Evaluating Software Security Change Requests: A COSMIC-based Quantification Approach

Mariem Haoues, Asma Sellami
Mir@cl Laboratory
University of Sfax, ISIMS
BP 242. 3021. Sfax-Tunisia
{mariem.haoues, asma.sellami}@isims.usf.tn

Hanêne Ben-Abdallah
Higher Colleges of Technology
Dubai, UAE
hbenabdallah@hct.ac.ae

Onur Demirörs
Izmir Institute of Technology
Department of Computer Engineering
onurdemirors@iyte.edu.tr

***Abstract*—Software project scope defines functional and non-functional requirements. These requirements may change to satisfy the customers' needs. However, the control of scope creep represents one of the success keys in software project management. Changes in non-functional requirements affect the ISO/IEC 25010 quality characteristics such as security, portability, etc. Furthermore, some of these quality characteristics may evolve throughout the software life cycle into functional requirements. In this paper, we explore the use of COSMIC method - ISO/IEC 19761 to quantify and evaluate security change requests. Measuring the functional size of security change requests allows stakeholders to make appropriate decisions about whether to accept, defer, or deny the change.**

***Index Terms*—Technical change, non-functional requirements, system requirements, security, functional size, COSMIC method - ISO/IEC 19761.**

I. Introduction

Software systems are exposed to a high number of changes throughout their life-cycle. Customers often change their needs even after the approval of the software project scope, which defines the functional and non-functional requirements. Customer change requests include adding new functionality, modifying non-functional requirements, correcting bugs, etc. However, change requests frequently result in cost overruns and delays, with about 75% of changes result in software failure [1]. Consequently, the control of scope creep represents one of the success keys in software project management. This control relies on processes to analyze and evaluate the impact and significance of each change request before its approval. To manage these processes effectively, decisions should be made quickly with reasonable effort and accuracy.

Based on the fact that the success of any management activity highly depends on the accuracy of software size measurement [2], software organizations frequently use software size measurement as a technique to estimate the development effort, make decisions, etc. (*cf.,* [3], [4]). Software size can be quantified by length or functionality measures. Length measures quantify the technical size of software from the developer's perspective based on counting, for example, the number of lines of code. In contrast, functionality measures quantify the functional size of software from the user's perspective. Functional Size Measurement (FSM) methods are classified into first generation methods (*e.g.,* IFPUG [5], Nesma [6], etc.) and second generation methods such as COSMIC [7]. Compared to other FSM methods, COSMIC is designed to objectively measure the functional size of a change to software as well as the size of software that is added, changed or deleted [8]. In our previous studies (*cf.,* [9], [10]), we used COSMIC to quantify requirements change requests that affect the functional user requirements. Based mainly on the functional size of the change, we evaluated the change status in order to identify changes that may lead to significant impact on the software development progress. In this paper, we extend these studies to cover changes in non-functional requirements.

The success of the modern software organizations depends not only on the rapid innovation but also on the ability to adapt to every-changing circumstance and to increase stakeholder's satisfaction. Thus, controlling the trade-offs between time, cost and scope (*i.e.,* functional and non-functional requirements) becomes mandatory. In addition, ignoring non-functional requirements may result in major estimation errors [11]. This means that non-functional requirements and functional requirements must be addressed similarly. In fact, dealing with non-functional requirements in an improperly way leads to the projects failure, considerable delivery delays and consequently increase the final cost [12].

System Non-Functional Requirements (system-NFR) can be detailed and specified as software Functional User Requirements (software-FUR) [13]. In fact, researchers distinguished non-functional requirements that may evolve wholly or partly into functional requirements (in this paper we call them software-FUR derived from system-NFR) from True NFR which cannot be considered as software functions such as technology and project constraints. Providing a complete list of software-FUR derived from system-NFR is not possible [14]. However, through our literature review, we noted that this list covers mainly the following quality characteristics: performance, compatibility, maintainability, reliability, and portability.

Figure 1 illustrates the evolution of system requirements and their respective measurements. As shown in this figure, it is not possible to size the True NFR using FSM methods throughout the software life-cycle phases. Whereas, the functional size of software-FUR derived from system-NFR can be measured using FSM methods.

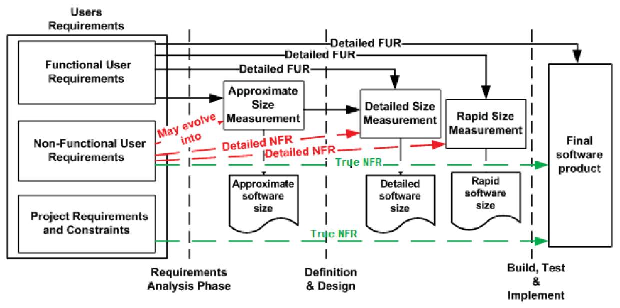

Fig. 1. COSMIC FSM method applied throughout the software life cycle

The main question that we address in this paper is: *how change requests in system-NFR, in particular security requirements, affect the software development progress?* Towards this end, we complement our COSMIC-based approach for functional change requests evaluation (*i.e.,* change requests that affect software-FUR) proposed in [10] by evaluating technical change requests (*i.e.,* change requests that affect system-NFR). We believe that the functional size of a technical change will certainly provide a more realistic evaluation of the change request, and hence a tighter estimate of the effort/resources needed to implement the change. In particular, we propose a COSMIC-based approach to quantify and evaluate the functional size of a technical change request affecting the software security, an important ISO/IEC 25010 quality characteristic [15].

The remainder of this paper is organized as follows: Section II presents background information about requirements change and the COSMIC FSM method. Section III first surveys some works studying the functional size measurement of non-functional requirements and the security requirements. Then, it briefly overviews our earlier work. Section IV describes how to use COSMIC in measuring technical change requests and illustrates the application of our approach using the case study "C-Registration System" [16] [17]. Section V discusses the results of this study and highlights several threats to its validity. Finally, section VI summarizes the presented work and outlines some of its possible extensions.

II. Background

A. Requirements Change

Research studies depict that insufficient requirements engineering causes the failure of more than 50% of software projects [18]. A correct and complete identification of the system requirements at the beginning of the software development is difficult. In addition, software is the most easily changed element in a software system [19]. Furthermore, because developers emphasize more on the functional side of the software underrating the non-functional quality characteristics [18], changes are proposed later in order to improve the software quality. This makes requirements change inevitable throughout the software life-cycle.

However, for the success of the software project, each change must be accompanied by an evaluation of its impact on the work product, development effort, etc. The main purpose of this evaluation is to determine whether the change is "in scope" or "out of scope" [19]. An "in scope" change can be accomplished with little or no disruption to the planned work activities. In contrast, an "out of scope" change incur adjustments to the budget, resources, and/or schedule, and/or modification or elimination of other product requirements.

Based on the change evaluation, decision-makers will consider the relevant information to make appropriate decisions regarding the change. These decisions are made usually based on the development team experience and they are mostly driven by business needs and time constraints [20]. However, the lack of quantitative information may jeopardize the software project success. Thus, quantitative information are needed to guide the decision-makers in deciding whether to accept, deny or defer the change.

B. COSMIC FSM Method Overview

COSMIC is increasingly being adapted in the software development industry. Software functional size measured with COSMIC is used mainly to evaluate the project performance and estimate the effort required for future projects in terms of cost and duration. On the other hand, customers use software size to control their suppliers' price/performance, the delivered product quality, etc. In addition to its advantages compared to the first FSM generation methods, COSMIC has been successfully used for approximate sizing variants for use in early/rapid project estimation [11], guidance on how to deal with non-functional requirements (*cf.,* [2], [12]), sizing functional change requests (*cf.,* [9], [10]), and making decisions on software project development (*cf.,* [21], [3]).

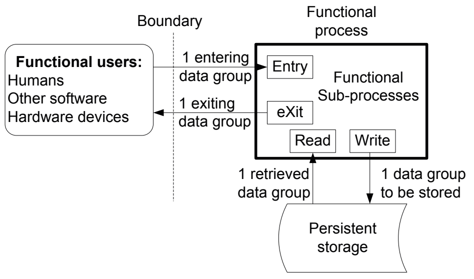

Fig. 2. COSMIC data movements [7]

When measuring the software functional size, COSMIC considers that each software implies a set of functional processes that can be triggered by triggering events via a functional user. Each functional process is detailed by a set of functional sub-processes. A sub-process can be either a data movement or a data manipulation. A data movement moves a single data group to/from the functional users or to/from the persistent storage. COSMIC identifies four types of data movements: **E**ntry, e**X**it, **W**rite, and **R**ead (see Figure 2). An Entry moves a data group from a user to the functional process.

An eXit moves a data-group from a functional process to the user. A Write moves a data-group inside the functional process to a persistent storage. And, a Read moves a data-group from persistent storage to the functional process. One COSMIC Function Point (CFP) is attributed for each data movement. The total software size is obtained by aggregating the sizes of all its functional processes.

COSMIC measures the size of a change to software and the size of software that is added, changed or deleted as well. It defines a change as *"any combination of additions of new data movements or of modifications or deletions of existing data movements"* [7]. The functional size of the change is given by the aggregation of the sizes of all the added, deleted and modified data movements.

III. Related Work

A. Measuring Non-Functional Requirements Overview

System-NFR are not easy for stakeholders to articulate but the software will not be useful without some of them [22]. However, software quality specification requires time and money. In addition, customers are always asking to improve the quality of their software. In order to reduce the impact of requirements creep, the authors in [22] proposed the quality function deployment technique that can assist developers in identifying and prioritizing customer expectations.

On the other hand, when estimating the software development effort, system-NFR should be measured since they certainly will increase the total software size [23]. As shown in Table I, several studies have recently proposed to measure the functional size of some system-NFR such as portability, security, performance, etc. Some of these studies used FSM methods such as COSMIC (*cf.,* [2], [12]), while others proposed their own methods and tools (*cf.,* [24], [25]). For instance, Kassab et al., [12] proposed to apply the COSMIC-FFP method in measuring the functional size of non-functional requirements in order to reduce the uncertainty in software estimation. They proposed a non-functional requirements size measurement method named NFSM and illustrated their approach through the "Availability" quality sub-characteristic from the credit card system accounts. The functional size of the quality sub-characteristic is used to select the appropriate functionality to be implemented (*i.e.,* functional process) in order to reduce the development effort.

On the other hand, Hakim et al., [26] used a combination of functional and structural size [27] to evaluate security in web applications. More specifically, this study focused on the "Authenticity" quality sub-characteristic in order to identify the risk of authenticity violation. Ungan et al., [2] proposed a method that makes the system-NFR more easily converted to functional requirements in order to reduce the gap between approximate and precise size measurements. This study used the FSM patterns to facilitate the measurement of non-functional requirements at later software life-cycle phases, in particular the security quality characteristic. Moreover, the Software Non-functional Assessment Process (SNAP) was proposed by IFPUG to measure technical and quality requirements in order to improve the effort estimation, the schedule planning, etc. They defined a new unit which is "SNAP Counting Uni". SNAP identified four categories and 14 sub-categories to measure the non-functional requirements.

Recently, Meridji et al., [28] used COSMIC to measure security requirements at the function and service level to assist developers in specifying system security requirements in early development stages. To capture the security concept, this study used soft-goal inter-dependency graphs and three main security types: system availability, confidentiality and integrity.

TABLE I
Using FSM Methods for Measuring System-NFR

Research studies	**Method**	**Quality characteristics**
Kassab *et al.,* [12]	NFSM based on COSMIC-FFP	Availability & Security
SNAP [24]	New sizing method inspired from IFPUG	Performance
AbuTalib *et al.,* [25]	New measurement method based on COSMIC	Portability
Ungan *et al.,* [2]	COSMIC	Security
Abran *et al.,* [4]	COSMIC	Portability
Hakim *et al.,* [26]	Combination of functional and structural size	Security
Meridji *et al.,* [28]	NFR model based on COSMIC	Security

As summarized in Table I, many researchers used COSMIC to quantify the functional size of security requirements (*cf.,* [2], [26]). However, none of them focused on measuring changes in software security.

B. Security Quality Characteristic

Nowadays, security has become a critical challenging quality characteristic for any software application. It is a concern for software engineers when building software that manages stakeholders' resources, including intellectual property and identity information [22]. In fact, ignoring security at the initial software life-cycle phases may lead to a serious threat. Hence, security must be addressed at the beginning of the software process, built into the design, implemented in the coding, and verified during testing [22]. However, designing secure software is not an easy task within the limitation of developer's security knowledge. This may justify the number of changes that ask for the software security improvement. To verify the security of software, international standards for software measurement could be used.

According to the COSMIC and IFPUG organizations [23], System-NFR are classified mainly into three categories: quality requirements, system environment requirements and technical requirements. Security is considered as one of the eight quality characteristics identified by ISO/IEC 25010. The ISO/IEC 25010 quality model defines security as *"the degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization"* [15]. It includes five sub-characteristics which are confidentiality, integrity, non-repudiation, accountability and authenticity:

- Confidentiality: ensures that data are accessible only to those authorized to have access.
- Integrity: prevents unauthorized access to or modification of, software or data.
- Non-repudiation: Non-repudiation: actions or events can be proven to have taken place.
- Accountability: actions of an entity can be traced uniquely to entity.
- Authenticity: the identity of a subject or resource can be proved to be the one claimed.

Security controls are defined either by a set of functionality that the software must provide (*e.g.,* session timeout, authentication, etc.) or a crosscutting functionality added to the regular software functions. Ungan et al., [2] consider that for each software application it is required to verify the authentication, logging, session management, access control, and encryption. To verify these security controls, ISO standards have been proposed to help in quantifying software security. However, they are poorly adopted by the industry, mainly due to the ambiguity of their measurements [2]. This justifies the number of research studies that used FSM methods to quantify software security. Furthermore, the eminent changes of security requirements have called for an analysis and evaluation processes to manage properly their impact on the software project success.

