Dyna
ISSN: 0012-7353
dyna@unalmed.edu.co
Universidad Nacional de Colombia
Colombia

Zapata-Jaramillo, Carlos Mario; Torres-Ricaurte, Diana María
Test effort: A pre-conceptual-schema-based representation
Dyna, vol. 81, núm. 186, agosto, 2014, pp. 132-137
Universidad Nacional de Colombia
Medellín, Colombia

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS

DYNA
http://dyna.medellin.unal.edu.co/

Test effort: A pre-conceptual-schema-based representation

Carlos Mario Zapata-Jaramillo [a] & Diana María Torres-Ricaurte [b]

[a] *Facultad de Minas, Universidad Nacional de Colombia, Colombia. cmzapata@unal.edu.co*
[b] *Facultad de Minas, Universidad Nacional de Colombia, Colombia. dimtorresri@unal.edu.co*

Received: September 2th, 2013. Received in revised form: March 21th, 2014. Accepted: June 25th, 2014.

Abstract

Software testing is intended to guarantee the quality of software products. By executing a test suite, we can identify and correct defects in software products. Several methods are used to estimate the effort invested in testing. Each method comprises many concepts in determining the test effort. However, the graphical representations of such methods barely represent the different concepts involved. Specifically, some formulas are missing, avoiding the possibility of performing test effort calculations. In this paper, we identify the concepts involved in measuring the test effort. The concepts are obtained from a state-of-the-art review. Finally, we propose a representation for integrating such concepts by using pre-conceptual schemas, a kind of diagram devoted to the domain knowledge representation in a natural-like language.

Keywords: testing; effort; factor; pre-conceptual schemas.

El esfuerzo en pruebas: Una representación basada en esquemas preconceptuales

Resumen

Las pruebas garantizan la calidad en los productos de software. Mediante la ejecución de un conjunto de pruebas se pueden identificar y corregir defectos presentes en los productos de software. Existen diferentes métodos que estiman el esfuerzo invertido en las pruebas. Cada método valora distintos conceptos en sus estimaciones. Sin embargo, las representaciones graficas de estos métodos muestran los conceptos particulares sin incluir mayor detalle. Específicamente, algunas de las fórmulas usadas para obtener la medida del esfuerzo se pierden en estas representaciones, por lo que no es posible calcular el esfuerzo solo con la información dispuesta en estos gráficos. En este artículo se identifican los conceptos presentes en la medición del esfuerzo de pruebas, que se obtienen de la revisión de varios métodos. Finalmente, se propone una representación que integre estos conceptos mediante esquemas preconceptuales, que constituyen un tipo de diagramas para la representación del conocimiento de un dominio en un lenguaje cercano al natural.

Palabras clave: pruebas, esfuerzo, factor, esquemas preconceptuales.

1. Introduction

Software testing is an intermediate process of the software development lifecycle. This process is intended to find defects of the software product, ensure software quality, and convince the customer that the product fulfills the specifications and functionality specified [1].

Test coordinators use the test effort estimation to plan their resources and schedules [2]. By using this estimation, test coordinators can suggest strategies for optimally allocating resources and minimizing execution times [3]. The test effort estimation is measured in man-per-time units, where time is necessary to execute a test suite.

Several test effort estimation methods have been proposed. Most of them are based on concepts involved in measuring the test process effort. However, there is no common agreement about the concepts to-be-included in the methods for estimating the test effort.

In order to integrate the concepts involved in measuring the test process effort, in this paper, first, we present the more representative concepts of test effort estimation methods obtained from the state of the art. Then, we propose a knowledge representation about the test process effort, in which we summarize the concepts and their relationship by using the so-called pre-conceptual schemas [4].

The remainder of this paper is organized as follows: in Section 2 we present the conceptualization of the test process effort, software measurement, and the pre-conceptual-schema-based representation. The graphical

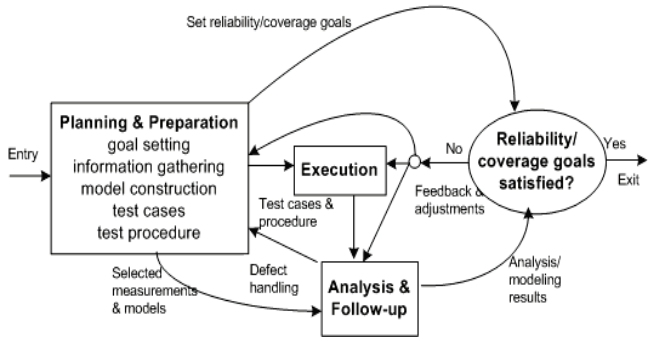

Figure 1. Generic testing process. Source: [5]

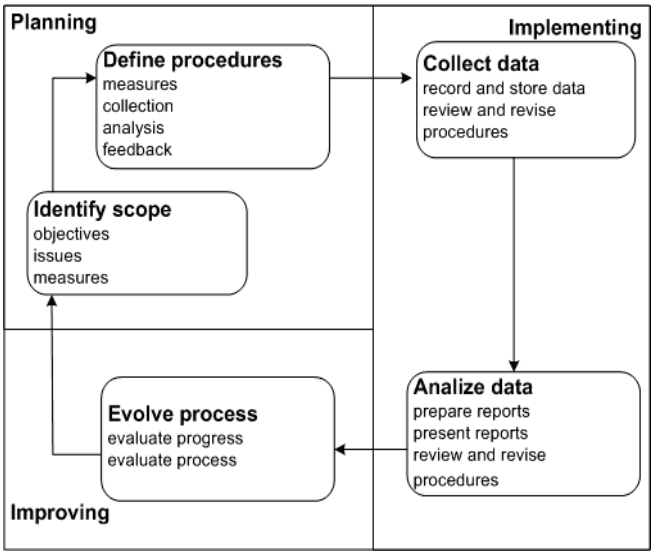

Figure 2. Software measurement process. Source: [6]

representation about test effort estimation is presented in Section 3. Then, in Section 4 we propose a pre-conceptual schema related to the concepts and relationships belonging to the test effort estimation. Finally, we present conclusions and future work.

2. Related work

2.1. Test process

The main goal of the test process is ensuring the software product is defect-free [1]. The main activities of this process are the execution of software and the observation of its behavior [5]. Once a failure is observed, the execution record is analyzed in order to locate the failure and its causes. These and other activities are illustrated in Fig. 1.

Test planning and preparation is the initial phase of the entire process. Test execution includes activities related to the observation and measurement of the software product. Test analysis and follow up include activities related to checking and analyzing failures and follow up about reviewing and removing such failures.

2.2. Software measurement and estimation

Software measurement has become a key aspect of the software engineering quality practices. Specifically, a measure is the number or symbol assigned to an entity. Each entity is characterized through an attribute according to an unambiguous rule [6]. Fig. 2 shows the activities to plan, implement, and improve a software measurement process.

Specifically, a metric is a quantifiable, directly observed software measurement. Such a measurement can be either calculated or predicted. A software quality metric is a function with software data inputs and numerical outputs. Such outputs represent to what extent a quality feature is present in the software development process [7].

Software metrics are mainly used with two purposes in mind: software measurement and estimation. Estimation is the process of predicting information about the software application based on incomplete, uncertain, and fuzzy input [8].

Test effort estimation is concerned with the test phase activities [6]. The test estimation process demands competence for software test manager in order to design and assess a cost-effective test strategy [9]. Test effort estimation includes the amount of resources, in terms of the man-per-time unit.

Test effort estimation is obtained with the factors affecting the effort spent on testing; some such factors are: software size, testing team experience, software complexity, etc.

2.3. Pre-conceptual schema representation

Knowledge representation emerged as an area of artificial intelligence for representing concepts about a particular domain. The aim of knowledge representation is specifying and analyzing reasoning about represented knowledge [10].

Pre-conceptual schemas [4] are intermediate knowledge representations between formal logic and natural language. The main symbols of pre-conceptual schemas are presented in the Fig. 3. Concepts are employed to represent nouns and nouns phrases; structural relationships are used to represent the verbs “be” and “have;” dynamic relationships include the so-called verbs “of activity or operation;” annotations or instances are possible values of the concepts; connections are used to link concepts to either a dynamic or structural relationship or vice versa; implications are cause-and-effect relationships; connectors annotations are used to link a set of instances; references include numbers that are used to link distant elements in the same schema [11].

3. Graphical representations of test effort estimation

During the past decades, some models and graphical representations of these models were created for estimating test effort. Here, we summarize the graphical representation related to either test effort models or techniques.

3.1. Software-size-based estimation

These approaches assume that the test effort estimation is directly proportional to the software size.

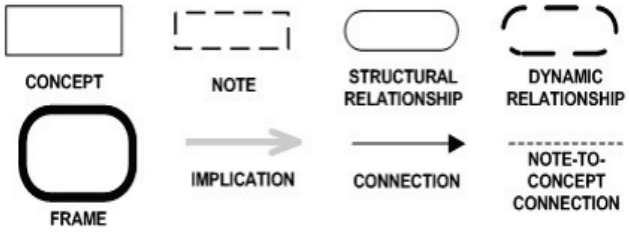

Figure 3. The main symbols of pre-conceptual schema. Source: [11]

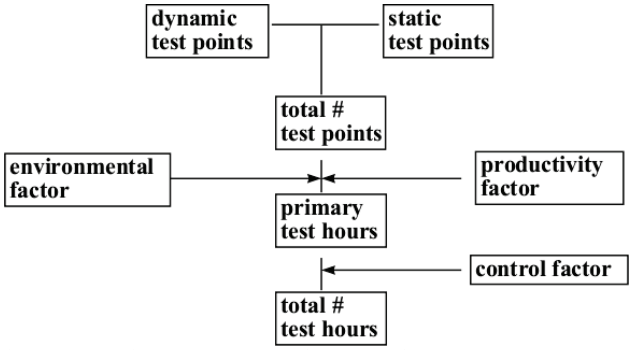

Figure 4. Overview of test point analysis procedure. Source: [12]

3.1.1. Test point analysis

Primary test hours are obtained from three factors: number of test points, environmental factor, and productivity factor [12]. The number of test points is calculated by using the amount of function points (FP) for each individual function. Total number of the test hours is the sum of primary test hours plus secondary test hours. Total number of the test hours represents total time needed to complete a set of the testing activities. In the Fig. 4 an overview about the test point analysis is shown.

The figure represents the flow of information needed for the test-point-based estimation. However, the way to obtain the factors is uncertain. Since the arrow syntax is intuitive, the graph representation appears incomplete because the concepts and their relationships can not be obtained.

3.1.2. Use-case-based estimation (test point analysis)

The number of tests cases is obtained by mapping the uses case specifications into the test process [13]. Each scenario and its exception flows for each use cases are inputs for test cases. After that, the effort estimation is calculated. In the Fig. 5, the project lifecycle model used is illustrated, but the graphical representation describing the test point method for test effort estimation is not presented.

First the test cases are enumerated, next different scenarios are evaluated for each test case and the effort is calculated.

3.2.1. Test-specification-based estimation

Input for the approach is a test suite and the output is a number which represents the effort in man-hours required

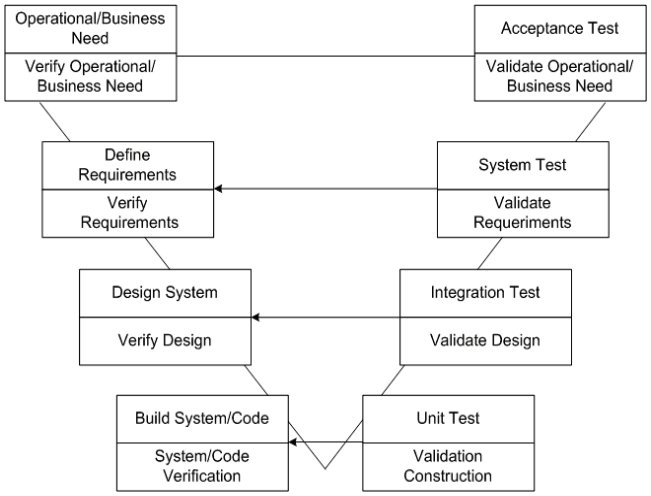

Figure 5. The V-Model. Source: [13]

for executing the test suite [2]. The test suite is defined in a controlled language. In Fig. 6 the estimation process is summarized.

3.2. *Test-case-enumeration-based estimation*

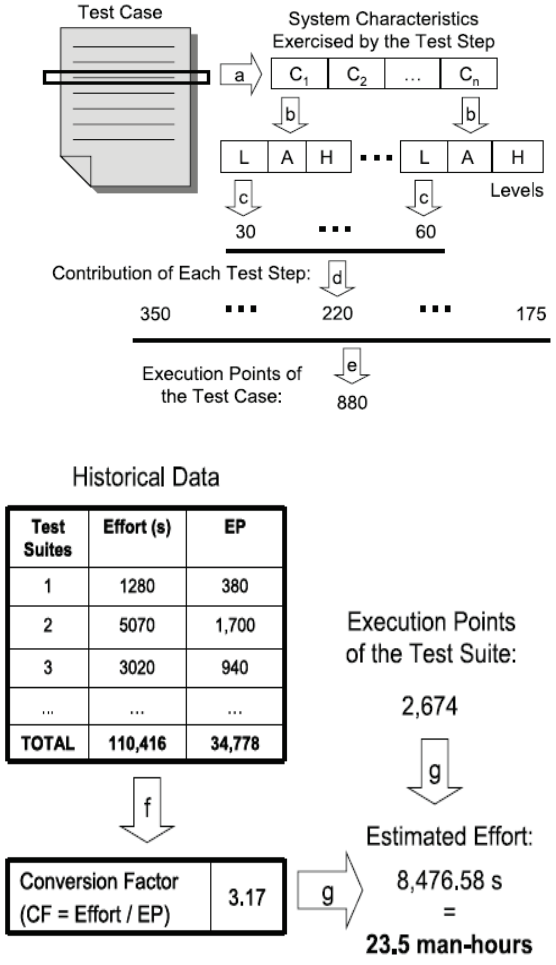

Figure 6. Estimation process with test specifications. Source: [2]

The two graphics describe the procedure of the method, however, the syntax is not defined; the ordinal scale is not defined; the origin of some values is not known and the format of such values are not standardized, finally, the concepts in two graphics are not connected.

3.2.2. Test-suite-execution-vector-based estimation

This approach is based on the fact that: "Similar tests have similar cost" [14]. First, test suite should be categorized by its attributes. Then, the similarity between a candidate and a definite test suite execution vector in the historical database is determined. After that, the estimation effort is changed to reflect an algorithm to predict the effort value for a given vector. Fig. 7 shows the tester rank model used in the approach and the Fig. 8 shows a big picture of the approach.

The graph uses the standard representation for its elements, but it is so general and a computational representation is difficult because all the information needed to estimate the effort is not present.

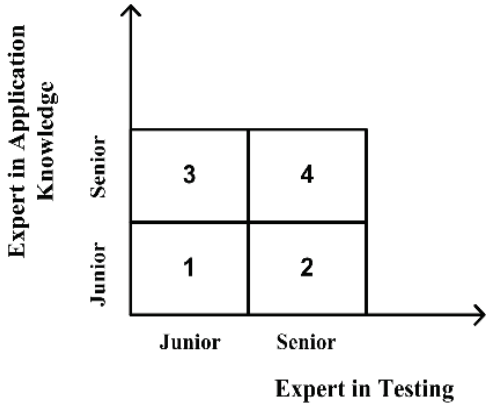

Figure 7. Tester rank. Source: [14]

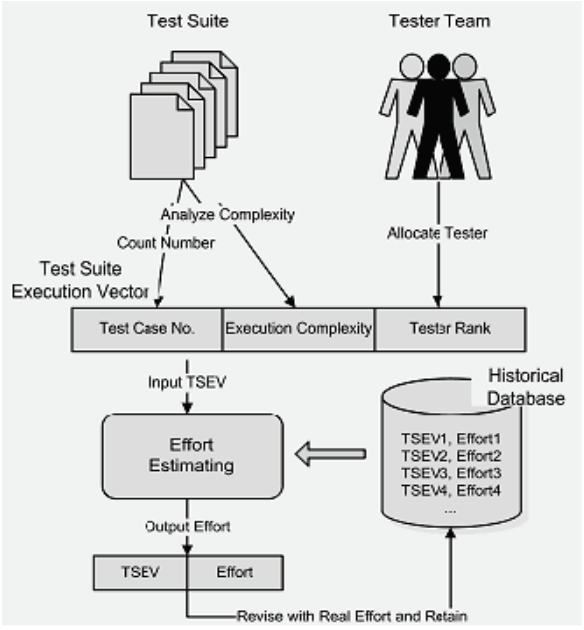

Figure 8. TSEV approach. Source: [14]

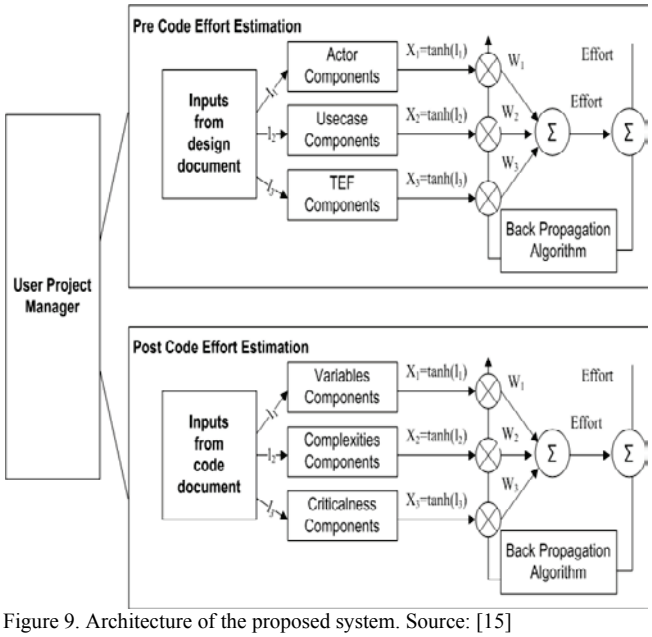

Figure 9. Architecture of the proposed system. Source: [15]

3.3. Testing-activity-based estimation

In this approach, we understand the project as a set of activities to be executed in the software development process.

3.3.1. Pre-coding- and post-coding-based estimation

The architecture of the model involves two components: pre- and post-effort estimation components and a learning rule, which is used as back propagation algorithm [15]. In Fig. 9 this approach is shown.

The graphical representation is close to the controlled language because the structure is similar to the block diagram. However, the syntax used is not defined, the symbols are used in others context but the interpretation cannot be made directly for this context, and the concepts in the boxes are not specified.

3.3.2. Three-phase-based estimation

The testing effort needed to assure the permissible number of field defects is estimated from effort needed for design and review activities in three phases: investigation of the effect of the number of field defects on effort, construction of a regression model for the number of field defects, and estimation of testing effort to assure field quality [16]. Fig. 10 shows a schema for defect injection and defect removal. Such a schema shows the defect evolution according to each phase, but the interpretation is difficult because the syntax is not present. The main concepts proposed in the method and their relationships are not present. The representation fails to clarify the estimation of the testing effort.

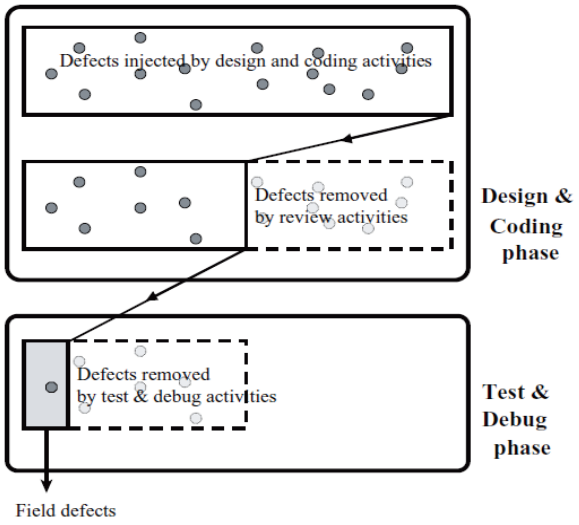

Figure 10. Defects injected and removed process. Source: [16]

4. Knowledge representation of the process of estimation effort

A state-of-the-art review of graphical representations of the testing effort estimation process was presented in Section 3. Based on this review, we can determine the different methods for estimating test effort and introduce several concepts affecting the measure. Several concepts introduced for the test effort process increases complexity of the representation of such concepts. Additionally, the scope of the graphical representations presented is broad with low specificity or so specific so that it is not possible to determine the relationship between the test effort process with actors, artifacts or any concept involved.

In this Section, we integrate the concepts of the effort estimation methods previously studied. The representation is created by using a pre-conceptual schema. We aim to express the concepts in a formal way but maintaining proximity with natural language.

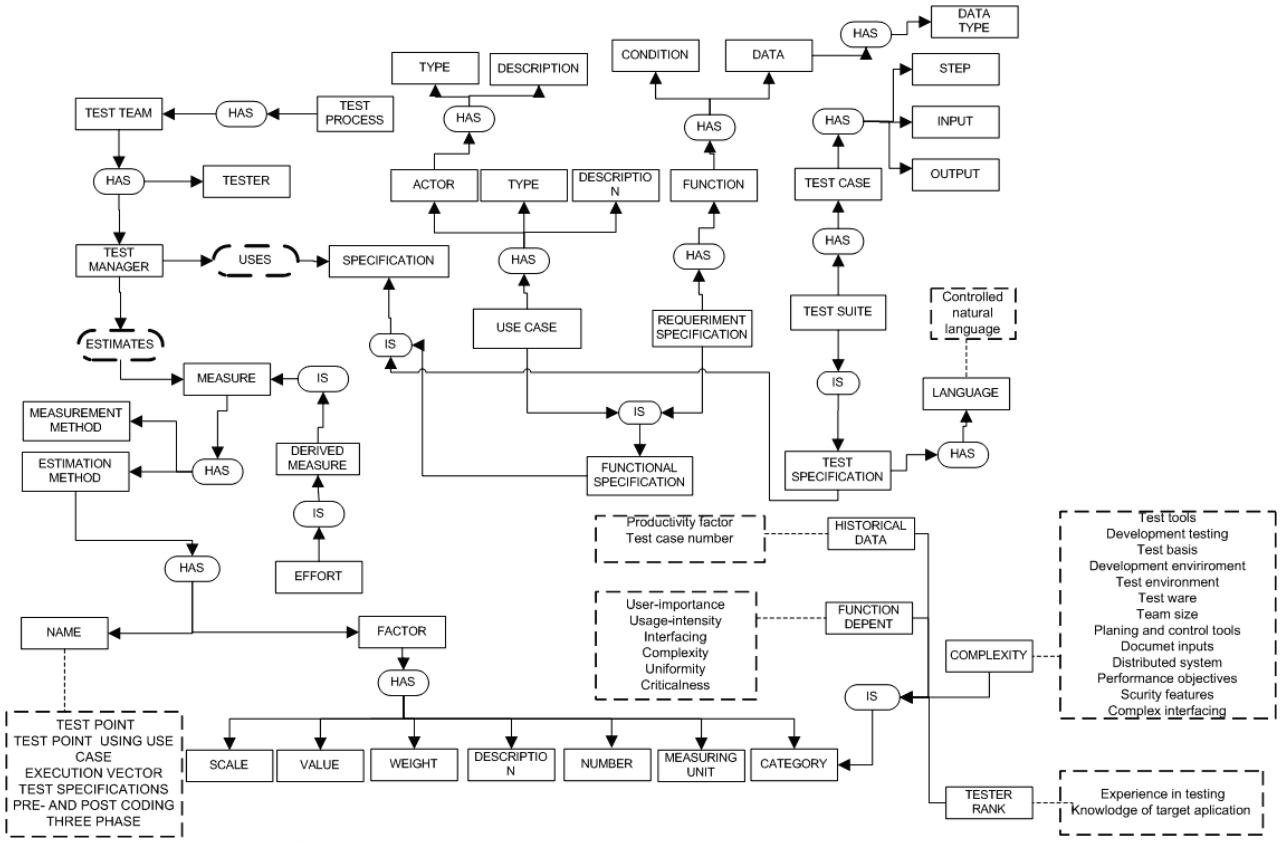

Figure 11. Knowledge representation about testing effort in pre-conceptual schemas. Source: the authors.

In the knowledge related to the test effort process estimation, we identify the following elements from the state of the art:

- The effort estimation methods are based on some specification: use case, requirements specifications, or test specification.
- The factors represent the attributes of the concept-related specifications. Each factor is associated with a direct metric.
- Some methods can use the same factor of the same specification, but the way to measure it makes differences between methods. For example, test execution complexity is a factor in test-specification-based estimation; such a factor represents the difficulty of interaction between the tester and the tested product [17]. Test execution is a factor in test-

suite-execution-vector-based estimation also, but in this approach it is inherent for a test suite in a specific environment and time [14].

- Some methods can use the factors with a different name, but the direct metric to value is the same one. For example, technical complexity factor is a list of the technical and environmental factors in use case based estimation [13]. Complexity factor is a list of the factors which can impact the time for executing a test [14]. Finally, environmental factor is a list of variables that indicate the degree to which the environment influences the text activities [12].
- Test roles have different names. We decide on using two roles: test manager and tester.
- Technical complexity factor, environmental factor, and complexity factor seems to be the same concept, because they contain a similar list of attributes. We select the second one as the name of the concept.
- There is no uniform treatment of some concepts involved in testing: test, test case, test suite, test specification and test plan are used with the same purpose.

We propose the pre-conceptual schema of the Fig. 11 as a representation of the domain about the test effort process estimation.

5. Conclusions and future work

In this paper we proposed a pre-conceptual schema related to the most representative concepts about the test effort and its relationships. This representation helps to clarify the test effort estimation process because it summarizes the common elements from the methods and presents them in a general way. Such a representation form allows understanding the basis of the test effort estimation, also identifying the main factors and its attributes.

The pre-conceptual schema can be used by either software managers or test managers to evaluate the test effort and to design testing effort estimation methods.

As future work, we propose the extension of the schema to account for most of the concepts in order to provide a complete schema for test effort estimation. Also, we propose the construction of an automated pre-conceptual schema as a tool for making decisions about software testing.

