Software Quality Observation Model based on the ISO/IEC 29110 for Very Small Software Development Entities

Alexander Redondo-Acuña and Beatriz Florian-Gaviria
Escuela de Ingeniería de Sistemas y Computación, GEDI Research Group, Universidad del Valle,

Keywords: ISO/IEC 29110, Software Development Process, Observation Model, Indicators, Software Metrics, Software Quality Management Models.

Abstract: An imbalance exists between quality of software development for researchers on the one hand, and productivity for software industry on the other hand. However, clients demand to have both. So, this is a gap between researchers and the software industry. Therefore, it is necessary to attune software quality research to the productivity. Also it is necessary that software industry can understand the benefit of incorporating quality practices bonded to productivity. This paper proposes an observation model that allows to model internal practices of a small software development organization in comparison to those described in the ISO/IEC 29110 standard. It consists of four main components. First a visual frame of three axes: 1) the process domains and subdomains based on the profile process; 2) Roles and 3) Maturity level. Second, a battery of indicators on this three-dimensional visual frame. Third, a series of surveys designed for primary data collection from employees performing roles of the model in Very Small Entities (VSEs), and fourth, results of surveys allow disclosing values to compute metrics of indicators.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

The practical impact of software engineering (SE) research is important for the software industry in the long run. Nevertheless, there is still a wide gap between SE researchers and the software industry. For instance, while SE research is more concerned about software quality, the software industry is more devoted to productivity (Ivanov et al., 2017).

The software industry is dealing with practical ways to understand the relation between software quality standards, the enhance of its software development process, and its relation with productivity and export. In general, the software industry needs to keep looking for new niches, customers and export their products or services. Although, in many countries it's necessary to certify the software quality process according to some standard. The ISO/IEC 29110 (ISO, 2016) is a standard created to improve the competitiveness and global inclusion of companies dedicated to software development, especially Very Small Entities (VSEs), organizations with 25 employees or less (Laporte and O'Connor, 2016). This standard defines the set of practices and minimum documentation for VSEs in the software development life cycle (Munoz et al., 2018). VSEs are called to undertake ventures with the purpose of reactivating the economy and building more opportunities of employment (Merchán and Urrea, 2007). One way of doing so is that VSEs being able to export, so, VSEs needs to implement ISO / IEC 29110 in the first step.

VSEs face some problems such as shortcomings in the application of good engineering practices that allow domestic producers to achieve better ratings against widely recognized quality models (Toro Lazo, 2013). Due to their characteristics, and limitations, these VSEs have difficulty in applying methodologies, techniques and best practices to improve the quality of their products and their productivity. Thus, usually this reflected in improvisation, lack of planning, inadequate sizing, overestimated benefits. The costs and times underestimated in their projects, become large items, which at the time of implementation translate into financial overruns or cancellation (Campo Amaya, 2008).

VSEs require studies and tools that produce detailed and accurate information on all significant variables to improve the competitiveness of this industry at the regional, national and international levels (Merchán and Urrea, 2007). In the case of Colombia, Software industry studies are made by FEDESOFT. However, these studies are more concerned with big

Redondo-Acuña, A. and Florian-Gaviria, B.
Software Quality Observation Model based on the ISO/IEC 29110 for Very Small Software Development Entities.
DOI: 10.5220/0009418504460453
In *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2020)*, pages 446-453
ISBN: 978-989-758-421-3
Copyright © 2020 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

software companies and not VSEs. VSEs in Colombia are of interest due to they are 80% of the software industry.

The ISO/IEC 29110 standard is a valuable contribution to the construction of a quality software development process for VSEs. However, VESs software development companies are confused at the process of understanding and obtaining the ISO/IEC 29110 international standard certification.

The paper (Abuchar Porras et al., 2012) concludes that knowledge is needed in the use of software development quality standards in VSEs, and to train employees the standard they implement. Also, recommend developing a measurement instrument for models, methodologies, software development life cycle and quality standards in software development processes.

These needs raise the central question of this research: how to observe the quality of software development processes in VSEs?. The objective of this paper is to give them a method to measure their quality status in the software development process to VSEs; addressed by the ISO/IEC 29110 Basic Profile and not to focus on the product quality of the software described in ISO/IEC 25010. The ISO/IEC 29110 Basic Profile includes very few references to product quality characteristics as described in ISO/IEC 25010 (García-Mireles, 2016). In the systematic review published in (Munoz et al., 2018). It has been shown that have been developed more frameworks than tools for quality management in software development processes. An example of this is (Krouska et al., 2019).