C. Using COSMIC to Quantify Change Requests

In our previous work [21], we presented a decision support that helps decision-makers responding to a functional change. This decision support takes into account the main factors that can be used in evaluating requirements change requests (the functional size of the change, the preference of the change requester and the effort required to implement the change). We used COSMIC to quantify the functional size of the change request in terms of CFP unit. Based mainly on the functional size of the functional change, we proposed a rapid classification of the change status along three categories (“Minor”, “Moderate” and “Major”) [9] and a detailed classification that includes six categories (“Trivial”, “Minor”, “Moderate”, “Major”, “Critical” and “Extreme”) [10]. A change request with the status “Major”, “Critical”, or “Extreme” has a potential impact on the software development progress. However, a change with the status “Trivial”, or “Minor” can be handled without any impact on the software development progress. Whereas, a change with the status “Moderate” may require few changes in the software development progress.

Given the change status, decisions can be made to respond to the change request. Change status is determined by comparing the change functional size with the average value R quantified for a single software (see Equation (1)). R is equal to the total functional size of the software divided by the number of functional processes it includes.

$$R = \sum_{i=1}^{n} FS(FPi)/n \tag{1}$$

where:

- $FS(FPi)$: the functional size of the functional process FPi.
- *n:* the number of functional processes in the software.

As a result of our studies [9] [10], we showed that quantifying a change request using COSMIC allows a more *realistic* evaluation of the change status and its impact.

IV. Evaluating Security Change Requests with COSMIC FSM Method

Recall that the purpose of this paper is to quantify and evaluate security changes using COSMIC. In our previous work [21], we focused only on functional changes. Thus, this section presents a detailed description of the security change request evaluation approach and illustrates it application through the business application “C-Registration System” [16] [17].

A. Description of the Security Change Requests Evaluation Approach

To guide the decision on a security change, our approach quantifies the functional size of the change using COSMIC. As introduced in Section I, System-NFR may evolve into Software-FUR. Security is a quality characteristic that can be expressed through software functionality and then quantified using FSM methods.

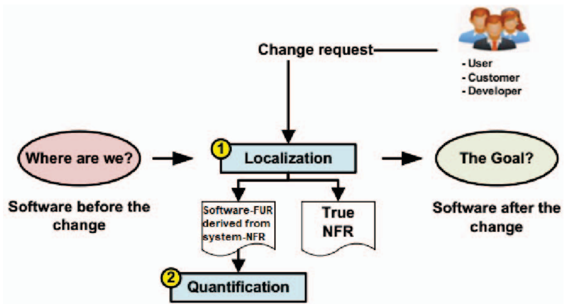

Fig. 3. Overview of the proposed approach

In Figure 3, we provide an overview of our approach. A security change request is proposed by customers, developers or users. The first step localizes the change as affecting software-FUR derived from system-NFR or True NFR. If the change affects software-FUR derived from system-NFR, it is possible to quantify its size using COSMIC and provide an evaluation of the change. Based on this evaluation, the decision-makers can make the appropriate decision responding to the change. This decision requires a deep analysis of the software situation before the change (*i.e.,* where we are) and the software situation after the change (*i.e.,* the goal) [29]. The situation before the change in our study is characterized by the functional size of the software before the change. Whereas, the software situation after the change is characterized by the functional size of the software after approving or rejecting the change and the functional size of the change as well. Based on this analysis, stakeholders can identify the possible solutions to achieve the goal.

As an example of a technical change, a change requester may ask to improve the “Customer Relationship Management” system performance for “Retrieve and View Customer Details” functionality by loading the ’customer details’ screen in 3 seconds or less; (Currently it takes 0.5 sec to 6 sec.) [14]. By providing a detailed analysis of this change, we note that the system is slow when the user asks for the details of a *big* customer with many products and many interactions. Thus, different solutions can be proposed to satisfy the change requester. For instance, Ben-Cnaan & Symons suggested to add an indicator (field) to customer’s table [14].

B. *Illustrative Example*

This section illustrates the application of our approach through the “C-Registration System (C-Reg)” [16] [17]. “C-Reg” is a business application that allows students to register for courses on-line and allows professors to manage their courses and to maintain their students’ grades. The first version of “C-Reg” includes 14 functional processes [16] and has a functional size of 105 CFP. The second version of “C-Reg” includes 19 functional processes and has a functional size of 97 CFP [17]. In our studies (*cf.,* [10], [21]), we quantified and evaluated changes from the first to the second version of “C-Reg”. Thus, we consider the first version of “C-Reg” as the “software before the change” situation and the second version of “C-Reg” as the “software after the change” situation.

