Using the PR*i*M method to Evaluate Requirements Models with COSMIC-FFP

Gemma Grau, Xavier Franch

Universitat Politècnica de Catalunya (UPC)
c/ Jordi Girona 1-3, Barcelona E-08034, Spain.
{ggrau, franch}@lsi.upc.edu

Abstract. The COSMIC-FFP is a standard method that has been proven effective for measuring the functional size of business applications and real-time software systems from their functional user requirements specification. Despite of this, the methods based on COSMIC-FFP usually require a mapping between the concepts in the requirements specification and their own terms and do not take into account non-functional requirements. On the other hand, PR*i*M is a method that aims at assessing non-functional properties at the early stages of the development process. PR*i*M uses the *i** framework to model the functional and non-functional requirements in terms of actors and dependencies among them. In this paper we present how the *i** constructs proposed in PR*i*M can be adapted to measure the functional size using COSMIC-FFP and, as PR*i*M works with requirements and allows the evaluation of non-functional properties, there is a remarkable benefit when using both methods altogether.

1 Introduction

The COSMIC-FFP [1] provides a method for measuring the functional size which is supported by the ISO/IEC 19761 [11]. The functional size is measured based on the functional user requirements specification of software systems on the domains of business applications and real-time software systems. Its effectiveness has converted it into a well-establish method and many measurement procedures have arisen to support it ([2], [10], among others). However, there still some open points. First, as COSMIC-FFP aims at measuring the functional size, non-functional properties are not taken into account. Second, there is a lack of a clear definition of the various concepts that contribute to software size, particularly at the requirements modelling level [3]. Finally, as stated in [10], for productivity reasons it is important to avoid model reconstruction dedicated for just measuring purposes.

On the other hand, PR*i*M [7] is a Process Reengineering *i** Method that addresses the specification, analysis and design of information systems from a reengineering point of view. This is based on the premise that, nowadays, most of the information systems are not built from the scratch, but from a legacy system or a human process that need to be reengineered. PR*i*M uses the *i** framework [14] for representing the software model of the system in terms of actors and dependencies between them. The PR*i*M method is conformed by the six phases presented in Fig. 1. The first phase

involves capturing and recording the information about the current process in order to inform further phases. During the second phase, the *i** model of the current process is build. In order to reengineer the current process, new goals are obtained, which is done in the third phase of the method. With the aim of satisfying these goals, several process alternatives are systematically generated during the fourth phase. In the fifth phase, the different alternative *i** models are evaluated by applying structural metrics over them [6]. Finally, in the sixth phase, PR*i*M proposes the generation of the new information system specification from the *i** model of the chosen alternative.

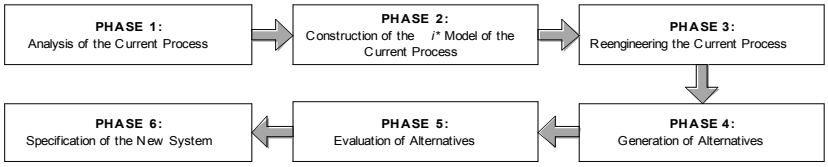

Fig 1. Overview of the PR*i*M method

The structural metrics proposed by PR*i*M focus on the evaluation of non-functional properties such as easy of use, process agility, maintainability, etc. However, despite that the *i** models are constructed taking into account the functional requirements of the system, functional metrics are not considered by the method. In [9] we propose a framework to include other techniques within a reengineering framework such as PR*i*M. In that context we found adequate to adapt the PR*i*M method to measure the functional size because we have observed strong similarities between the representation of the mappings used in COSMIC-FFP and the ones represented in PR*i*M. Thus, we propose to calculate functional size to complement the set of metrics proposed by PR*i*M, and by using the tool support provided by J-PR*i*M [8]. As the PR*i*M method provides techniques for the elicitation of requirements, the generation of design alternatives and the evaluation of non-functional properties, we believe that the use of both techniques altogether provides mutual benefits.