All in all, this paper proposes an observation model that allows to model internal practices of a small software development organization in comparison to those described in the ISO/IEC 29110 standard. This work, funded by the Regional Government, it is intended to be tested for the Valle del Cauca region in Colombia. But, this observation model could be applied globally. The proposed observation model allows us to make visible the activities and guides that the ISO/IEC 29110 standard demands for each maturity level, and disclose which of them are performed by a VSE. VSEs would be aware of the utility of the standard, its guides, and the implementation of standard activities in software development to improve the software quality process.

This paper is structured as follows. Section 2 describes the proposed model to measure software quality development based on ISO/IEC 29110. Section 3 presents the software to gather data, to compute indicators, and to produce data visualization. Section 4 presents the battery of indicators. Section 5 provides the validation of the battery indicators. Section 6 concludes the paper.

2 PROPOSED MODEL

The evaluation of administration and development of software projects in VSEs requires the observation and measurement of performed processes and tasks. Within the study of the software life cycle, the processes and tasks of development, we can find a series of concepts, actors, and goals or objectives that are commonly handled by the models, standards, and development standards. The ISO/IEC 29110 standard is devoted to defining processes and tasks for the VSEs in software development. In this standard deployment packages (DP) (ISO/IEC, 2011) are a set of artifacts developed to facilitate the implementation of a set of practices but, a deployment package is not a complete process reference model. Moreover, this standard also defines the roles involved in each process. The standard defines four levels of maturity that VSEs can certificate depending on their number of DP and the number of roles performed.

To observe and evaluate the quality of processes and tasks performed by different VSEs, the ISO/IEC 29110 is a good conceptual framework to lead it, but in this standard it is not clear how to measure the quality of each task. It is necessary to produce a way to measure or assess the quality of each task and process. The set of measures can determine a global assessment of the quality of the VSEs, and to establish how far are these VSEs to achieve a certification that allows it to export software. Finally, visual representation to understand the evaluation points could allow VSEs to be more aware of their overall quality condition. The observation model of three axes was created to be able to visualize the assessment of software development tasks performed by a VSE and to be able to observe the general level of maturity of the VSE. The sections below explain in detail the three axes of the proposed observation model and its relationship with the ISO/IEC 29110 standard.

2.1 Process Domains and Subdomains

Process domains don't need to have a sequence in their construction, they do not have process dependencies. Therefore, process domains can be built or not depending on the needs or the profile of the VSE. Existing nine process domains in the observation model focuses on the Software Implementation process. Thus, process domains in the proposed model are: 1) Requirements Analysis (RA), 2) Architecture and Detailed design (DA), 3) Construction and

Unit Testing (CT), 4) Integration and Testing (IT), 5) Verification and Validation (VV), 6) Version Control (VC), 7) Self-Assessment (SA), 8) Product Delivery (PD). 9) Project Manager (PMD).

Each process domain has subdomains that are performed by one or more roles. In observation model, subdomains must be developed within each process domain. As the subdomains can be met in different ways, it can not be a simple checklist. It is necessary to create indicators to measure since companies have different practices to do subdomains. All subdomains of the SI process will list next.

Requirements Analysis (RA)

1. Review the project plan with the work team(WT).
2. Elicit acquirer and other stakeholders' requirements and analyze system context.
3. Review stakeholders' requirements specifications with the project manager(PMR).
4. Baseline stakeholders' requirements specification.
5. Capture system requirements and interfaces.
6. Capture system element(s) and interface requirements.
7. Verify and obtain WT agreement on the system requirements specification.
8. Validate that system requirements specification satisfies stakeholders' requirements specification.
9. Define or update traceability between requirements.

Architecture and Detailed Design (DA)

1. WT review of the project plan to determine task assignment.
2. Design the system functional architecture and associated interfaces, allocation of the functional to the physical architecture.
3. WT review of the system requirements specifications.
4. Functional and physical design verified and defects corrected.
5. Verified IVV plan (integration, verification, validation, qualification) and verification procedures.
6. Traceability between the functional architecture definition and the System Requirements and between the physical architecture definition, the system elements and the functional architecture definition.
7. Design products placed under configuration management.

Construction and Unit Testing(CT)

1. WT review of the project plan to determine task assignment.
2. Work team review of the physical design.
3. Hardware System Elements to be developed and tested.
4. Software system elements to be developed and tested.
5. Traceability between hardware construction, software construction and physical architecture.

Integration and Testing (IT)

1. Understand test cases and test procedures.
2. Set or update the testing environment.
3. Integrates the software using software components and defines or updates test cases and test procedures for integration.
4. Perform tests using test cases and test procedures for integration and document results in a test report.

Verification and validation (VV)

1. Verification of design, test cases and test procedures.
2. Verify software construction.
3. Software test for integration.
4. Verification of maintenance documentation.

Version Control (VC)

1. Planning and setting up repository
2. Version identification
3. Change control

Self-Assessment (SA)

1. Analysis of process coverage.
2. Identification of improvement opportunities.
3. Walk-through before performing a process.
4. Walk-through after performing a process.

Product Delivery (PD)

1. Assign tasks to the work team members related to their role, according to the current project plan.
2. Understand software configuration.
3. Document the maintenance documentation or update the current one.
4. Verification of the maintenance documentation.
5. Incorporate the maintenance documentation as a baseline for the software configuration.
6. Perform delivery according to delivery instructions.

Project Manager (PMD)

1. Project planning process.
2. Project plan execution.
3. Project assessment and control process.
4. Project closure.

2.2 Roles

A defined function that a member of the project team must perform (noa, 2017). This is the alphabetical list of the roles in the proposed model: Analyst (AN), customer, designer (DES), the programmer (PR), project manager (PMR), technical leader (TL), work team (WT). Table 2 is an alphabetical list of the roles, abbreviations and description of required competencies.

Table 1: Role Description. Based on ISO/IEC 29110.

Role	Competencies
Analyst (AN)	Knowledge and experience eliciting, specifying and analyzing the requirements. Knowledge in designing user interfaces. Knowledge of the revision, editing techniques and experience on software development and maintenance.
Designer (DES)	Knowledge and experience in the software components and architecture design. Knowledge and experience in the planning and performance of integration and system tests. Knowledge of the revision, editing techniques and experience on software development and maintenance.
Programmer (PR)	Knowledge and/or experience in programming, integration and unit tests. Knowledge of the revision, editing techniques and experience on software development and maintenance.
Project manager (PMR)	Leadership capability with experience making decisions, planning, personnel management, delegation and supervision, finances and software development.
Technical leader (TL)	Knowledge and experience in software development and maintenance.
Work team (WT)	Knowledge and experience according to their roles on the project: AN, DES, and/or PR.

2.3 Maturity Levels of VSEs

Maturity levels are the stages of maturation of processes that appear in the progressive path towards growth and stability as independent and competitive software development. In the proposal model, maturity levels are in agreement with some profile of the ISO/IEC 29110 standard: Entry level (EL) is targeted to VSEs working on small projects and for start-up VSEs. Basic level (BL) describes the development practices of a single application by a single project team. Intermediate Level (IL) is targeted at VSEs developing multiple projects with more than one team. Advanced Level (AL) is targeted at VSEs wishing to sustain and grow as independent competitive businesses. This set of four levels, providing a progressive approach to satisfying the requirements of a generic profile group, is based on (O'Connor and Laporte, 2017).

Figure 1 shows the data visualization of an example of a VSE assessment using this observation model. Indicators of the observation model are calculated and placed according to the axes of the model.

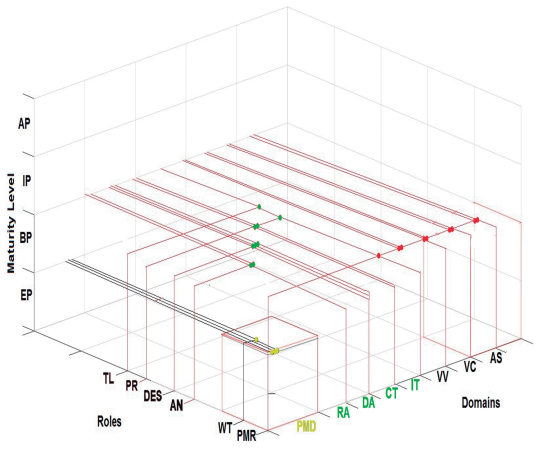


Figure 1: Visualization of a VSE assessment.
Source: The authors

3 WEB APPLICATION

A web software application is under construction to primary source data collection, to compute indicators, and to produce data visualizations. It has a first functional module of surveys. Surveys must be applied to people performing each one of the roles of software development within the VSEs. The second part of the web application is the one that computes the indicators, produces their interpretations and appropriate visualizations.

To gather solid and valid data, that can be analyzed uniformly and coherently, an adequate instrument is needed to standardize the process. The instruments used are surveys for each role. The phases or steps in the data gathering recommended by Ruiz (Bolívar, 2002) were used. Surveys will serve to collect inputs of indicators. Each indicator will need one or more questions that capture the needed values to compute the metric of the indicator. Surveys are delivered by roles, in total there are six surveys constructed, questions can be updated. Figure 2 shows the preliminary design for the analyst role survey.

The presentation of results (indicators and other reports) in the user interface uses elements such as tables, graphs, files and info-graphics. “Dashboards and visualization are cognitive tools that improve your ‘span of control’ over a lot of business data. These tools help people visually identify trends, patterns and anomalies, reason about what they see and help guide them toward effective decisions. As such, these tools need to leverage people's visual capabilities.” (Brath and Peters, 2004)

Figure 3 shows the example of visualization for the indicator ”advance of the execution” in the requirements analysis domain.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

Table 2: Characteristics of the 22 proposed indicators.

	ID	Name	Metric	unaccepted	accept	optimal

01	RA-1-AN-PB	Advance of execution of tasks of analysis of requirements	A = Tasks total;
			B = Tasks already completed;
			PA = % of advances planned tasks
			PA = (B/A) *100
				<70%	>70%	>90%

02	RA-2-AN-PB	Annual average of advances of analysis of requirements	PA_i = % de avance RA by project; $n_{proyectos}$ = # of project by year; PAA = $\sum PA_i$ / $n_{proyectos}$	<70%	>70%	>90%

03	DA-1-DES-PB	Number of information security incidents	NIS = # totaltotal incidents SI ;
			NIA = # failed access attempts ;
			NIC = # incidents unauthorized access ;
			NID = # destruction or loss of data.
			NIS = NIA + NIC + NID
				>100	<50	<30

04	DA-2-DES-PB	Average of operating systems	Xi = # operating systems (OS) to support by projects;
			A = # annual projects;
			X = average OS;
			X = $\sum X_i$ / A
				0	1	>1

05	DA-3-DES-PB	Average number of different programming languages implemented in the project	A = # functional requirements;
			Xi = programming language to implement the requirement;
			PTI = $\sum X_i$ / A
				>100	<50	<30

06	CT-1-PR-PB	Test coverage	A = # of test cases of the testing plan; B = # of test cases performed; $TC = (A/B)$	<0.7	>0.7	>0.8

07	CT-2-PR-PB	Successful corrections	A = # corrections did not resolve bugs;
			B = # corrections resolve bugs, but inject new defect;
			C = # faulty corrections;
			$C = A + B$;
			D = # total corrections made;
			E = # total successful corrections;
			$E = D - C$;
			$PCE = (E/D) * 100$
				<70%	>70%	>80%

08	IT-1-PR-PB	% of corrected incidents	B = # of incidents found;
			A = # incidents to correct;
			$X = (A/B) * 100$
				<70%	>70%	>90%

09	IT-2-TL-PB	Response time	A = time Design specifications; B = time to test the complete route of a transition; C = time to test complete product modules; $X = A + B + C(h) hour$	>24h	<16h	<8h
10	IT-3-PMR-PB	Sufficiency of the tests	A = # of test cases in the testing plan; B = # of required test cases; $X = A/B$	<70%	>70%	>90%
11	VV-1-PMR-PB	Obvious functions	A = # of obvious functions to the user. B = # total functions. $X = A/B$	<0.7	>0.7	>0.8

12	VV-2-PMR-PB	Change Registrant	A = # confirmed changes;
			B = # total changes modified;
			$X = A/B$
				<0.7	>0.7	>0.9

13	VC-1-PMR-PB	Percentage of artifacts for version control created by project	PA = # total of planned artifacts;
			CA = # created artifacts;
			$PCA = (AR/AP) * 100$
				<0.7	>0.7	>0.9

14	VC-2-PMR-PB	Average number of possible artifacts created in the year	NP = # projects; PCA_i = % artifacts by projects; NPA = average artifacts created by year; NPA = $\sum PCA_i$/NP	<70%	>70%	>90%
15	SA-1-PMR-PB	Functionality coverage	C = Functionality coverage; A = # of missing functionalities detected in the evaluation; B = # of functionalities established in the specification; $C = 1 - A/B$	<0.7	>0.7	>0.9
16	SA-2-PMR-PB	Average number of possible artifacts created in the year	NP = # number of projects; NPA_i = number of possible artifacts created by project; NPA = ($\sum NPA_i$)/NP	<0.7	>0.7	>0.9
17	PD-1-PMR-PB	Sum of projects accepted by the client during the year	n = # of project by year; PA = # Projects accepted; $PA = PA/n$	<0.7	>0.7	>0.9
18	PD-2-PMR-PB	Ease of software installation	FI = Flexibility and customization of the software installation capacity; A = # of installation operations implemented; B = # of installation operations required; $FI = A/B$	<0.7	>0.7	>0.9

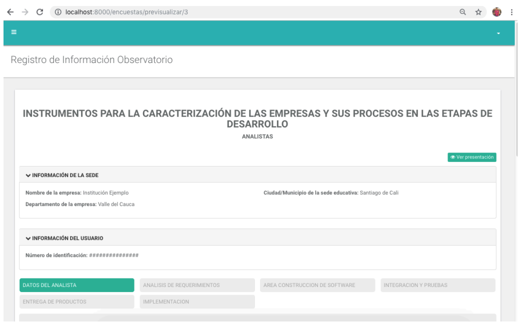
19	PMD-1-PMR-PE	% compliance with established quality characteristics	A = # quality requirements implemented;
			B = # total quality requirements established;
			$X = (A/B) * 100$
				<70%	>70%	>90%

20	PMD-2-PMR-PE	% of time required to complete project	A = Time used to carry out the project; B = Planned time for the realization of the project; $X = (A/B) * 100$	<70%	>70%	>90%

21	PMD-3-WT-PE	Advance of execution of planned activities	A = # of complete tasks;
			B = # of proposed task;
			$X = (A/B) * 100$
				<70%	>70%	>90%

22	PMD-4-WT-PE	Difference between planned and actual tasks	A = # task planned;
			B = # actual task does;
			$X = A - B$
				>A/2	<A/2	0

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS


localhost:8000/encuestas/previsualizar/3

Registro de Información Observatorio

INSTRUMENTOS PARA LA CARACTERIZACIÓN DE LAS EMPRESAS Y SUS PROCESOS EN LAS ETAPAS DE DESARROLLO

ANALISTAS

Ver presentación

INFORMACIÓN DE LA SEDE

Nombre de la empresa: Institución Ejemplo

Ciudad/Municipio de la sede educativa: Santiago de Cali

Departamento de la empresa: Valle del Cauca

INFORMACIÓN DEL USUARIO

Número de identificación: ##############

DATOS DEL ANALISTA | ANALISIS DE REQUERIMIENTOS | AREA CONSTRUCCION DE SOFTWARE | INTEGRACION Y PRUEBAS | ENTREGA DE PRODUCTOS | IMPLEMENTACION

Figure 2: Survey header in the prototype software made for the role: Analyst.

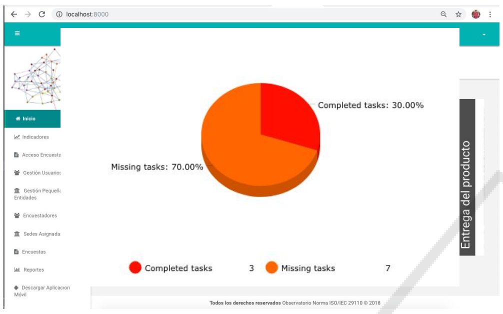


Figure 3: Example of data visualization for an indicator.

4 BATTERY OF INDICATORS

An indicator is an instrument that provides quantitative evidence about whether a completed condition exists or whether a certain result has been achieved or not. In the case of not achieving the result allows to evaluate the progress made. In this case, compliance with the standard must be measured about management and implementation activities made by the VSE, concerning the software life cycle.

At the time of the conceptual analysis, 22 indicators were identified based on the needs of the government sector from which 6 indicators emerged: (MinTic, 2015), 10 indicators of the ISO standards (12207, 29110)(Singh, 1996) (ISO/IEC, 2015) and academic concepts such as (Kan, 2002) were built into 6 indicators.

To select the appropriate indicators, the following strategy was used. 1. Seek indicators used by government, academy, and private industry. 2. Select from that set of indicators those that could be calculated using variables collected by data from primary harvest (surveys).

After applying this strategy, the resulting indicators were arranged in the observation cube. Ergo the proper domain process, subdomain, role, and maturity level where selected for each indicator in the set. Thereby, indicators are located at some point within the observation cube see Figure 1. An indicator position depends on the process domain/sub-domain, the role, and the maturity level of the observed company. It is important to notice that, although many indicators were discarded for this study, because it was not feasible to calculate them using surveys to employees, other indicators can be added to the three-dimensional observation model if another source of calculation were possible in the future.

In this first version, the battery of indicators is more devoted to the project manager role, because it is the role with more interaction throughout the software development, although there are a few indicators for other roles. Also, in this first version of the battery of indicators, the maturity level entry profile and basic profile were preferred, because they are currently the profiles for which the standard defines guidelines and documentation.

Figure 4 portrays the methodology used to construct and validate the model and its indicators in eleven steps.

It is important to describe the detailed information that describes the indicators and it is contained in the indicator's sheet. Characteristics described in these sheets are: Acronym identifier, name, description of use, how often it is needed to measure it, metric, input-data for computing, information about how to interpret it as well as not to misinterpret it, type of measure (numerical, percentage, etc.), process domain, subdomain (task), role (person who performs the task), maturity level (entry or basic), link to the reference.

Due to this extension, it is not possible to show completely the 22 sheets of indicators in this paper. However, the authors present here a simplification of sheets in Table 2. This table presents some characteristics of the 22 indicators. From the original sheets were filtered 6 columns. Column (ID) is the indicator identifier. It consists of an acronym in which the first two letters are the initials of the domain, followed by a consecutive number, followed by the initials of the role and finally the level of maturity. Column (Name), Column (Metric) is the identifier of the metric applied to compute the indicator. The 22 metrics of these indicators are detailed in Table 2. To interpret the result of an indicator's metric, three levels were defined for all indicators, namely: column unacceptable, column acceptable, and column optimal. These levels bring an interpretation more useful than just a Boolean evaluation that qualifies if the enterprise complies with the task or not. In Figure 1 the color point represents the assessment of indicators. The three levels of evaluation are represented with three colors namely: red for unacceptable level, yellow for an acceptable level, and green for optimal level.