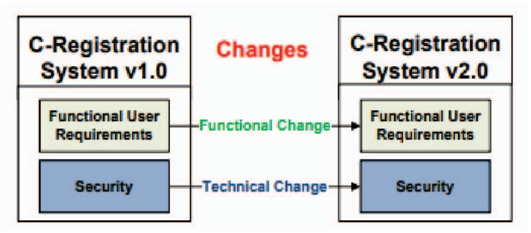
C-Registration System v1.0
Changes
C-Registration System v2.0
Functional User Requirements
Functional Change
Functional User Requirements
Security
Technical Change
Security

Fig. 4. Changes categories between the two versions of the “C-Registration System” case study

As shown in Figure 4, the changes between the two versions of this case study can be classified into either functional changes or technical changes:

- Functional changes are changes between the two versions related to the functional user requirements. A detailed analysis of these functional changes is given in [10].
- Technical changes are changes between the two versions related to the software-FUR derived from system-NFR. In this paper, we focus on changes that are required to perform the security controls as identified by [2].

To provide a more accurate evaluation of the approved changes between the two versions of the “C-Reg”, we quantify the functional size of these changes at two levels of details. The first level is related to the functional level and the second level is related to the technical level (*e.g.,* the security requirements). In our previous work [10], we quantified and evaluated the changes between the two versions of this case study at the first level (*i.e.,* functional level). In this paper we complete our work by quantifying and evaluating the functional size of the security controls that are incorporated into functional changes. This quantification will provide a more *accurate* evaluation of these changes. As we zoom-in to a lower level of details, technical changes become easily visible. In addition, the availability of the actual changes’ size at two levels of details (functional and technical) will have an added benefits for estimating the effort required to implement the changes (approximate and/or more accurate effort estimates).

For instance, let us focus on the following three functional processes: “Add a Professor”, “Modify a professor” and “Enquire on Course Offerings”. The functional size measurement results of both functional and security controls changes for these three functional processes are given in Table II. The functional sizes of the functional processes “Add a Professor” and “Enquire on Course Offerings” in the second version of the “C-Reg” with the security controls have been provided by [2]. To quantify the functional size of “Modify a professor” functional process in the second version of the “C-Reg” with the security controls, we use the FSM patterns as proposed by [2]. These FSM patterns are also applied to size security controls that are related to the three selected functional processes in the first version of the “C-Reg” [16]. In Table V, we provide a detailed measurement of the functional size of the “Modify a professor” functional process with security controls in both the first and second versions of the “C-Reg” case study.

Regarding the functional size of functional requirements and as shown in Table II, the functional process “Modify a professor” has a functional size of 6 CFP in the first version of “C-Reg” and a functional size of 3 CFP in the second version. Hence, the functional change required in this functional process has a functional size of 3 CFP. This functional change corresponds to the deletion of three data movements from the first version of “C-Reg” (one Entry, one Read and one eXit). Taking into account the security controls incorporated into the same functional process “Modify a professor”, we observed that in the first version of the “C-Reg”, this functional process has a functional size of 11 CFP. While the functional size of that functional process in the second version of the “C-Reg” is 5 CFP instead of 11 CFP. The technical change related to the security controls in this functional process has a functional size of 6 CFP. These changes correspond to the security controls related to six deleted data movements (two Reads, two eXits, and two Entries) from the first version.

Table III summarizes the total changes functional size between the first and the second versions of “C-Reg” for the three selected functional processes. For example, the functional size of the technical changes in “Modify a professor” (*i.e.,* changes related to the security controls) is equal to 66% of the total changes functional size (including functional and technical levels). Thus, the functional size of the changes related to the security controls is more important than the functional size of

TABLE II
SUMMARY OF THE FUNCTIONAL AND TECHNICAL CHANGES SIZE MEASUREMENT

Functional Process (FP)	Change Description	Functional Size of FUR			Functional Size of System-NFR		
		FSi(FP)	FSf(FP)	FS(FC)	FSi(FPS)	FSf(FPS)	FS(TC)
Add a professor	Delete 1 Read	5 CFP	4 CFP	1 CFP	7 CFP	6 CFP	1 CFP
Modify a professor	Delete 1 Entry, 1 Read and 1 eXit	6 CFP	3 CFP	3 CFP	11 CFP	5 CFP	6 CFP
Enquire on Course Offerings (Professor)	Add the FP (Enquire on Course Offerings (Professor))	–	6 CFP	6 CFP	–	12 CFP	12 CFP

FSi(FP): the functional size of the functional process in the 1st version, **FSf(FP):** the functional size of the functional process in the 2nd version, **FS(FC):** the functional size of the functional change, and **FS(TC)**: the functional size of the technical change.