In order to verify that COSMIC-FFP can be correctly applied within the PR*i*M context, we have applied the measurement process steps proposed in [12], which are: 1) Design of the process method, 2) Application of the measurement method rules, 3) Analysis of the measurement result, and 4) Exploitation of the measurement result. However, due to the lack of space, here we only present some parts of the firsts two steps. For the design of the process method we focus on: 1) the mapping between the concepts of PR*i*M and the ones of the COSMIC-FFP metamodel; and, 2) on the definition of the numerical assignment rules. On the other hand, for the application of the measurement method we focus on: 1) how the software documentation is obtained and the software model is constructed using the J-PR*i*M tool; and, 2) how we apply the numerical assignment rules using the structural metrics. The proposed process is exemplified by using the C-Registration Case Study [13].

The remainder of this paper is organized as follows. In section 2 we introduce the *i** framework. As a first step for using COSMIC-FFP in PR*i*M, in section 3, we present the mapping between the modelling and evaluation concepts in both methods. In section 4 we show how the functional size is measured in PR*i*M and, in section 5, we introduce how non-functional properties can be evaluated within this method. Finally, in section 6 we present the conclusions and future work.

2 The *i** Framework

The *i** framework is a goal-oriented language defined by Eric Yu [14] with the aim of modelling and reasoning about organizational environments and their information systems. For doing so, it offers a formal representation of the involved actors and their behaviours allowing the consideration of both functional and non-functional requirements. The *i** framework proposes the use of two types of models for modelling systems, each one corresponding to a different abstraction level: a Strategic Dependency (SD) model represents the strategic level by means of the dependencies between the actors, whilst the Strategic Rationale (SR) model represents the rational level by means of showing the intentionality inside each one of the represented actors. As COSMIC-FFP takes into account the interaction between the actors rather than in its internal behaviour, in this paper we focus on SD models.

A SD model consists of a set of nodes that represent actors and a set of dependencies that represent the relationships among them, expressing that an actor (*depender*) depends on some other (*dependee*) in order to obtain some objective (*dependum*). The *dependum* is an intentional element that can belong to one of the following four types: goal, task, resource, and softgoal. The semantics are:

- For goal dependencies, the *dependee* is free to make whatever decisions are necessary to achieve the goal. For instance, in Fig. 2, the *Registar* depends on the *C-Registration System* for the goal *Student information is maintained*.
- For task dependencies, the *depender* depends upon the *dependee* to attain a goal following a prescriptive procedure. For instance, in Fig. 2, the *Professor* depends on the *C-Registration System* to *Submit student grades*, which has to be done following its own procedure.
- For resource dependencies, the *depender* depends upon a *dependee* for the availability of a physical or informational entity. For instance, in Fig. 2, the *C-Registration System* depends on the *Student* to obtain the entity *Course registration*.
- For softgoal dependencies, the *depender* depends upon the *dependee* to attain some goal, perform some task, or produce some resource, in a particular way. The *Student* depends on the *C-Registration System* for a *Secure remote access to the system*.

The graphical notation is shown in Fig. 2. For more details we refer to [14].

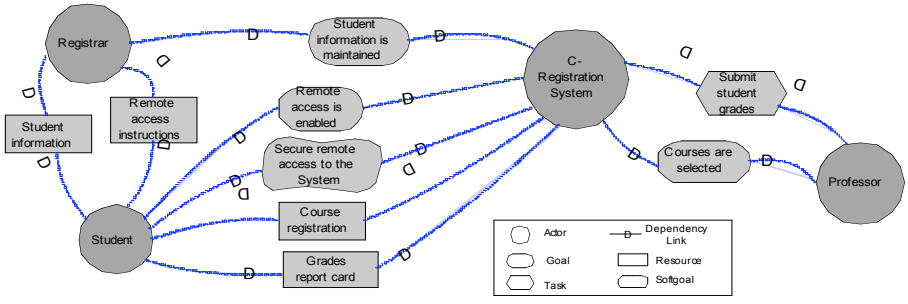

Fig.2. Excerpt of an *i** model for the C-Registration System

4

3 Mapping Phase: From COSMIC-FFP to PR*i*M

The first of the measurement process steps proposed in [12] refers to the design of the process method, which includes: 1) the definition of the objectives; 2) the characterization of the concept to be measured; 3) the design or selection of a metamodel for the object to be measured; and, 4) the definition of the numerical assignment rules. In this section we address the last two aspects, focusing on the mapping of concepts between COSMIC-FFP and PR*i*M.

