Citation: Rojas, S.A., Diosa, H.A. y Melgarejo M. (2013).
The KITE Model for Assessment of Academic Software Products. In: Ingenier´ıa, Vol. 18, No. 2, pp. 6–18

The KITE Model for Assessment of Academic Software Products

Sergio Rojas-Galeano
Grupo de investigación LAMIC
srojas@udistrital.edu.co

Henry Alberto Diosa
Grupo de investigación ARQUISOFT
hdiosa@udistrital.edu.co

Miguel Melgarejo
Grupo de investigación LAMIC
mmelgarejo@udistrital.edu.co

Abstract

We reflect on the topic of assessing the merit of software products developed by research groups within the academia. To this end, a model is proposed to define the score of an arbitrary software product. The model consists of four determinants, namely new knowledge dissemination effect (K), impact in target population (I), technological innovation (T), and engineering achievement (E). These determinants are integrated into a ”KITE” graphical model. The model admits both geometric and numeric interpretations, enabling decision makers to analyze profiles of software productivity for a particular academic unit from a quantitative or qualitative viewpoint. The ratings, which enable software to be scored regarding each determinant, are also described. Following the model, preliminary test lists are sketched as a proposal of measurement instruments for these scores.

Key words: Assessment of software products, technological innovation, academic groups productivity.

Resumen

Se presenta a continuación una propuesta para valorar productos de software desarrollados por grupos de investigación en el ámbito académico. Con este objetivo, se describe un modelo que consiste de cuatro ejes determinantes para la medición o valoración de un producto de software cualquiera: El efecto en la diseminación avance en el conocimiento (K), el impacto en la población usuaria potencial (I), la innovación tecnológica (T) y los aspectos de calidad del producto desde la perspectiva de la disciplina de la Ingeniería de Software (E). Estos determinantes se integran en un modelo gráfico que hemos denominado KITE. El modelo admite tanto interpretación numérica como geométrica para facilitar, a los tomadores de decisiones, el análisis de perfiles de productividad de software de una unidad académica, desde el punto de vista cuantitativo o cualitativo. Las escalas de las valoraciones para cada determinante son también descritas de manera sucinta y están acompañadas de listas de chequeo preliminares, esquematizadas como una propuesta de instrumentalización de la medición en concordancia con el modelo.

Fecha recibido: sep. 4/2013
Fecha modificado: oct. 28/2013
Fecha aceptado: oct. 31/2013

Palabras claves: Evaluación de productos de software, innovación tecnológica, productividad de grupos académicos.

1. Introduction

Productivity of research groups is associated with the amount and calibre of their intellectual output, particularly in the form of research papers, books, patents, screen- plays, musical compositions, and so forth. Quality and visibility assessment of such output is achieved usually by means of self-regulatory dynamics of knowledge dissemination within academic communities. For example, an influential paper is expected to be published in a journal with a high impact factor, which guarantees a strict peer-review examination by some of the fellow experts in the corresponding field. Another example would be the case of a classic book whose permanent reader demand requires the published in several editions on a regular basis. Similar dynamics would apply to the other products (patents, scripts, scores). In other words, the assessment of these products is a relatively straightforward task using such implicit, and at the same time, objective mechanism (although well-documented criticisms and flaws have been presented for established metrics such as the impact factor [8], [11], [13], [16]).

In contrast, the case of academic software products seems blurry. This is because software is a product usually confined to the limits of industrial output, not faculty output. To the best of our knowledge there is no clear definition of ratings for quality, dissemination, usefulness, innovativeness or other software attributes in an academic context. On the other hand, however, software can be an important factor when defining aspects such as the orientation of an academic unit (scientific or technological), its target population (local, national, worldwide), its standards of community-interaction and so on. Therefore, a model for objective evaluation and assessment of such products is necessary and must be made clear and available to the community. As any measurement instrument, such a model should allow calibration of the software pro- ductivity of a particular academic unit, and also be helpful in guiding their efforts towards the production of high-quality, valuable and user-friendly academic software. The latter is a challenging task. Software evaluation is usually confined to the in- dustry, where a wide range of metrics and estimation models have been proposed and consolidated [6], [12], [15]. There, software development is driven by profits. Within the academia however, software is developed on the basis of its contribution to re- search projects. In this context, software is driven by knowledge and innovation. The purpose of this paper is to discuss these elements and organize them into a model that can be used to evaluate the merit of software products and moreover, that serves as a tool for academic decision-makers when identifying the profiles of software productivity in their faculty, and also when making suitable policies to support, promote and reward any achievement accordingly.