TABLE III
TECHNICAL AND FUNCTIONAL CHANGES FUNCTIONAL SIZE MEASUREMENT

Functional Process	Total size	FS(FC)	FS(TC)
Add a Professor	2 CFP	1 CFP (50%)	1 CFP (50%)
Modify a professor	9 CFP	3 CFP (33%)	6 CFP (66%)
Enquire on Course Offerings	18 CFP	6 CFP (33%)	12 CFP (66%)

TABLE IV
CHANGES' STATUS EVALUATION

Functional Process	Change Status	
	with security controls	without security controls
Add a Professor	Minor	Trivial
Modify a professor	Major	Minor
Enquire on Course Offerings	Major	Moderate

the changes related to the functional level. This was expected because the expansion of the changes. We conclude that for an appropriate evaluation of a change request it is required to move to the technical level.

Using our approach to evaluate the status of functional changes in [10], we proved the hypothesis that the size of a technical change will lead to a big impact on the total change functional size. As it is illustrated in Table IV, the change status with security controls and without security controls are not the same for all the three selected functional processes. For example, the status of the changes in the "Modify a professor" functional process without the security controls quantification is "Minor". However, with the security controls quantification, the status of these changes becomes "Major".

V. DISCUSSION AND THREATS TO VALIDITY

To the best of our knowledge, the software development industry lacks quantitative evaluation approach for technical changes, although the wide attention gave to the non-functional requirements. Hence, they will certainly benefit from the approach that we propose in this paper.

For an appropriate quantification/evaluation of a change, it is required to measure its *actual* functional size including both functional and technical levels. In fact, at different levels of expansion (*i.e.,* details), the description of the same change is different. As we move to a more detailed level, new details of the change will appear, and hence more data movements can be identified. Thus, a change request may look as functional change at the beginning of its analysis, but it may also affect the system-NFR in reality (*e.g.,* security, portability, etc.).

Furthermore, Table III shows the importance of the second level (*i.e.,* technical level) in the quantification/evaluation of any change request. In fact, the percentages of the functional size of changes related to the security controls (FS(TC) in Table III) are at least equal to 50% of the total functional size of the change. Hence, the security controls functional size has a *big* impact on the total change size. For this reason, quantifying the functional size of any change at the technical level will provide a more realistic evaluation of the change request, and hence an appropriate estimation of the effort required for its implementation. Consequently, an accurate decision responding to the change request will be made.

Threats to the validity of our study are pertinent to internal validity, external validity, and finally construct validity.

- **Internal validity:** the main threat to the internal validity of our approach is that the change, to be accurately quantified, must be well detailed at both functional and technical levels. However, this level of details is not always available in practice. Nonetheless, further studies will be needed to quantify changes at a high abstraction level of details. Moreover, in the quantification of the functional size of the security controls, we used the FSM patterns to size security non-functional requirements proposed by [2]. Hence, we were limited on the security controls identified in this study. However, these patterns are given based on a particular interpretation of relevant quality controls (*e.g.,* access control, encryption, etc.).
- **External validity:** deals with the generalization of the results of this study to other case studies. In fact, the lack of case studies makes the generalization of this study's results difficult. However, to test the proposed approach in practice, we are collecting data from companies. On the other hand, the herein presented work can be generalized to cover other system-NFR since researchers proved that COSMIC can be successfully used to quantify the functional size of system-NFR (*e.g.,* portability [25], maintainability [30], etc.). Thus, we believe that this method can be used for sizing change requests affecting not only

TABLE V
FS (“Modify a professor”) of C-Reg v1.0 and C-Reg v2.0 Incorporating the Selected Security Controls