The metamodel selected for representing the software model to be evaluated is the PR*i*M *i** metamodel. In PR*i*M, the *i** model is constructed in two different processes in order to differentiate the functionality that is performed by the system (Operational *i** Model) from the strategic needs of the organization (Intentional *i** Model). Thus, the Intentional *i** Model takes into account non-functional requirements and, as we are interested in the functional user requirements, here we only address the Operational *i** Model.

The Operational *i** Model is constructed based upon the information available from the current process or on the description of how the new process has to be. In order to facilitate the further construction of the model, this information is summarized into Detailed Interaction Scripts (DIS). DIS are scenario-based templates that describe the information of each activity of the current process by means of its preconditions, postconditions, triggering events, and a list of the actions undertaken in the activity. For each action, it specifies the actor that initiates the action (*initiator*), the name of the action, the resource involved (differentiating if is produced, provided, or consumed by the *initiator*) and the actor to which the action is addressed (*addressee*). PR*i*M does not enforce any scenario-based technique for filling the DIS templates and so, it is possible to apply use cases or any other scenario-based technique for documenting the current process as long as it follows the structure proposed. The reason behind is that the PR*i*M method provides precise rules that allows to transform the information on the DIS to the Operational *i** Model, which can be done automatically with appropriate tool support (i.e. J-PR*i*M [8]).

COSMIC-FFP also analyses the functional processes, and differentiates its subprocesses in order to identify the data movement implied in each. It distinguishes three types of actors: users or engineered devices, software belonging to the boundary of the system, and persistent storages. Depending on the direction of the data movement it also distinguishes between an entry, exit, read and write. Table 1 presents the definitions for this concepts and establishes and analogy with the *i** framework. We remark that in order to establish this analogy, we assume that it is possible to classify the *i** actors as belonging to User, Boundary or Persistent Storage. The COSMIC-FFP functional size is calculated by assigning to each data movement, a single unit of measure which is, by convention, equal to 1 cfsu (cosmic functional size unit). Therefore, the total size of the software being measured corresponds to the addition of all data movements recognized by the COSMIC-FFP method.

From the information on Table 1, we observe some similarites between the COSMIC-FFP measurement process model and PR*i*M. Thus, in Fig. 3, we have established a set of mapping relationships between the concepts needed in the Functional User Requirements (FUR) of the COSMIC-FFP generic software model,

Table 1. Mapping of the COSMIC-FFP concepts to the *i** Concepts

COSMIC-FFP concept (from [1])		***i**** **Concept**
User	Any person that specifies Functional User Requirements and/or interacts with the software.	Actor that represents one or more human roles that have functional dependencies over the software under study.
Boundary	A Conceptual interface between the software under study and its users.	Actor that represents the different pieces of software under study.
Persistent Storage	Storage which enables a functional process to store or retrieve data.	Actor that represents the entities that manage data in a persistent way.
Data movement	Component of a functional process that moves one or more data attributes belonging to a single data group.	Any dependency where the *dependum* is a resource.
Entry (E)	Data movement type that moves a data group from a user across the boundary into the functional process where it is required. An Entry does not update the data it moves.	*Dependum*: Resource (data group) *Depender*: Boundary (functional process) *Dependee*: User
Exit (X)	Data movement type that moves a data group from a functional process across the boundary to the user that requires it. An Exit does not read the data it moves.	*Dependum*: Resource (data group) *Depender*: User *Dependee*: Boundary (functional process)
Read (R)	Data movement type that moves a data group from persistent storage within reach of the functional process which requires it.	*Dependum*: Resource (data group) *Depender*: Boundary (functional process) *Dependee*: Persistent Storage
Write (W)	Data movement type that moves a data group lying inside a functional process to persistent storage.	*Dependum*: Recource (data group) *Dependee*: Persistent Storage *Dependee*: Boundary (functional process)