2. The model

The difficulty in defining a model for software assessment within the academia lies in the fact that, besides the intrinsic complexity of software development, there are research-related factors that must be taken into account. Some examples of the sort of questions that may rise in this regard are as follows: Is the software product implementing and distributing new ideas or technologies in a given field or subject? Is the software product needed by, relevant to or widely-usable by the community it was designed for? Is the software product robust, fast, well-documented, available, safe, reliable, reusable? Or even better, to what extent is the software product complying with all of the previous attributes?

Our attempt to solve these questions is described in the following. We devised four determinants that are relevant to define the value a software product within the academia, namely, new knowledge dissemination effect (K), impact in target population (I), technological innovation (T), and engineering practices adopted during its development (E). These determinants can be geometrically combined as the axes of four quadrants (K-I, I-T, T-E, and E-K) in a two-dimensional plane. We assume an arbitrary software product can be rated in each determinant; then, by joining with straight lines the scoring marks in each determinant, a frame is obtained whose inner area would define the product final score (assessment). Such frame visually resembles the shape of a rhomboidal kite, which inspired the name of the model (see Figure 1). As we shall discuss in the next section, the maximum ratings in each determinant would be different, as also would be the actual contribution to the total area from the triangular regions in each quadrant. Hence, the maximum contributions for a total score (area) of 100% according to the efforts devoted to the development as well as to the outlook for potential use of the software, were appraised as follows[1]:

- **E-T quadrant.** The area of this component would depend on the quality of software engineering attributes adopted during product development, and also on the degree of technological innovation it comprises. Its measurement would be firmly supported by the standards embraced within the software industry, particularly in terms of what is commonly accepted as good engineering practices as well as the widely-known innovation frameworks. The ratings of the product

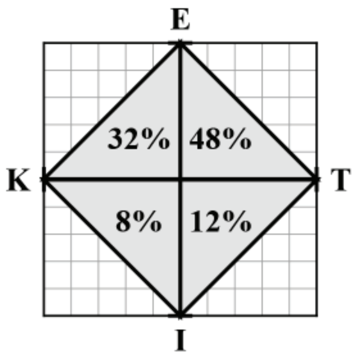

Figure 1: The KITE model for software assessment

1 We remark that the estimation of these proportions was made according to our expectations on the merits of each determinant for a first-class research-oriented academic institution. However these proportions can be thought of as model parameters adjustable to other academic institution profiles (training-only, technical or vocational schools).

regarding these aspects would be the core of the assessment model, thus allowing a maximum contribution to the total score of 48%.

- **K-E quadrant.** In the context of academia, software production would be ideally closely related to research. The purpose of measuring this aspect is to grant additional merit to high-quality software intended to support or promote the dissemination and application of new knowledge. Consequently a relevant contribution to the total score of up to 32% is allocated to this component.
- **I-T quadrant.** The incorporation of the I determinant is aimed at promoting wider distribution and awareness of academic software. Therefore, this component is intended to evaluate the scope and impact on local and external target- communities, where the software product may become useful technology. We decided to associate a maximum contribution of 12% of the total score to this aspect.
- **K-I quadrant.** Similar to the I-T component, the purpose here is to give some credit to the extent and capacity of the software to disseminate new embod- ied knowledge (if any) to its intended research audience. Although secondary from an industrial-oriented viewpoint, this aspect is regarded as a particularly important goal for academic-oriented software. Hence, a minor yet relevant maximum contribution of 8% to the total score is assigned.