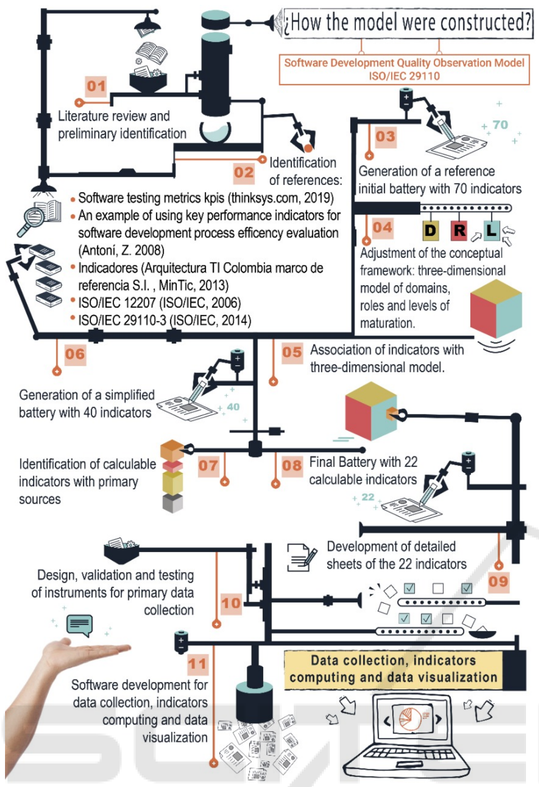

¿How the model were constructed?
Software Development Quality Observation Model ISO/IEC 29110
01
Literature review and preliminary identification
02
Identification of references:
• Software testing metrics kpis (thinksys.com, 2019)
• An example of using key performance indicators for software development process efficency evaluation (Antoní, Z. 2008)
• Indicadores (Arquitectura TI Colombia marco de referencia S.I. , MinTic, 2013)
• ISO/IEC 12207 (ISO/IEC, 2006)
• ISO/IEC 29110-3 (ISO/IEC, 2014)
03
Generation of a reference initial battery with 70 indicators
70
04
D
R
L
Adjustment of the conceptual framework: three-dimensional model of domains, roles and levels of maturation.
05
Association of indicators with three-dimensional model.
06
Generation of a simplified battery with 40 indicators
40
07
Identification of calculable indicators with primary sources
08
Final Battery with 22 calculable indicators
22
09
Development of detailed sheets of the 22 indicators
10
Design, validation and testing of instruments for primary data collection
11
Software development for data collection, indicators computing and data visualization
Data collection, indicators computing and data visualization

Figure 4: Construction methodology of model and its indicators.

5 INDICATORS' VALIDATION

To validate the battery of indicators, a spreadsheet was created to allow VSEs to assess whether indicators met or not 12 quality criteria. The evaluation criteria were: 1) Is the indicator easy to understand?, 2) Is the indicator of interest for VSEs?, 3) Is adequate the time in which the indicator should be collected?, 4) Is the indicator unambiguous?, 5) Can the indicator be validated?, 6) Can the indicator observation be trustworthy?, 7) Is the metric to compute the indicator correct?, 8) Can results of computing be interpreted correctly?, 9) Is the indicator well classified on its domain?, 10) Is the indicator well classified in its role?, 11) Is the indicator well classified on its maturity level?, 12) Does the indicator express what it wanted to measure? The criteria for the election were based on (Kerzner, 2013). VSEs evaluate these criteria and allow detecting errors in the proposed indicators. If the VSEs considered a quality criterion not met, a comment of explanation or justification was demanded.

For this validation, at least 15 VSEs were invited. Only 3 of them accept to be part of the validation study: CEO Orlando Rincón, on behalf of PARQUESOFT (VSEs of software development cluster), GreenHorizon, and RADY. The three VSEs validated the quality criteria for the battery of indicators. The validation results showed three main errors. First, "missing variables in 3 metrics" (Criterion 7). Second, "the description of what the metric in 3 indicators is short" (Criterion 12 and Criterion 1), and "for 15 indicators the columns of levels of interpretation are not filled" (Criterion 8). All 21 errors where fixed to produce the last version of the battery of indicators. In detail, all metrics were looked over by comparing with reference metrics. New variables were added for some indicators. All interpretations of metrics were looked over and it was changed to those that had not the three levels of interpretation (optimum, acceptable, unacceptable).

6 CONCLUSIONS

This paper proposes an observation model that allows to model the internal practices of an organization and compare those with those described in the ISO/IEC 29110 standard. The artifact visualizes the evaluation of quality tasks expected to be performed by VSEs in order to obtain the ISO/IEC 29110 standard certification. This work vision is to contribute to attune software quality research with necessities and worries of the software industry. The observation model is a three-dimensional model having 22 indicators to assess quality tasks for VSEs. The assessment depends on a series of surveys that people with specific roles in the entity fill in. Indicators are computed with data collected from surveys. The assessment interpretation for each indicator has three levels (optimum, acceptable, unacceptable). A three-dimensional visualization shows the assessment of the set of indicators for a VSE. The levels of interpretation are represented in colors (green, yellow and red) respectively. Besides, each indicator has a particular visualization and a complete sheet of characteristics. A software tool is being developed to promote this observation model for VSEs on their way to ISO/IEC 29110 quality certification. Globally, this software tool and its observation model would contribute to have studies and reports on the real tasks and practices performed by VSEs in the software development process. Moreover, it would encourage the integration of VSEs of software development to strengthen collaboration and research. Besides, it would allow the training of trained persons. An empirical validation of the model, including its battery of indicators, seems to show the complacency of VSEs of software development with the proposed model of assessment. More data are needed in order to have a quantitative validation of the model. But it is very difficult to involve enterprises in research steps. The gap we mentioned early is in both sides, from researchers to the software industry, and the other way round. This observation model is not exclusive for Colombia, it could be applied globally by any VSEs of software development to interpret its level of agreement with the ISO/IEC 29110 standard.