“C-Registration System” case study v1.0					
Functional User	**Sub-Process Description**	**Data Group**	**Data movement type**	**CFP (first level)**	**CFP (second level)**
Registrar	~~Registrar enters Professor ID~~	~~Professor ID~~	~~Entry~~	~~1~~	-
	~~Read validation reference for input validation~~	~~Input Val.Ref~~	~~Read~~	-	~~1~~
	~~Access Control~~	~~Credentials~~	~~eXit~~	-	~~1~~
	~~Access Control~~	~~Privileges~~	~~Entry~~	-	~~1~~
	~~The system retrieves the Professor information~~	~~Professor data~~	~~Read~~	~~1~~	-
	~~Read Logging~~	~~Data read details~~	~~Read~~	-	~~1~~
	~~Encryption Request~~	~~Data Pack~~	~~eXit~~	-	~~1~~
	~~Encryption Return~~	~~Data Pack~~	~~Entry~~	-	~~1~~
	~~The system displays the Professor information~~	~~Professor data~~	~~eXit~~	~~1~~	-
	~~Read validation reference for output validation~~	~~Output Format~~	-	-	-
	The Registrar enters the modified Professor data	Professor data	Entry	1	-
	Read validation reference for input validation	Input Val.Ref	Read	-	1
	Access Control	Credentials	eXit	-	1
	Access Control	Privileges	Entry	-	1
	~~When changes are complete, the Registrar selects Save~~	-	-	-	-
	The system updates the Professor information	Professor data	Write	1	-
	Write Logging	Data write details	Write	-	1
	Read validation reference for output validation	Output Val.Ref.	Read	-	1
	Display error message	Message	eXit	1	-
Total				**6 CFP**	**11 CFP**
“C-Registration System” case study v2.0					
Functional User	**Sub-Process Description**	**Data Group**	**Data movement type**	**CFP (first level)**	**CFP (second level)**
Registrar	The Registrar enters the modified Professor data	Professor data	Entry	1	-
	Read validation reference for input validation	Input Val.Ref	Read	-	1
	Access Control	Credentials	eXit	-	1
	Access Control	Privileges	Entry	-	1
	C-Reg validates the details and updates the Professor record	Professor details	Write	1	-
	Write Logging	Data write details	Write	-	1
	Read validation reference for output validation	Output Val.Ref.	Read	-	1
	Display error message	Message	eXit	1	-
Total				**3 CFP**	**5 CFP**

CFP (first level): size measurement of data movements related to the functionality, **CFP (second level):** size measurement of data movements related to the security controls.

the ISO 25010 quality characteristics (*e.g.,* portability, performance, etc.) but also quality sub-characteristics (*e.g.,* availability, flexibility, etc.).

- **Construct validity:** is related to the relationship between theory and observation. In our case, this approach must be applied in industrial practice. In fact, the illustrative example “C-Registration System” case study showed that the proposed approach can be used to quantify the actual size of both functional and technical changes, and therefore identify the *real* change status. In addition, the use of this approach provides a basis for clarifying the boundary between functional requirements and software-NFR. However, we believe that testing this approach with real data is important to ensure its validity.

VI. Conclusion

Changing software requirements can be a challenging task in the software development project. They may degrade the software design and quality. In addition, changing system requirements made the effort estimation difficult. Certainly, changes in system requirements are one of the biggest foes for the software development team. However, changes are inevitable throughout the software life-cycle. For this reason, the use of the COSMIC standard in quantifying the functional size of a change and identifying its impact on the software development progress will be helpful for the development team. Based mainly on the functional size of the change, decision-makers will make the appropriate decision whether to accept, defer or deny the change.

In our previous work, we measured the functional size of a change at the functional level [21]. In this paper, by zooming-in the change, we quantified the functional size of the change request at the second level of details (*i.e.,* technical level), in particular the security quality characteristic. Zooming-in again may lead to the appearance of more data movements related to quality sub-characteristics. However, the main focus of this paper was on the security quality characteristic. Thus, we showed, through the “C-Registration System” case study, that the functional size of the change at the technical level has a significant impact on the total change size. Hence, decisions made without taking into account the technical level may not reflect the quality aspect of the change.

The main question that we will address in our future work is *how functional changes impact on the software quality?.*

For this purpose, we will focus on other system-NFR such as portability, performance, etc. In addition, although the illustrative example showed the feasibility of the proposed approach in this paper, it is also required to conduct an empirical study to assess the importance of the technical aspect in the evaluation of a change in practice. To prepare for such empirical study, we are in the process of implementing CASE tools to get automatically the quantification results.

REFERENCES

[1] T. Creasey and J. Hiatt, *Best Practices in Change Management, Prosci, Inc.; 1st edition*, 2014.

[2] E. Ungan, S. Trudel, and L. Poulin, "Using fsm patterns to size security non-functional requirements with cosmic," in *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, ser. IWSM Mensura '17, 2017, pp. 64–76.

[3] B. Ozkan, O. Turetken, and O. Demirors, "Software functional size: For cost estimation and more," in *Software Process Improvement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 59–69.

[4] A. Abran, *Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers*. Wiley-IEEE Computer Society, 2015.

[5] IFPUG, *ISO/IEC 20926:2009, Software and systems engineering – Software measurement – IFPUG functional size measurement method*, December 2009.

[6] NESMA, *ISO/IEC 24570:2018, Software engineering - NESMA functional size measurement method - Definitions and counting guidelines for the application of function point analysis*, October 2018.

[7] COSMIC, *The COSMIC Functional Size Measurement Method, Version 4.0.2, Measurement Manual*, October 2017.

[8] C. Symons, *A Comparison of the Key Differences between the IFPUG and COSMIC Functional Size Measurement Methods*, The Common Software Measurement International Consortium, 2011.