Now, an arbitrary software product can be assessed by computing the score ($S_\Diamond$) obtained from the sum of the areas in the resulting triangular regions in each quadrant. Let us denote the ratings of the software product in each determinant as *K, I , T , E,* and the area of triangles *K -E-T* and *K -I -T* as Δ_E and Δ_I respectively; then the final score is straightforward to compute:

$$\begin{aligned} S_\Diamond &= \triangle_E + \triangle_I \\ &= \left(\frac{KE}{2} + \frac{TE}{2}\right) + \left(\frac{KI}{2} + \frac{TI}{2}\right) \\ &= \tfrac{1}{2}(K+T)(E+I) \end{aligned} \tag{1}$$

It is worth noting that the model can be regarded from two different points of view. In the first one, the kite can be split into an upper half and a bottom half. In this view the maximum contribution of the *K -E-T* triangle (upper half) accounts for 80% of the total score. This percentage would be highly correlated to the E score, in other words, to the software engineering effort and practices involved during the development of the product, which is intuitively the most relevant aspect to be assessed in a software product. Notice that the final contribution of this triangle would be modulated by the degree of technological innovation as well as the new-knowledge injection. On the other hand, the *K -I -T* triangle (bottom half) contribution represents the impact of the software product in terms of visibility and usage, once again, modulated by the *K* and *T* scores. The latter is an important aspect of communication in academia, which motivated the inclusion of this component. Nevertheless, we consider it as a less relevant target for software development, hence the smaller score allocation from the total score (a maximum of 20%).

From another point of view, the kite can be split into a right-hand half and left- hand half. In this view, the *E-T -I* triangle (right half) would account for a maximum of 60% of the total score. This reflects the common scenario of a software froduct that becomes technology of choice, again depending on its engineering maturity and visibility impact. A higher weight in the final scoring is assigned to this aspect. The *E-K -I* triangle (left half) however, also makes a contribution of up to 40% of the total score. This is specifically aimed at academic contexts where promoting the association between software production and research is extremely relevant.

In order to compute Equation (1), we have designed a number of tests that determine the merit of the software product in each determinant. The criteria, range, ratings and rationale of such tests are discussed next.

3 Determinants range and ratings

Let us assume that the finest academic software product will shape up a kite with an area equivalent to 100 points. Thus, we define the following range for the ratings in each determinant:

$$E \in [0, 16], K \in [0, 4], T \in [0, 6], I \in [0, 4]$$

Observe that within those ranges, a first-rate academic software product $S^\star$ would get the ratings $E = 16$, $K = 4$, $T = 6$, $I = 4$. By setting up unitary scales in each axis, the KITE model of $S^\star$ would be rendered as in Figure 2. Notice that despite the irregularity of the shapes in the resulting triangles, this arrangement will preserve the maximum proportion of contributions mentioned in Figure 1. In fact, the area (score) of $S^\star$ would amount to:

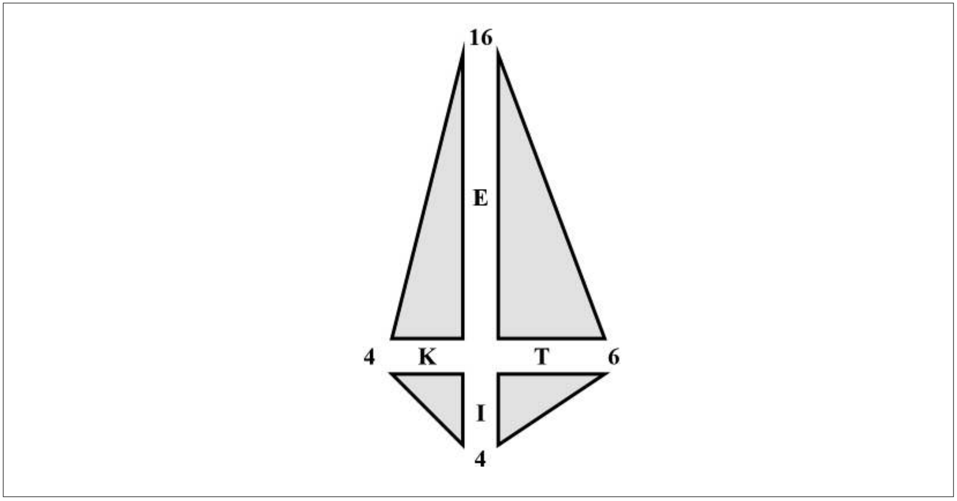

Figure 2: The unnormalised kite for S*

$$\begin{aligned} S^{\star}_{\diamondsuit} &= \triangle_{KE} + \triangle_{ET} + \triangle_{TI} + \triangle_{IK} \\ &= \left(\tfrac{16(4)}{2} + \tfrac{16(6)}{2} + \tfrac{4(6)}{2} + \tfrac{4(4)}{2} \right) = 100, \end{aligned}$$

which also equals the score that would have been computed trough Equation (1). The remainder of this section focuses on the rationale behind these ranges, as well as on the proposal of tests designed to measure the ratings in each determinant. Assuming that engineering should be the crux behind of software creation process, we shall proceed first with the E determinant.

3.1 The E determinant

Academic software is on its own nature, commonly regarded as the early prototypes of proof-of-concept or proof-of-technology endeavours that stem from non-industrial academic factories (in the best scenario) or more frequently, from academic research groups. Nevertheless, we believe that academic software production must be guided by the principles of software engineering so as to guarantee to a certain extent the development of high-quality products that would eventually embark on a feasible trail of future industrial development. The more quality attributes the product achieves, the more potential benefit or profit the academic group receives in return for their invested efforts and costs. In the light of such remarks, this determinant is pivotal to the model, since it measures the degree of fulfillment of software engineering practices adopted during the making. Its purpose is to motivate compliance with minimal standards in order to guarantee the development of valuable software, software that would be really helpful or appreciated by its target community.

The measurement instrument of this determinant is inspired by established and widely-known software estimation models in the industry. The instrument consists on a series of check-lists for a number of technical attributes defined in [3]. The designs of these tests are based on the models of [6], [12], [15], adapted in scope and pertinence to an academic context. The definitions of each quality attribute $\{E_i\}_{i=1}^{8}$ are given below, where each E_i is a number between [0, 2]. The tests designed as measurement tools for these attributes are provided in different guides or technical books as [14], [12] while the preliminary KITE's checklists for this determinant are showed in Appendix B.

***E_1*: Robustness.** Resistance against improper, malicious or illegitimate inputs or operating environments for the software.

***E_2*: Maintainability-Extensibility.** Simplicity in updating the software product either by adding new features or changing existing (possibly flawed) features, or else, in scaling up its capabilities.

***E_3*: Performance.** Efficiency in managing machine resources (processor time, memory, bandwidth, etc.) in order to accomplish the intended purpose of the software, specially for large data volumes.

E_4: **Usability.** User-friendliness: how easy or convenient to use the software product actually is.

E_5: **Integrity.** The quality of maintaining consistency as well as safeguarding the information processed by the software product.

E_6: **Portability.** Possibility of running the software product on more than one operating system or hardware platform with minimal effort.

E_7: **Compatibility.** Support of input and output data formats and persistence schemes used by the same software in previous versions, or by other related software tools, without major conversion or modifications required.

E_8: **Documentation.** The availability of technical documentation related to the development and utilisation of the software product (design diagrams, listings, test reports, manuals, user-guides, online help, etc.).

The final rating in this determinant would be given by the sum in Equation (2). The range would be clearly, $E \in [14]$.

$$E = \sum_{i=1}^{8} E_i \tag{2}$$

3.2 The K determinant

In contrast to industry settings, research-driven development of software can be a relevant goal for an academic unit. Consequently, the model was designed so as to assign some merit to a software product developed as a dissemination device for new knowledge originated in either basic or applied research. In this respect the *K* determinant modulates the contribution of the E determinant to the overall assessment of the product, since the contribution of the Δ_{KE} would be $+\frac{KE}{2}$. In other words, the more firmly engineered and research-supporting is the product, the higher the score it will obtain.

In scoring this determinant we took some inspiration from the widely accepted practice of publishing scholarly papers, which is closely related to the premise of knowledge dissemination stated above. Thus, the following two criteria were defined:

K_0 : **Ingenuity.** Is the software product a realization of a previously unknown natural, social, organizational, scientific, algorithmic or computing model?

K_1 **: Dissemination.** To what extent the fundamental research associated to the software product has been communicated to the relevant scientific community?

The first criteria is an indicator variable that characterises the software product as either the output of a research study or not ($K_0 \in \{0, 1\}$). The second criteria is associated to the dissemination of the research foundation that prompted the creation of the software product, i.e. its publication in scholarly peer-reviewed journals, con- ferences or scientific repositories. The range of this variable is $K_1 \in [0, 4]$. The score of the determinant is computed using Equation (3). Evidently, $K \in [0, 4]$. The checklists designed to rate these criteria are given in Appendix A.