References

[1] TEC Technology Evaluation Centers [en línea], Revisión sistemática, Montreal, Canada [consulta, 24 de Marzo de 2013]. Available at: http://whitepapers.technologyevaluation.com.

[2] Aranha, E. and Borba, P., An Estimation Model for test execution effort, Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, 2007, pp. 107-116.

[3] De Almeida, E., De Abreu, B. and Moraes, R., An alternative approach to test effort estimation base on use cases, Proceedings of the International Conference on Software Testing Verification and Validation, 2009, pp. 279-288.

[4] Zapata, C. M. and Arango, F., Un ambiente para la obtención automática de diagramas UML a partir de un lenguaje controlado. DYNA, 74(153), pp. 223-236, 2007.

[5] Tian, J., Software quality engineering: Testing, Quality Assurance, and Quantifiable Improvement. Los Alamitos: Wiley-IEEE Press, 2005.

[6] Farooq, S. U., Quadri, M. K., and Ahmad, N., Software measurement and metrics role in effective software testing, International Journal of Engineering Science and Technology, 3, pp. 671-680, 2011.

[7] IEEE Std 1061-1998, Standard for a Software Quality Metrics Methodology, IEEE, 1998.

[8] Jones, C., Applied software measurement: Global analysis of productivity and quality, New York: Mc Graw Hill Professional, 2008.

[9] Nirpal, P. B. and Kale, K.V., A brief overview of software testing metrics. International Journal on Computer Science and Engineering, 3, pp. 204-211, 2011.

[10] Lakemeyer, G. and Nebel, B., Foundations of knowledge representation and reasoning. Berlin: Springer-Verlag, 1994.

[11] Zapata, C. M., Ocampo, C. A. and Giraldo, G. L., Representación del conocimiento en currículo mediante esquemas preconceptuales, Pedagogía y Saberes, 31, pp. 78-88, 2009.

[12] Veenendaal, E. V., Dekkers, T., Test point analysis: A Method for test estimation, en Kusters, R. J., Cowderoy, A., Heemstra, F. J., and Veenendaal, E., Project Control for Software Quality, Maastricht, Shaker Publishing BV, 1999, pp.45-59.

[13] Nageswaran, S., Test effort estimation using use case points, Proceedings of the Quality Week, pp. 1-6, 2001.

[14] Xiaochum, Z., Li, H., Junbo, C., and Lu, C., An experience-based approach for test execution effort estimation, Proceedings of The 9th International Conference for Young Computer Scientist, 2008, pp. 1193-1198.

[15] Abhishek, C., Kumar, V.P., Vitta, H., and Srivastava, P.R., Test effort estimation using neuronal network, Journal of Software Engineering and Applications, 3, pp. 331-340, 2010.

[16] Mizuno, O., Shigematsu, E., Takagi, Y. and Kikuno, T., On estimating testing effort needed to assure field quality in software development, Proceedings of the 13th International Symposium on Software Reliability Engineering, 2002, pp. 139-146.

[17] Aranha, E. and Borba, P., Estimating manual test execution effort and capacity based on execution points, International Journal of Computers and Applications, 31, pp. 167-172, 2009.

C. M. Zapata-Jaramillo, currently he is Associate Professor in the Computer and Desicion Sciences Department, at the Faculty of Mines, at Universidad Nacional de Colombia, Medellín Campus. He holds a degree in Civil Engineering, a Specialization in Information System Management, a MSc. in Systems Engineering, and a PhD. in Engineering (focused on Information Systems); all his titles are from the Universidad Nacional de Colombia. His research areas include: Software Engineering, Natural Language Processing, Computational Linguistics, and Pedagogical strategies for teaching Engineering.
ORCID: http://orcid.org/0000-0002-0628-4097

D. M. Torres-Ricaurte, received the Bs. Eng in Systems from Universidad del Valle, and MSc. degree in Systems Engineering from Universidad Nacional de Colombia, Medellín Campus. Her research areas include: software quality and software testing.