ACKNOWLEDGMENT

The authors want to acknowledge to Gobernación del Valle del Cauca, INFIVALLE, PACIFITIC, and UNIVERSIDAD DEL VALLE for the master thesis scholarship granted to Alexander Redondo Acuña in the frame of Agreement CI-115-2016. Also, thanks to VSEs for their helping in model validation (Orlando Rincón (CEO PARQUESOFT), GreenHorizon, RADY). The authors are grateful to CIER-SUR and the GEDI research group for sharing their preliminary research on the education innovation observatory project.

REFERENCES

(2016). Iso/iec tr 29110-1, systems and software engineering — lifecycle profiles for very small entities (vses) — part 1: Overview [technical report].

(2017). *ISO/IEC/IEEE 24765: 2017(E): ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary*. IEEE.

Abuchar Porras, A., Cárdenas Quintero, B., and López, D. A. (2012). Observatorio de prácticas de desarrollo de software en minpyme y pymes de bogotá. *Ciencia e Ingeniería*, pages 114–130.

Bolívar, C. (2002). Instrumentos de investigación educativa. procedimiento para su diseño y evaluación. *CIDEC. Capanegra, H (2002), El Gobierno electrónico hacia una verdadera reforma del estado.[Documento en línea] Disponible en: http://www. clad. org. ve/fulltext/0043101. pdf [con acceso el 20-09-2004]*.

Brath, R. and Peters, M. (2004). Dashboard design: Why design is important. *DM Direct*, 85:1011285–1.

Campo Amaya, L. F. (2008). Modelos de capacidad y madurez y la industria del software en colombia. *Generación Digital*, 7:22–25.

García-Mireles, G. A. (2016). Addressing product quality characteristics using the iso/iec 29110. In *Trends and Applications in Software Engineering*, pages 25–34. Springer.

ISO/IEC (2011). *Systems and software engineering — Lifecycle profiles for Very Small Entities (VSEs) — Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile*. Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 7, Software and systems engineering., Switzerland.

ISO/IEC (2015). *Systems and software engineering — Lifecycle profiles for Very Small Entities (VSEs) — Part 3-1: Assessment guide*. ISO/IEC JTC 1/SC 7 Software and systems engineering, Las Vegas, US.

Ivanov, V., Rogers, A., Succi, G., Yi, J., and Zorin, V. (2017). What do software engineers care about? gaps between research and practice.

Kan, S. H. (2002). *Metrics and models in software quality engineering*. Addison-Wesley Longman Publishing Co., Inc.

Kerzner, H. (2013). *Project Management Metrics, KPIs, and Dashboards: A Guide to Measuring and Monitoring Project Performance*.

Krouska, A., Troussas, C., and Virvou, M. (2019). A literature review of social networking-based learning systems using a novel iso-based framework. *Intelligent Decision Technologies*, 13(1):23–39.

Laporte, C. Y. and O'Connor, R. V. (2016). Implementing process improvement in very small enterprises with iso/iec 29110: A multiple case study analysis. In *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 125–130.

Merchán, L. and Urrea, A. (2007). Caracterización de las empresas pertenecientes a la industria emergente de software del sur occidente colombiano caso red de parques parquesoft. *Avances en sistemas e informática*, 4:10.

MinTic (2015). Marco de referencia de arquitectura empresarial para la gestión de tecnologías de la información (ti), a adoptar en las instituciones del sector público colombiano.

Munoz, M., Mejia, J., and Ibarra, S. (2018). Tools and practices to software quality assurance: A systematic literature review.

Munoz, M., Mejia, J., and Lagunas, A. (2018). Implementation of the iso/iec 29110 standard in agile environments: A systematic literature review. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6.

O'Connor, R. V. and Laporte, C. Y. (2017). The evolution of the iso/iec 29110 set of standards and guides. *International Journal of Information Technologies and Systems Approach (IJITSA)*, 10(1):1–21.

Singh, R. (1996). International standard iso/iec 12207 software life cycle processes. *Software Process: Improvement and Practice*, 2(1):35–50.

Toro Lazo, A. (2013). Caracterización del proceso de desarrollo de software en colombia: una mirada desde las pymes productoras. *Revista Páginas de la UCP*, 92:92–98.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS