A Meta-model for the Assessment of Non-Functional Requirement Size

M. Kassab¹, M. Daneva², O. Ormandjieva¹

¹Concordia University, Canada, {moh_kass, ormandj}@cse.concordia.ca

²University of Twente, The Netherlands, m.daneva@utwente.nl

Abstract

Non-functional requirements (NFRs) pose unique challenges in estimating the effort it would take to implement them. This is mainly because of their unique nature; NFRs are subjective, relative, interactive and tending to have a broad impact on the system as a whole. Nevertheless, it is crucial, when making decisions about the scope of software by given resources and budget, to furnish a justifying and quantitative analysis based on both Functional Requirements (FRs) and NFRs. This paper presents a meta-model which complements the FR dimension with the NFRs as another dimension to be used in effort estimation approaches. The meta-model is deployed to extend the use of the COSMIC functional size measurement method to measure the size of NFRs, as effort is a function of size. We report on a case study to demonstrate our approach in context.

1. Introduction

Estimating the size and effort variables is key to successful software project management [20]. A good estimation of these variables available right from the start in a project gives the project manager confidence about any future course of action, since many of the decisions made during development depend on, or are influenced by, the initial estimations. In practice, however, many organizations focus their estimation of the effort on functional requirements (FRs) [15], while non-functional requirements (NFRs) remain, by and large, neglected. This is mainly because NFRs are subjective by nature [2], and they tend to become scattered among multiple modules when they are mapped from the n-dimensional requirements domain to the one-dimensional solution space. Furthermore, NFRs can often interact, in the sense that attempts to fulfill one NFR can help or hinder the fulfillment of other NFRs at a particular level of functionality. Such an interaction creates an extensive network of interdependencies and trade-offs among NFRs which is not easy to model, nor is it easy to estimate its influence on the effort [2]. Nevertheless, it is crucial,

when making decisions on the scope of software projects by given resources and budget, to furnish a justifying and quantitative analysis based on both FRs and NFRs. Valid examples of errors of omission or commission in properly taking into account NFRs which led to catastrophic project failures are: London Ambulance System (LAS) in 1992 [1], Mars Climate Orbiter in 1998 [3], Therac 25: The Medical Linear Accelerator [4], and the Mercedes A-Class (1997) [5].

This paper presents a meta-model which complements the FR dimension with the NFRs as another dimension to be used in effort estimation approaches. We propose the meta-model as a solution to counterbalance the need to deal comprehensively with the effect of a particular NFR on the effort of building the software project. Specifically, we address this need by: (1) measuring the functional size of the NFR in isolation from its relations with the system functionalities, as effort is a function of size [9]; (2) understanding and specifying those relations of the NFR; and (3) re-measuring the functional size of the NFR in association with the system using the measurement from (1) and the specification of NFR's relations from (2).

In this paper, we use the COSMIC [7, 8] functional size measurement method to quantify NFR size in a software project. We deploy the meta-model to extend the use of COSMIC to measure the size of NFRs. We also report on a case study to demonstrate our approach in context.

The remainder of this paper is organized as follows: Section 2 provides a background on related work and the COSMIC method. Section 3 presents our requirements meta-model. Section 4 introduces the NFR size measurement method. Section 5 provides a discussion on the applicability of the approach. Section 6 concludes the paper.

2. Background

2.1. Related work

Existing functional size measurement (FSM) methods have been primarily focused on sizing the

functionality of a software system. Size measures are expressed as single numbers (function points (FP) [7, 12, 13, 16]), or multidimensional ‘arrays’ designed to reflect how many of certain types of items there are in a system [14]. To the best of our knowledge, the existing literature does not report on the use of FP in the process of estimating NFRs, nor on the use of the multidimensional size measure described in [14]. The existing function point-based FSM techniques have so far addressed the topic of NFRs only with respect to the task of adjusting the (unadjusted) FP counts to the project context or the environment in which the system is supposed to work. For example, the International Function Point Users Group (IFPUG) [22] has been approaching the inclusion of NFRs in the final FP count by using qualitative judgments about the system’s environment. The current version of the IFPUG Function Point Analysis (FPA) manual [16] speaks of a set of General System Characteristics and Value Adjustment Factors, all meant to address – though in different ways – the NFRs that a project may include.

Currently, there are four FSM models which are proposed by the COSMIC consortium and IFPUG member associations (namely, NESMA [13], UKSMA [12], COSMIC [7], and IFPUG [16]) and which are recognized as ISO standards. We compared and contrasted the ways in which NFRs are treated in these FSM standards. For each standard, we looked at what NFR artifact is used as input to the FSM process, how this artifact is evaluated (Table 1), and which FSM counting component reflects the NFRs. We found that all four FSM standards provide, at best, checklists which estimators can use to perform qualitative assessments of certain factors of the system’s environment. However, these assessments reflect the subjective view of the professionals who run the FSM process. The FSM standards say nothing about what should be put in place to enable estimators to ensure the reproducibility of their assessment results regarding the NFRs in a project. The Mark II FPA manual [12] refers to recent statistical analysis results and suggests that neither the Value Adjustment Factors from the IFPUG method [16] nor the Technical Complexity Adjustment (TCA) factors from the Mark II FPA method [12] represent well the influence on size of the various characteristics these two methods try to take into account. Indeed, the Mark II FPA manual says that the TCA factors are included only because of continuity with previous versions, and recommends that these factors be ignored altogether (p. 63 in [12]) when sizing applications within a single technical environment (where the TCA is likely to be constant).

Table 1. The four ISO FSM standards.

Proposal	Input NFR artifact	Assessment	Counting component
COSMIC [7]	not included	not applicable	none
NESMA [13]	Textual NFR	qualitative	General System Characteristics, Value Adjustment Factors
MARK II FPA method v.1.3.1 [12]	Textual NFR	qualitative	Technical Complexity Adjustment
IFPUG [16]	Textual NFR	qualitative	General System Characteristics, Value Adjustment Factors

2.2. The COSMIC method

For the purposes of this research, we have chosen to use the COSMIC FSM method [7] developed by the Common Software Measurement International Consortium (COSMIC) and now adopted as an international standard (ISO/IEC 19761 [8]). We chose this method in particular because it conforms to all ISO requirements (ISO 14143-1 [10]) for functional size measurement, and addresses some of the major theoretical weaknesses of the earlier FPA techniques like Albrecht’s FPs [11]. COSMIC focuses on the “user view” of functional requirements, and is applicable throughout the development life cycle, from the requirements phase right through to the implementation and maintenance phases. The process of measuring software functional size using the COSMIC method implies that the software functional processes and their triggering events be identified.

In COSMIC, the basic functional components are data movements. The unit of measure is 1 COSMIC Function Point (CFP) which refers to a movement of one or more data attributes belonging to a single data group. Data movements can be of four types: Entry, Exit, Read or Write. The functional process is an elementary component of a set of user requirements triggered by one or more triggering events, either directly or indirectly, via an actor. The triggering event is an event occurring outside the boundary of the measured software and initiates one or more functional processes. The data movements of each functional process are sequences of events. A functional process comprises at least two data movement types: an Entry plus at least either an Exit or a Write. An Entry moves a data group, which is a set of data attributes, from a user across the boundary into the functional process, while an Exit moves a data group from a functional process across the boundary to the user requiring it. A Write moves a

data group lying inside the functional process to persistent storage, and a Read moves a data group from persistent storage to the functional process. Figure 1 illustrates the generic flow of data attributes through software from a functional perspective.

Figure 1: Generic flow of data attributes through software from a functional perspective [7]

3. Explicit NFR modeling

In order to explicitly reason about the impact of NFRs on the effort required to build the software, it is necessary that the corresponding NFRs and their relations be explicitly modeled. In this section, we introduce the requirement relations meta-model, which is schematically represented in the UML domain model in Figure 2. In this figure, NFRs are modeled as parts of a requirements group which, in turn, is a part of a requirements model. The left-hand side of Figure 2 presents the functional model, and shows that an FR is mapped to the COSMIC FSM model, which distinguishes between the four data movements (Entry, Exit, Read, and Write) discussed in section 2.2. The right-hand side of Figure 2 shows the part of the meta-model that models the hierarchy of NFRs and their relations. Four relations are identified, namely, association, decomposition, operationalization, and interactivity.