$$K = K_0 K_1 \tag{3}$$

3.3 The T determinant

This determinant is aimed at measuring the degree of technological innovation achieved through the software product. One of the difficulties in defining this measuring aspect is that there is no definitive agreement regarding the meaning of innovation in the software industry [4]. Based on discussions reported recently on the literature [4], [5], [9], [10] and also on our own experience in the field, we settled for the following viewpoint. Key to the concept of innovation is the formulation of a novel idea. With respect to the problem at hand, a novel idea would be realized when the software product is proposing a new usage of a known technology (e.g. in application soft- ware, when transferring technology from one field of application to another, or from one community to another), or when the software itself yields new technology (espe- cially in system or embedded software when new architectures, protocols or models of computation are proposed). Thus, technological innovation would refer to the process of embarking on, testing, adjusting and refining that novel idea, in such a way that its materialisation within a new context or process produces a positive effect [9].We focus now on defining how to measure technological innovation.

We restrict ourselves to the concept of product innovation since other facets of innovation (process, marketing, or organizational) in our opinion are not intrinsically relevant to a software product. Building upon the approach in [10], which has been rigorously validated in the industry, the following aspects are considered, suitably adjusted to the context of academic software (they are denoted $\{T_i\}_{i=1}^{6}$ with $T_i \in [0, 1]$). The corresponding checklists are reported in Appendix D.

T_1: **Novelty.** Does the software product embody new technology or a previously unknown application (to a field or problem) of a known technology?

T_2: **Scope.** Is the software product new to the world/country/academic institution?

T_3: **Competitiveness.** In what ways does the software product outperform other known similar products or previous versions (aesthetic/core/performance)?

T_4: **Continuous improvement.** To what extent are the software attributes improved over previous versions (in any attribute $E_1 - E_8$ or in saving costs)?

T_5 : **Originality.** Is it a self-contained software product (pioneering innovation), an add-on to an existing installation (incremental innovation) or an upgrade (boosting innovation)?

T_6: **Endogenicity.** Is the source of innovation endogenous to the academic unit, exogenous or combined?

The final rating would be given by Equation (4) in a scale $T \in [0, 6]$.

$$T = \sum_{i=1}^{6} T_i \tag{4}$$

3.4 The I determinant

This determinant was included to adjust the assessment of the software product with respect to aspects such as visibility, availability, openness and utilisation by its target community. These aspects are combined into what we regard as the "impact" of the software product. The rationale is that the merit of the software product should be be proportional not only to the quality of the software *per se* but also to the impact it is having or might have on its latest or future releases. In this sense the model attempts to promote, on the one hand, continuity of software projects for the academic group that creates the software product (to allow improvement through new technologies and functionalities as the product spreads and popularises within its target community). On the other hand, this will motivate the academic units to evaluate the relevance, effect and scope of their associated software authors or factories on a regular basis.

The definition of these criteria, denoted $\{I_i\}_{i=1}^{4}$ with $I_i \in [0, 1]$, was tailored tothe context of academia as explained below. The checklists designed to measure this determinant are provided in Appendix C.

I_1 : **Coverage.** The extent to which the software product is visible for its intended audience (local/regional/world-wide).

I_2 : **Availability.** Convenient support regarding the deployment of the software prod- uct (release/versioning/ download/installation).

I_3 : **Utilisation.** Evidence of usage and positive feedback from the community (academic/otherwise).

I_4 : **Openness.** The level of restriction when distributing, using or changing the source code of the software product (open source/source available/proprietary).

The final rating associated to this determinant is computed using Equation (5), resulting in a range $I \in [0, 4]$.

$$I = \sum_{i=1}^{4} I_i \tag{5}$$

4 Conclusions

Increasing rates of academic research output, due to the growth of their corresponding software production lines, motivate the proposal of comprehensive and impartial models for the evaluation of such products so as to provide unbiased assessment of their quality, impact, innovation and originality. Furthermore, such models may become useful tools for researchers and decision-makers alike. Researchers can use them to open up new perspectives in considering academic software development seriously (as one of their career aims). Decision-makers can use them to identify the suitable software production profiles of their institutions, and also to support and reward their faculty accordingly. We expect the rationale behind the KITE model described in this paper to contribute to taking initial steps towards such scenario.

In addition to its hypothetical use as an academic software productivity-profiling tool, the model can also be considered as an evaluation instrument within compensation schemes in the academia. Such schemes are designed to award faculty with salary bonuses proportional to the quality and visibility of their academic products. A well-balanced appraisal between the merit and the reward of software products in this context will encourage academic groups to develop first-class research-oriented or technology-oriented software. The latter is likely to have a positive impact on academic productivity, as it has been already highlighted in various studies in other areas [1], [7].

As a final remark with respect to the KITE model itself, it is worth noting that its modularity and conceptual abstraction make it feasible to extend the model to the assessment of other kinds of academic products, such as hardware, business processes, integrated circuit layouts, industrial designs and technical standards or any other engineering prototypes. In fact, these products share a common technological nature as artifacts resulting from engineering and scientific principles. The extension of the model would require a deeper insight into some definitions in order to achieve generalization, perhaps as a meta-model formulation. We anticipate that, in such formulation, the notions ascribed to the E determinant would be pivotal (those concerning the rigorous exercise of the relevant engineering branch or other involved disciplines in order to create a first-class product). The path to develop this meta-model is still under discussion considering that either a deductive or an inductive construction may be possible. For the time being, we are working on the concrete instruments needed to make the KITE model operative. Instrumentation and validation of the model will be reported in a forthcoming study.

References

[1] Elie A. Akl, Joerg J. Meerpohl, Dany Raad, Giulia Piaggio, Manlio Mattioni, Marco G. Paggi, Aymone Gurtner, Stefano Mattarocci, Rizwan Tahir, Paola Muti, and Holger J. Schnemann. Effects of assessing the productivity of faculty in academic medical centres: a systematic review. *Canadian Medical Association Journal*, 184(11):E602–E612, 2012.

[2] Dino Dai Zovi Chris Wysopal, Lucas Nelson and Elfriede Dustin. *The Art of Software Security Testing. Identifying Software Security Flaws*. Symantec-Press, 2007. [3] Colciencias. Resolución No. 00285 de 2004, 2004.

[4] Henry Edison, Nauman Bin Ali, and Richard Torkar. Towards innovation measurement in the soft- ware industry. *J. Syst. Softw.*, 86(5):1390–1407, May 2013.

[5] Organisation for Economic Co-operation, Development, and Statistical Office of the Euro- pean Communities. *Oslo manual: guidelines for collecting and interpreting innovation data*. Mea- surement of Scientific and Technological Activities. 2005.

[6] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. Software qualities and principles. In *The Computer Science and Engineering Handbook. Second Edition*, pages 101–1 a 101–25. 2004.

[7] Claudia Gonzalez-Brambila and Francisco M. Veloso. The determinants of research output and impact: A study of mexican researchers. *Research Policy*, 36(7):1035–1051, September 2007.

[8] Bob Grant. New impact factors yield surprises. *The Scientist*, 2010.

[9] Paul H. Jensen and Elizabeth Webster. Another look at the relationships between innovation proxies. *Australian Economic Papers*, 48(3):252–269, 2009.

[10] Gary Jordan and Esbjrn Segelod. Software innovativeness: outcomes on project performance, knowledge enhancement, and external linkages. *R & D Management*, 36(2):127–142, 2006.

[11] Eugenio Matijasevic. Acta Médica Colombiana y las fuerzas de producción social. *Acta Medica Colombiana*, 35:107 – 112, 09 2010.

[12] J. McGovern. *A Practical Guide to Enterprise Architecture*. The Coad series. Prentice Hall PTR, 2004.

[13] Tom Misteli. Eliminating the impact of the impact factor. *The Journal of Cell Biology*, 201(5):651–652, 2013.

[14] Glenford J. Myers. *The ART of SOFTWARE TESTING*. John Wiley and Sons Inc., 2004.

[15] I. Sommerville. *Software Engineering*. International computer science series. Addison-Wesley, 2007.

[16] A. W. Wilhite and E. A. Fong. Coercive citation in academic publishing. *Science*, 335(6068):542–3, 2012.