the information documented in the DIS tables of PR*i*M and the *i** concepts. At the left of Fig. 3, we observe that COSMIC-FFP is based upon a *Functional Process* which has a *Triggering Event* and several *Subprocesses* associated to it. Each *Subprocess* has a Data Group that can be of the type: entry (E), exit (X), read (R) or write (W). In the DIS, each *Functional Process* is represented by an *Activity*; the *Triggering Event* is part of the *Conditions* associated to the *Activity*; and, each *Subprocess* is

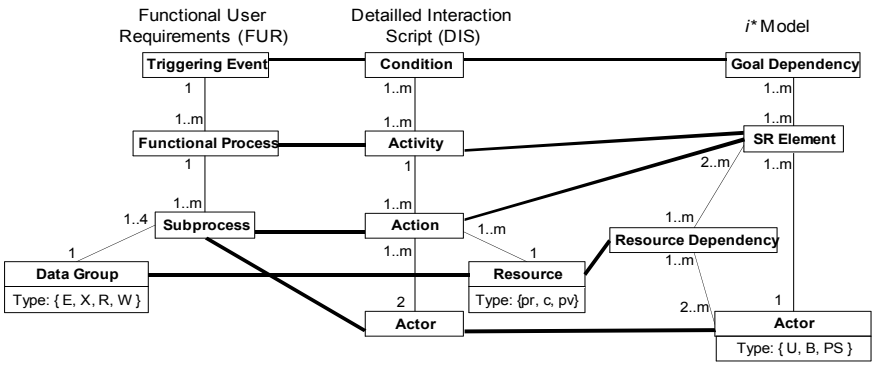

Fig. 3. Mapping across the different metamodels

represented by the concept of an *Action*. There is a correspondence between the concepts of *Data Group* and *Resource*, although the distinction between the *Data Group* types is implicit in the DIS information because it depends on the *Actors* that participate in the action. As we have already mentioned, PR*i*M proposes a set of automatic rules to transform DIS into *i** Models (see [7] for details), where *Conditions* are transformed into *Goal Dependencies*, *Activities* and *Actions* are represented into SR elements, and *Resource Dependencies* are established between the different *Actors*. In order to help the evaluation of the *i** Model with COSMIC, we propose a classification of the *i** actors into the following types: user (U), boundary (B), and persistent storage (PS).

The PR*i*M method proposes structural metrics for evaluating the *i** models and, in order to apply COSMIC-FFP, the numerical assignment rules are defined using the formulas and concepts proposed in [6], [7] which differentiate between actor-based and dependency-based functions. As in COSMIC-FFP the unit of measurement are the Data Groups, and in *i** Data Groups are represented as dependencies. We are interested in dependency-based functions, which can be defined as follows.

Given a property P and an *i** SD model that represents a system model M = (A, D), where A is the set of the actors and D the dependencies among them, a dependency-based architectural metric for P over M is of the form:

$$P(M) = \frac{\Sigma d{:}\ d(\mathrm{a,b}) \in \mathrm{D}{:}\ \mathrm{filter}_M(d) \times \mathrm{correctionFactor}_M(a,b)}{\mathrm{limit}_P(\mathrm{D})}$$

being filter_M: D→[0,1] a function that assigns a weight to the every *dependum* (e.g., if the *dependum* is goal, resource, task, softgoal if it is from a specific kind), and $\mathrm{correctionFactor}_M$: A→[0,1] a function that correct the weight accordingly to the kind of actor that the *depender* and the *dependee* are, respectively. Finally, limit_P(D): A→ [1, ||A||] is a function that normalizes the result obtained.

In order to measure the functional size, we have adjusted these factors according to the criteria established in Table 1. As we are interested in counting the total amount of cfsu, we do not apply the normalization value and $\mathrm{limit}_P(\mathrm{D}) = 1$. Therefore, we define the metric as:

$$\text{Functional Size (M)} = \Sigma \mathrm{d}{:}\ \mathrm{d} \in \mathrm{D}{:}\ \text{functional\_size(d)}$$

Where,

$$\mathrm{filter}_M(d) = \begin{cases} 1, \text{ if } d \in \text{Resource} \\ 0, \text{ otherwise} \end{cases}$$

$$\mathrm{correctionFactor}_M(a,b) = \begin{cases} 1, \text{ if } a \in \text{Boundary and } b \in \text{User} \\ 1, \text{ if } a \in \text{User and } b \in \text{Software} \\ 1, \text{ if } a \in \text{Boundary and } b \in \text{Persistent Storage} \\ 1, \text{ if } a \in \text{Persistent Storage and } b \in \text{Boundary} \\ 0, \text{ otherwise} \end{cases}$$

$$\mathrm{limit}_P(\mathrm{D}) = 1$$

4 Measurement Phase: Evaluating COSMIC-FFP with PR*i*M

Once the measurement method has been designed, the measurement process presented in [12] proposes three steps for its application: 1) software documentation gathering

and construction of the software model; 2) application of the numerical assignment rules; and 3) measurement result analysis and exploitation of the result.

As the final purpose of the measurement phase is to ensure that the defined mapping rules and the numerical assignment rules allow a reliable calculation of the functional size, we have apply the method to the three case studies presented at [5], and compliant to ISO 19761 [11]. For doing it, we have introduced the case studies in our tool J-PR*i*M [8], and we have obtained positive results with all of them. To illustrate the process, we present the C-Registration Case Study [13].

For the software documentation gathering, we have used the information provided in the case study (that is, the problem statement and the rules provided by COSMIC-FFP for establishing the Application Boundary, the Data Groups and the Functional Processes) in order to ensure that we were working with the same elements of the case study. Therefore, based on the provided Functional Processes we have established the activities of PR*i*M and, for each activity, we have described its actions by filling the DIS template. We remark that, in order to be compliant with the method, when describing the actions we have made explicit those actions that involve storing or retrieving information from the Persistent Storage. Although this was not considered in PR*i*M, it has been possible to introduce this information correctly from the problem statement provided. Fig. 4 shows the screenshot of J-PR*i*M when defining the activities for the C-Registration System. At the top we have the defined activities (at the left, the list of the activities and, at the right, its individual information). At the bottom we have the DIS template showing, for each action, the actor *initiator*, the resource involved (which can be consumed, produced or provided), and the actor *addressee*. At the left of Fig. 4 it is possible to see the representation of the Operational *i** Model for the C-Registration System, which is generated automatically from the introduced information.

Once the Operational *i** Model has been generated, we have applied the COSMIC-FFP dependency-based metric as defined in the previous section. Fig. 5 presents the screenshot of the graphical interface that we have added in PR*i*M to facilitate the application of COSMIC-FFP. At the top-left side we show the different alternatives that can be evaluated, as the Operational *i** Model contains all the information for calculating the functional size it is the one that has been selected. However, we remark that other Alternative *i** Models could be generated according to the guidelines proposed in the PR*i*M method [7]. At the top-right side we have three boxes for classifying the actors according to the boundaries of the system (Users, Application Boundary, and Persistent Storage). Finally, at the bottom, we can see the evaluation of each activity (or functional process), whilst the overall result is presented at the end of the list.

Once the result has been calculated, we have checked it with the result obtained in the case study. We have to mention that the first results were different from expected. In the first execution the reason was that, as we wanted to avoid copying the functional process data movements, we have generated our own action description. In this description we considered the command introduced by the user as a triggering event, whilst in some functional processes of the case study this is considered as a data movement. We have added these data movements and we have regenerated the Operational *i** Model. In the second execution, the final result is 107 cfsu, one unit

8

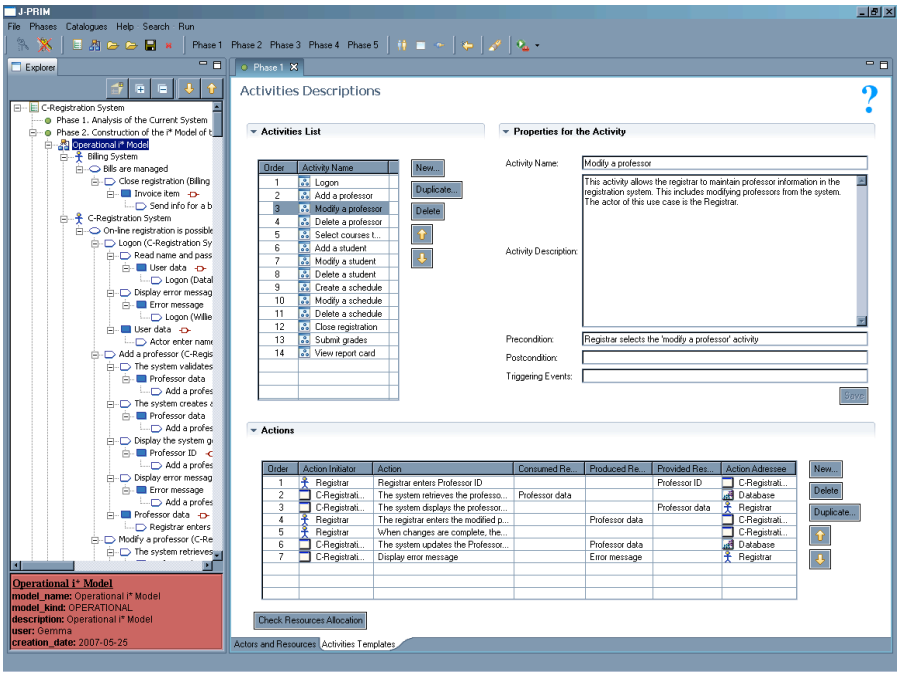

Fig 4. Screenshot of J-PRiM: Activity definition and action description using the DIS template

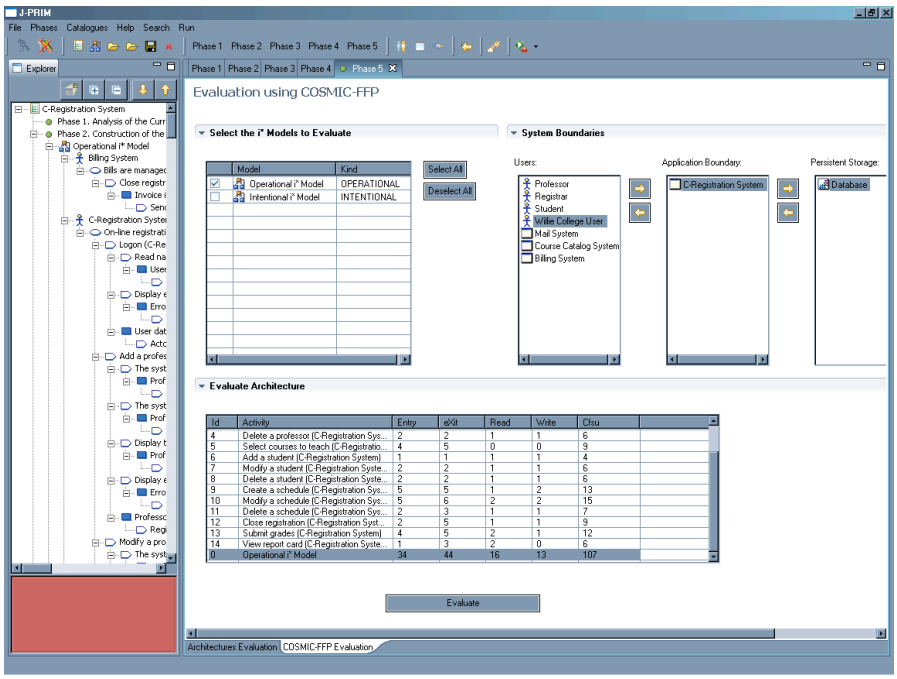

Fig 5. Screenshot of J-PRiM: Evaluation of COSMIC-FFP for the selected Operational *i** Model

higher than expected, which is due to a miscalculation on the final results presented in the C-Registration case study which scores 106 cfsu.

5 Non-functional Measurements with PR*i*M

As we have previously mentioned, PR*i*M is a reengineering method that supports the generation and evaluation of alternatives. In PR*i*M the generation of alternatives is done based on the *i** model of the current process (namely the Operational *i** Model) and by reallocating the responsibilities between its actors. For instance, in the C-Registration Case Study, we can consider two different alternatives: A) In the current situation as described in the C-Registration Case Study, the responsibility of modifying the student and the professor information falls on the *Registrar* actor. In the this alternative we reallocate the responsibility of modify the student information onto the *Student* actor, and the one of modify the professor data onto the *Professor* actor; B) Also in the current situation, the responsibility of accessing the system falls to different actors. As an alternative, we can make it completely fall onto the *Registrar* and, so, we consider that the student and the professor always request him for introducing their data.

The PR*i*M method supports the generation of alternatives and their evaluation using structural metrics. The metrics proposed in PR*i*M can be organizational (dealing with non-functional properties of the organization) or architectural (dealing with non-functional properties of the software). For instance, in Table 2 the Operational *i** Model and the two alternatives proposed are evaluated regarding to the organizational properties: *Ease of communication* and *Process agility*, as they are defined in [7]. When defining *Ease of communication*, we consider that the communication is easier when the two interacting actors are human, and it is damaged when it involves the software system. Under these premises, the results obtained for the C-Registration System are as expected because the value of *Ease of communication* increases when the Registrar gets more responsibilities and the Professor and the Student have first to communicate with him to access the system. On the other hand, for *Process agility* we consider that software actors are more agile than human actors and, thus, its value decreases as the Professor and the Student depends on the Registrar, as this human intermediate actor makes the process less agile. Therefore, despite the three alternatives score the same functional size, they provide different non-functional properties. We remark that, although this is not the case in the example, alternatives showing different ways of interacting with the software boundary could lead to different results of cfsu.

Table 2. Evaluation of the Alternatives

Alternative	Cfsu	Ease of Communication	Process Agility
Operational *i Model**	107	0.2824	0.7903
Alternative A	107	0.2796	0.8040
Alternative B	107	0.3761	0.6351

6 Conclusions and Future Work

PR*i*M is a process reengineering method that aims at assessing non-functional properties of the system using an *i** requirements model. In order to provide PR*i*M with functional size measurement capabilities, we have adapted the COSMIC-FFP measurement process model to PR*i*M, and we have checked that the measurement results are correct by replicating existing case studies. We have also used these case studies to generate alternatives and evaluate their non-functional properties with the structural metrics proposed in PR*i*M.

Based on the results obtained so far, we argue that both methods benefit from this process. On the one hand, COSMIC-FFP provides PR*i*M with a standardized measurement process for evaluating the functional size, that has been already validated and it is currently used in a wide variety of situations and domains [4]. Because of that, COSMIC-FFP also provides knowledge and experience for calculating the functional size. For instance, the questions for validating that a candidate process is a COSMIC-FFP functional process [1], or the guidelines for the identification of the entry data movement type provided in [3].

On the other hand, PR*i*M provides COSMIC-FFP with the possibility of using a unique *i** requirements model for representing both functional and non-functional requirements, as well as techniques for gathering the requirements. It also provides guidelines for generating alternatives and structural metrics for evaluating non-functional properties. All these activities can be undertaken by using *i** models, which avoid having more than one representation, and can be integrated into the development process because it is possible to generate the use cases specification of the information system from the resulting *i** model.