Association. NFRs do not represent stand-alone goals, as their existence is always associated with other elements of the system. In this work, we define three elements (association points) with which an NFR and its derived solutions (the so-called

operationalizations) can be associated throughout the software development process:

- **The FR (and sub process mapped from the FR):** This refers to the context for functionality-related NFRs. For example, associating the *fast response time* NFR with *place order* functionality would indicate that the system must execute the *place order* functionality within an acceptable length of time. If an NFR is associated with functionality, then some or all the offspring sub processes that are mapped from this functionality will inherit this association. Yet, an NFR could be associated with an offspring sub process without being associated with the parent functionality.

- **Resource:** This refers to external entity-related NFRs. Example of such NFRs would be: *The software maintainers have to have 2 years of Oracle database experience.* This is an operating constraint which is associated with the candidates for the maintenance position for the system; they are considered as resources for the project.

- **Project:** This refers to those NFRs providing a precise context to the project or development process. Examples of such NFRs would be: *The project will follow the Rational Unified Process (RUP)* and *The activities X, Y, Z will be skipped for this project.*

Decomposition. This refers to the relation that decomposes a high-level NFR into more specific sub-NFRs. In each decomposition, the offspring NFRs can contribute partially or fully towards satisficing the parent NFR. Let us consider the requirement, *managing transactions with good security.* The *security* requirement constitutes quite a broad topic [2]. To deal effectively with such a requirement, the NFR may need to be broken down into smaller components, so that an effective solution can be found. Thus, *security* can be decomposed into *integrity*, *confidentiality*, and *availability*. The decomposition can be “ANDed” (all NFR offspring are required to achieve the parent NFR goal) or “ORed” (it is sufficient that one of the offspring be achieved instead, the choice of offspring being guided by the stakeholders).

Operationalization. This refers to the relation that refines the NFR into solutions in the target system that will satisfice the NFR. These solutions provide operations, functions, or design decisions in the target system to meet the needs stated in the NFRs. Those operationalizations that correspond to functions or operations are mapped to the COSMIC model for the purpose of measuring the functional size of the parent NFR, as will be discussed in the next section. We note, however, that the existence of an association between a parent NFR and an FR (e.g. *security* and

place order) implies that an association exists between one or more of those operationalizations which are derived from the parent NFR and the sub processes (data movements) mapped from the FR. Figure 3 illustrates this situation. The question mark notation “?” indicates that a further analysis from the stakeholders is required to determine the existence of the relation.

Figure 3. Implicit relations among NFRs, operationalizations, and association points.

Figure 2: The meta-model for NFRs, FRs, their relations, and their mappings to the COSMIC model

Interactivity. NFRs by themselves do not interact, as they represent static goals to be achieved. However, their associations with functionalities could interact, in that attempts to fulfill one NFR at a certain association point can hinder (negative interaction) or help (positive interaction) the fulfillment of other NFRs at the same association point, e.g. *security* and *performance* at *place order* functionality. Two NFRs negatively affect each other if they can be traced to the same association point

and, at the same time, compete for the same resources. This can be illustrated with the *security* and *performance* example above (the resource is CPU time). Conflict-resolving algorithms need then to be applied to solve the conflicts between negatively interacting NFRs based on an optimization of the resources.

Positive interaction would involve an offspring NFR and its parent NFR in the case of “ANDed” decomposition. In “ORed” decomposition, only the

sub NFRs, which are selected by the stakeholders, will positively affect the parent NFR. The interaction is not necessarily a symmetrical relation.

4. Measuring the functional size of NFRs

While the COSMIC method was originally proposed to measure functional user requirements, in this section, we extend its use to measuring the functional size of the operationalized NFRs. The process of measuring the functional size for a particular NFR is carried out in three steps:

Step 1: The NFR is considered in isolation from its association relations. COSMIC is used to measure the functional size for those operationalizations refined from the NFR and which correspond to functions/operations. The size of the NFR is the sum of the size of all selected operationalizations.

Step 2: The NFR's association relations with the FRs are clearly captured. The inheritance of the association between the operationalizations from the NFR side and the corresponding sub processes from the functional side must be clearly defined, as suggested by Figure 3.

Step 3: The total size of the NFR within the system is then calculated by measuring the total changes of the functional size of functionalities triggered by introducing the associated NFR.

We completed our first application of this procedure in a case study setting at a company site. To illustrate how it works, we use the OZ Mobile Email application developed by OZ Communications in Montreal. This application consists of the OZ Mobile Email Gateway and the OZ Mobile Email Client. The high-level context diagram of the application is presented in Figure 4. The Mobile Email Gateway provides mobile operators with the necessary protocol adaptations, billing, reporting, and customer care interfaces they require to effectively deliver branded portal email services to their subscribers. As a result, mobile operators can increase their average revenue per user and directly impact their bottom line with a variety of flexible billing options.

The OZ Mobile Email Client provides the user interface. Using recognizable and branded clients, the mobile email experience mirrors the familiar 'look and feel' of the PC, generating instant consumer adoption and virtually eliminating the learning curve.

The communication between the client and the gateway is established through a SYNCML protocol which is an XML based standard for data synchronization.

Figure 4: Mobile Email solution

To illustrate the measurement procedure, we will limit the discussion to two pieces of functionality: (1) User asks to read an email message; and (2) User composes and sends a new email. The specification of these functionalities is illustrated in Figure 5 (Appendix 1). The COSMIC models are generated for each component (here, Client and Gateway), as outlined below:

The chosen functional requirements - Read and Send, each consists of two functional processes, which are further refined into data movements (see section 2.2). The identified data groups for these Read and Send FRs are: 1) Read request data group (includes data on the requested message), 2) Read response data group (includes the requested message to be read), 3) Send request data group (include the composed message to be sent) and 4) Send response data group (confirmative message).

The functional size for each FR corresponds to the addition of all identified data movements. The initial calculated functional size for the Client component is 11 CFP (see Tables 2 and 4) and 12 CFP for the Gateway component (see Tables 3 and 5)

Table 2: Client component (*Send a message functionality*)

ID	Process description	Triggering event	Data Movement	Data Group	Data movement Type	CFP
FP1	Send	Request event	Receive send request	Send request	E	1
			Save the message in the buffer	Send request	W	1
			Send message to gateway	Send request	X	1
		Response event	Receive confirmation	Send response	E	1
			Translate message	Send response	W	1
			Display confirmation	Send response	X	1
Total functional size of Send FUR for Client component in CFP =						6

Table 3: Gateway Component (*Send a message functionality*)

ID	Process description	Triggering event	Data Movement	Data Group	Data movement Type	CFP
FP1	Send	Request event	Receive send request	Send request	E	1

			Translate a message to IMAP/POP3	Send request	W	1
			Send message to mail server	Send request	X	1
		Response event	Receive confirmation	Send response	E	1
			Translate message to SYNCML	Send response	W	1
			Send confirmation to client	Send response	X	1
Total functional size of Send for Gateway component in CFP =						6

Table 4: Client component (*Read a message* functionality)

ID	Process description	Triggering event	Data Movement	Data Group	Data movement Type	CF P
FP2	Read	Request event	Receive read request	Read Request	E	1
			Send message to gateway	Read Request	X	1
		Response event	Receive user's message	Read Response	E	1
			Translate message	Read Response	W	1
			Display message	Read Response	X	1
Total functional size of Read for Client component in CFP =						5

Table 5: Gateway Component (*Read a message* functionality)

ID	Process description	Triggering event	Data Movement	Data Group	Data movement Type	CF P
FP 2	Read	Request event	Receive read request	Read Request	E	1
			Translate the request to IMAP/POP3	Read Request	W	1
			Send request to mail server	Read Request	X	1
		Response event	Receive user's message	Read Response	E	1
			Translate message to SYNCML	Read Response	W	1
			Send the requested message to client	Read Response	X	1
Total functional size of Read for Gateway component in CFP =						6

In order to optimize the user experience for devices with limitations (e.g. screen size, memory, processing speed) and wireless networks with constrained bandwidth, some NFRs have to be adapted by the requirements model of the project. In this paper, we consider adaptation of the performance requirement. Performance is defined as the amount of useful work accomplished by software compared to the time and resources used. To deal effectively with such a requirement, a *good performance* requirement may need to be broken down into smaller components, so that an effective solution can be found. Thus, *performance* can be *decomposed* into *short response time* for the exchanged transactions between the client and the gateway and *high*

throughput (rate of processing work) for the network bandwidth.