Appendix A. K tests

K attributes	
K_0	The software implements a previously not known natural, social, organizational, scientific, algorithmic or computing model.
K_1	The research study originating the software has been disseminated to the academic community in recognised scholarly journals or well-known academic conferences on a relevant field.
K final score: $K = K_0 K_1$	

Appendix B. E tests

E attributes	
E_1	Resistance towards invalid input data or incorrect commands.
	Fault-tolerance to operating system or hardware crashes.
	Agreement between the software product and its specification.
	The software product can be adjusted to unforeseen changes in its underlying operating environment.
E_2	The software product admits incorporation of external changes with low effort.
	Support to functionality extensions in future versions.
	A well-defined version control procedure for the software.
	The software product adheres to architectural styles allowing easy scalability (e.g. blackboard style, publish-subscribe style).
	Capability of scaling-up designs in order to improve performance.
	The software uses configuration files for operating parameter settings.
	The user interface is decoupled from the domain logic.
	The software can be easily debugged.
	The software exhibits adaptive manteinability.
	Well-documented architectural design.
E_3	Compliance with stated expected response times for each use-case or functionalities.
	The software delivers within admissible stated response times (average, minimal, maximal).
	The software is able to handle concurrency.
	Usability under lower or bad rates of performance.
	Compliance with stated expected response times for batch processing, if any.
E_4	The user interface is self-explained or easy to understand.
	The user interfaces is customizable.
	Depth vs breadth ratio of user options is appropriate.
	The software can be adapted easily to new operating systems or hardware.
E_5	The software mitigates the impact of expected security breaches.
	The software properly catches security breaches.
	High safety level against known vulnerabilities.
	High data-reliability level for simulated operating system or hardware breakdowns.
	High data-integrity level for unexpected breakdowns or unauthorized access.
E_6	The software is platform-independent.
	New layers of software can be added to the original product.
	The software provides portability to other hardware platforms.
E_7	The software supports standard technologies for system integration.
	The software architecture (subsystems or components) is well-documented and comprehensible.
	The software provides versioning information for subsystems and components.
E_8	Functional model documentation is provided.
	Structure, domain and persistance models are well-documented.
	Dynamic and behaviour models are well-doucmented.
	User guide and administrator manual are well-documented and comprehensible.
	The software is equipped with extensive and friendly online help assistance.
E final score: $T = \sum_{i=1}^{4} I_i$	

Appendix C. I tests

I attributes	
I_1	The software has been announced or advertised in worldwide/local coverage.
	The software is released in English language or is parameterisable to a suitable language for the intended public
I_2	The software is hosted and distributed in a public software repository or in private dedicated down- loading server.
	The software is provided with an install/setup assistant application.
I_3	The reported number of different users of the software is larger than 10/100/1000.
	More of 50% of users have given positive reviews to the software.
I_4	The software was released with an open source or proprietary license.
I final score: $I = \sum_{i=1}^{4} I_i$	

Appendix D. T tests

T attributes	
T_1	The software is a new technology as such or an unknown application to a field or problem of a known technology.
T_2	The software is new to the world/country/institution.
T_3	The software implements a distinct core or aesthetic feature compared to similar products or previous versions.
	The software improves over memory consumption or execution times compared to similar products or previous versions.
T_4	The software improves any of its quality attributes $E_1 \cdots E_8$.
	The software saves in installation costs.
T_5	The software is released as a new installation or an upgrade version.
T_6	The idea originating the innovation realised by the software was conceived by authors affiliated to the institu- tion or affiliated to an external institution or both.
T final score: $T = \sum_{i=1}^{6} T_i$	

Sergio A. Rojas-Galeano

He has received a BEng. in Systems Engineering from National University of Colombia, a MSc. in Intelligent Systems from University of London (2005), and a PhD. in Computer Science from University of London (2009). His areas of research interest include machine learning, data mining, optimization, and scientific software. He is currently an assistant professor at the District University of Bogota and also a researcher in the LAMIC research group. e-mail: srojas@udistrital.edu.co

Henry Diosa

He has received a BEng. in Systems Engineering from National University of Colombia, a MSc. in Information Sciences from District University of Bogota, and a PhD. in Computing Science from Valle University. His areas of research interest include software architecture and software engineering. He is currently an assistant professor at the District University of Bogota and also the leader of the ARQUISOFT research group. e-mail: hdiosa@udistrital.edu.co

Miguel Melgarejo

He has received a BEng. in Electronics Engineering from District University of Bogota, a MSc. in Elec- tronics Engineering from Los Andes University, and is currently a PhD in Electronics Engineering at Javeriana University. His areas of research interest include machine learning, fuzzy logic systems, com- plex systems and scientific software. He is currently an assistant professor at the District University of Bogota and also a researcher in the LAMIC research group. e-mail: mmelgarejo@udistrital.edu.co