Approximate COSMIC Size: the Quick/Early Method

Gabriele De Vito
Cid Software Studio
Napoli, Italy
gdevito@cidsoftware.it

Filomena Ferrucci
Department of Management and Information Technology
University of Salerno, Italy
fferrucci@unisa.it

***Abstract*—The COSMIC method is based on the counting of data movements from/to persistent storage and users. This requires some kind of data analysis to identify data groups. Project managers often need functional size at the beginning of the project life cycle but usually data analysis is not performed in the requirements elicitation phase. Moreover, rapid sizing can be requested since there is insufficient time or resources to apply the standard detailed method. Thus, there is the need of a simplified and rapid COSMIC measurement method to be applied to requirements elicitation documents. Such a method should avoid the use of scale factors since incorrect calibrations of the scale factors can lead to inaccurate approximations. To address the problem we proposed a simplified measurement process (named Quick/Early) that can be applied on the use case models and aims to reduce the measurement time, reaching a trade-off between an accurate measurement and time available. To support the measurement process we proposed a template for use cases that makes the functional components immediately identifiable, i.e., the sequences of events. The application of the method on a case study showed a good accuracy.**

Keywords— