007). GNU General Public License, Retrieved: August 15, 2007, from *GNU – General Public License – GNU Project*. Web Site: <http://www.gnu.org/copyleft/gpl.html>
- [29] Sync Consultores C.A. (2007). Sync Consultores, Web Site: <http://www.sync.com.ve>