After an extensive round of meetings and discussions, the software architects decided to optimize the *response time* and *throughput* by means of the following two solutions: (1) a compression algorithm, which compresses the requests and responses exchanged between the device application and the gateway; and (2) breaking a message requested to be read into smaller pages, each 1 Kb in size, after which only the first page is sent to the client, with the option for the user to request the other pages from the gateway in separate transactions.

The suggested operationalizations proved to reduce the response time perceived by the end-user in similar projects. They also reduced the amount of wireless traffic. The performance requirement, along with its decomposition, operationalization, and association relations, are depicted in Figure 5 (Appendix 1).

The OZmail protocol compression algorithm reduces the size of the protocol elements or XML markup, and not of the actual email data. The algorithm is based on a static compression dictionary which contains a list of the most common protocol fragments. During compression, the source XML message is split up into dictionary and non-dictionary words (logic). A special dictionary is searched (Read) and each fragment that maps to a dictionary word is replaced with the corresponding index (Write). A fragment which does not map to a dictionary word is replaced with its length in bytes using UTF-8 encoding plus 1000 followed by the fragment itself (Write). During decompression, these sub processes are reversed. In total, the functional size for the compression operationalization is obtained by summing up all the identified data movements. The initial calculated functional size is $3 * 2 = 6$ CFP.

The breaking down of a message by the gateway into smaller pages was mapped into three sub processes: The gateway recognizes that the message size exceeds 1 Kb and decides to break it down into smaller pieces (Entry), The gateway writes the first page of the message into a special buffer to be sent to the client (Write) right away, and then the gateway stores the rest of the message into a special memory (Write) for future requested transactions. The functional size for breaking the message down into pages is 3 CFP.

To calculate the functional size of the performance NFR, we consider the association of the performance requirement and the association of their derived operationalizations presented in Figure 5 (Appendix 1). The compression algorithm (including both the compression and decompression) has to be called

once for each data groups. This increases the total functional size for both functionalities by $(4 * 6 = 24 \text{ CFP})$. In the case of breaking down the message, it is called on once for “read message”. Thus, the functional size of the “read message” is increased by 3 CFP. The calculated functional size for performance is, therefore, the sum of the two functional sizes: $24 + 3 = 27 \text{ CFP}$. The updated total functional size for both functionalities (“send a message” and “read a message”) after introducing the performance requirement is $27 + 11 + 12 = 50 \text{ CFP}$.

5. Discussion

Measuring the functional size of NFRs as presented in our approach falls under the Count, Compute, Judge estimation technique [21], which means, basically, that the first course of action consists of counting and computing. If there is a way to directly count and compute some value to provide the estimate, this should be the best option, since it usually provides the most accurate result. If “count and compute” is not possible, then “judge” is considered, but as a last resort only, as it introduces the greatest opportunity for bias and error.

As illustrated in section 4, our approach is applicable to the NFRs associated with FRs and operationalized through functions/processes which could be mapped to the COSMIC model. Nevertheless, the goal-oriented RE community [17, 18, 19] considers that not all NFRs should be decomposed into functions/processes. If NFRs serve as norms [18] or as criteria for making architectural design choices, then they should not be decomposed into FRs. Examples are global NFRs like survivability, reporting, and customizability. For those NFRs, expert judgment calibrated via historical project data is the main technique for arriving at effort estimates. Our future work includes further investigation of these NFRs.

6. Conclusion

This paper proposed a solution based on the requirements meta-model to deal with the problem of quantitatively assessing the NFR modeling process early in the project. We also performed an experimental case study in a real-live project setting to check the applicability of the solution approach. The overall conclusion from the study is that our approach seems promising. As this is an early conclusion, our immediate next step is to conduct multiple case studies in different settings to assess the usefulness of the counting technique. Only then will

we be able to provide meaningful recommendations to practicing software managers on how to scope their projects.

To the best of our knowledge, the software industry lacks quantitative and objective effort estimation methods for NFRs, and would certainly benefit from the precise and objective size measurement approach proposed in this paper. This is the motivation for two research activities to be conducted in the near future: (i) determine how the size of NFRs impacts the total project cost, and (ii) derive guidelines for how to systematically deal with NFRs which can be decomposed into FRs up to a certain level of functionality.

References

- [1] A. Finkelstein, J. Dowell, “A Comedy of Errors: The London Ambulance Service Case Study”, *Proc. 8th Int'l Workshop Software Spec and Design*, 1996, pp. 2-5.
- [2] Chung L., Nixon B. A., Yu E., Mylopoulos J., *Non-functional Requirements in Software Engineering*, Kluwer Academic Publishing, 2000.
- [3] K. K. Breitman, J. C. S. P. Leite, A. Finkelstein, “The World's Stage: A Survey on Requirements Engineering Using a Real-Life Case Study,” *Journal of the Brazilian Computer Society*, No. 1 Vol. 6, July 1999, pp. 13-37.
- [4] L. Leveson, C. S. Turner, “An Investigation of the Therac-25 Accidents,” *IEEE Computer*, 26, 7, 1993, pp. 18-41.
- [5] Mercedes A-Class: “Mercedes: Wie sicher ist die A-Klasse?” German news magazine: “Der Spiegel”, ISSN 0038-7452, October 27, 1997, p. 120; English translation: <http://www.geocities.com/MotorCity/downs/9323/aclaca.p.htm>, last visited on February 11, 2005.
- [6] Ebert, C., Dumke, R., Bundschuh, M., Schmietendorf, A., *Best Practices in Software Measurement: How to use metrics to improve project and process performance*, Springer, 1st ed., 2004.
- [7] Abran, A., Desharnais, J.-M., Oligny, S., St-Pierre, D., Symons, C.: COSMIC FFP – Measurement Manual (COSMIC Implementation Guide to ISO/IEC 19761:2003), École de technologie supérieure – Université du Québec, Montréal, Canada, (2003), URL: <http://www.gelog.etsmtl.ca/cosmic-ffp/manual.jsp>.
- [8] ISO/IEC 19761. Software Engineering: COSMIC-FFP – A functional size measurement method, International Organization for Standardization – ISO, Geneva, (2003).
- [9] Pfleeger, S. L., F. Wu, R. Lewis: Software Cost Estimation and Sizing Methods: Issues and Guidelines, RAND Corporation, 2005.
- [10] ISO 14143-1, Functional size measurement – Definitions of concepts, International Organization for Standardization – ISO, Geneva, 1988.
- [11] Albrecht, A. J. and Gaffney, J. E., Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation, *IEEE Trans. Software Eng.* vol. SE-9, no. 6, 1983, 639-648.

- [12] UKSMA, Estimating with Mark II, v.1.3.1., ISO/IEC 20968:2002(E), www.ukσμα.co.uk
- [13] NESMA, NESMA Functional Size Measurement method compliant to ISO/IEC 24570, 2006, www.nesma.nl
- [14] Stensrud, Alternative Approaches to Effort Prediction of ERP projects, Journal of Information and Software Technology, 43 (7), 2001, pp. 413-423.
- [15] Working IEEE/IFIP Conference on Software Architecture (WICSA) Feb 18-22, 2008, Vancouver, Canada, www.wicsa.org.
- [16] IFPUG 4.1 Unadjusted Functional Size Measurement Method - Counting Practices Manual, ISO/IEC 20926:2003, first edition, 2003-10-01 www.ifpug.org
- [17] Mylopoulos, J., Goal-oriented Requirements Engineering, Keynote speech at the 14th IEEE International Conference on Requirements Engineering, IEEE Computer Society Press, 2006.
- [18] Glinz, M., Rethinking the Notion of Non-Functional Requirements, Proc. of the 3rd World Congress for Software Quality, Munich, Germany, 2005.
- [19] Wieringa, R., The Declarative Problem Frame: Designing Systems that Create and Use Norms, Proc. of the 10th IEEE Int'l Workshop on Software Specification and Design, IEEE Computer Society Press, 2000, 75-85.
- [20] Ebert, C., Dumke, R., Bundschuh, M., Schmietendorf, A., *Best Practices in Software Measurement: How to use metrics to improve project and process performance*, Springer, 1st ed., 2004.
- [21] Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.
- [22] FP Users Group: www.ifpug.org

Appendix 1

Figure 5: Illustration of FR and NFR relations through the Mobile Email system case study.