[9] M. Haoues, A. Sellami, H. Ben-Abdallah, and N. Elleuch Ben Ayed, "Quantitative functional change impact analysis in activity diagrams: A cosmic-based approach," in *The 25th International Workshop on Software Measurement and the 10th International Conference on Software Process and Product Measurement (IWSM/Mensura), Kraków, Poland, 5-7 October*, 2015, pp. 78–95.

[10] M. Haoues, A. Sellami, and H. Ben-Abdallah, "Functional change impact analysis in use cases: An approach based on COSMIC functional size measurement," *Science of Computer Programming*, vol. 135, pp. 88 – 104, 2017, special Issue on Advances in Software Measurement.

[11] F. W. Vogelezang, E. R. Poort, E. van der Vliet, H. R. J. Mols, R. Nijland, and J. de Vries, "A shortcut to estimating non-functional requirements?: Architecture driven estimation as the key to good cost predictions," in *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, ser. IWSM Mensura '17. ACM, 2017, pp. 77–81.

[12] M. Kassab, O. Ormandjieva, M. Daneva, and A. Abran, "Software process and product measurement." Springer-Verlag, 2008, ch. Non-Functional Requirements Size Measurement Method (NFSM) with COSMIC-FFP, pp. 168–182.

[13] K. Al-Sarayreh, "Identification, specification and measurement, using international standards, of the system non functional requirements allocated to real-time embedded software," Master's thesis, Ecole de technologie superieure, University of Quebec, Montreal, Canada, 2011.

[14] T. Ben-Cnaan and C. Symons. Accounting for non functional and project requirements - cosmic and ifpug development. Danske Bank. [Online]. Available: https://fr.slideshare.net/iwsm-mensura/accounting-for-non-functional-and-project-requirements-cosmic-and-ifpug-development-talmon-bencnaan-charles-symons-54258283

[15] ISO/IEC, "25010 system and software quality models," Tech. Rep., 2010.

[16] *Software Engineering Research, C-Registration System case study with ISO/IEC 19761, Version 1.0*, 2008.

[17] *Software Engineering Research, C-Registration System case study with ISO/IEC 19761, Version 2.0*, 2015.

[18] P. Singh, D. Singh, and A. Sharma, "Rule-based system for automated classification of non-functional requirements from requirement specifications," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2016, pp. 620–626.

[19] R. Fairley, *Managing and Leading Software Projects*. Wiley-IEEE Computer Society Pr, 2009.

[20] S. Ghosh, S. Ramaswamy, and R. P. Jetley, "Towards requirements change decision support," in *The 20th Asia-Pacific Software Engineering Conference (APSEC). Bangkok, Thailand, 2-5 Dec*, vol. 1, Dec 2013, pp. 148–155.

[21] M. Haoues, A. Sellami, and H. B. Abdallah, "Towards functional change decision support based on cosmic fsm method," *Journal of Information and Software Technology*, vol. 110, pp. 78–91, 2019.

[22] B. R. Maxim and M. Kessentini, "Chapter 2 - an introduction to modern software quality assurance," in *Software Quality Assurance*. Boston: Morgan Kaufmann, 2016, pp. 19 – 46.

[23] COSMIC and IFPUG, *Glossary of terms for Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating*, September 2015.

[24] IFPUG, "The software non-functional assessment process," http://www.ifpug.org/about-snap/, 2018.

[25] F. A. Talib, D. Giannacopoulos, and A. Abran, "Designing a measurement method for the portability non-functional requirement," in *The Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement*, Oct 2013, pp. 38–43.

[26] H. Hakim, A. Sellami, and H. Ben-Abdallah, "Evaluating security in web application design using functional and structural size measurements," in *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement, IWSM-MENSURA 2016, Berlin, Germany, October 5-7, 2016*, 2016, pp. 182–190.

[27] A. Sellami, H. Hakim, A. Abran, and H. Ben-Abdallah, "A measurement method for sizing the structure of UML sequence diagrams," *Information & Software Technology*, vol. 59, pp. 222–232, 2015.

[28] K. Meridji, K. T. Al-Sarayreh, A. Abran, and S. Trudel, "System security requirements: A framework for early identification, specification and measurement of related software requirements," *Computer Standards Interfaces*, 2019.

[29] K. Gilb, *Evolutionary Project Management & Product Development*. Evo - Evolutionary Project Management, 2017.

[30] K. T. Al-Sarayreh, A. Abran, and J. J. Cuadrado-Gallego, "A standards-based model of system maintainability requirements," *Journal of Software: Evolution and Process*, vol. 25, no. 5, pp. 459–505, 2013.