According to the usability of our proposal, the *i** classification that we present, indicating which actor is a user, belongs to the boundary or represents the persistent storage, makes implicit which dependencies are an entry, an exit, a read or a write, facilitating the implicit classification of the Data Groups. Finally, as we have included COSMIC-FFP in the tool J-PR*i*M, we already have tool-support for generating and evaluating the models, making possible to define variations over the COSMIC-FFP evaluation formula with little effort.

As future work we will address other metrics based on the functional size, such as the ones for assessing software product lines. As Data Groups are often obtained from class diagrams, we plan to study how to get the class diagrams in addition to the use cases in PR*i*M.

Acknowledgements. This work has been partially supported by the CICYT programme project TIN2004-07461-C02-01. Gemma Grau work is supported by an UPC research scholarship.

References

1. Abran, A., Desharnais, J.M., Oligny, S., St-Pierre, D., Symons, C.: “COSMIC-FFP Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761:2003)”.

Version 2.2, Common Software Measurement International Consortium, 2003. Available at: http://www.lrgl.uqam.ca/cosmic-ffp/.

2. Condori-Fernández, N., Abrahão, S., Pastor, O.: "Towards a Functional Size Measure for Object-Oriented Systems from Requirements Specifications". In *Proceedings of the 4th International Conference on Quality Software*, QSIC 2004. pp. 94-101.
3. Condori-Fernández, N., Pastor, O.: "Evaluating the Productivity and Reproducibility of a Measurement Procedure". In *Proceedings of the 2nd International Workshop on Quality of Information Systems*, QoIS 2006. pp: 352-361.
4. The COSMIC-FFP at: http://www.cosmicon.com. Last visited: September 2007.
5. The COSMIC-FFP at: http://www.lrgl.uqam.ca/cosmic-ffp/. Last visited: September 2007.
6. Franch, X., Grau, G., Quer, C.: "A Framework for the Definition of Metrics for Actor-Dependency Models". In *Proceedings of the 12th IEEE International Conference on Requirements Engineering*, RE 2004. pp: 348-349.
7. Grau, G., Franch, X., Maiden, N.A.M.: "PR*i*M: an *i**-based process reengineering method for information systems specification". To appear in Information and Software Technology.
8. Grau, G., Franch, X.: "ReeF: Defining a Customizable Reengineering Framework". To appear in *Proceedings of the 19th International Conference on Advanced Information Systems Engineering,* CAiSE 2007. Springer-Verlag, LNCS 4495. pp. 485-500.
9. Grau, G., Franch, X., Ávila, S.: "J-PR*i*M: A Java Tool for a Process Reengineering *i** Methodology". In *Proceedings of the 14th IEEE International Conference on Requirements Engineering*, RE 2006. pp. 352-353.
10. Habela, P., Glowacki E., Serafinski T., Subieta K.: "Adapting Use Case Model for COSMIC-FFP Based Measurement". In *Proceedings of the 15th International Workshop on SoftwareMeasurement*, IWSM 2005. pp. 195-207.
11. ISO/IEC 19761:2003. "Software Engineering – COSMIC-FFP – A functional size measurement method", International Organization for standarization, 2203.
12. Jacquet, J.P., Abran, A.: "From Software Metrics to Software Measurement Methods: A Process Models". In *Proceedings of the 3rd International Software Engineering Standards Symposium*, ISESS'97. pp. 128-135.
13. Khelifi, A., Abran, A., Symons, C., Desharnais, J.M., Machado, F., Jayakumar, J., Leterthuis, A.: "The C-Registration System Case Study with ISO 19761 (2003)". Available at: http://www.gelog.etsmtl.ca/cosmic-ffp/casestudies/. Last visited: September 2007.
14. Yu, E.: *Modelling Strategic Relationships for Process Reengineering*. PhD. thesis, University of Toronto, 1995.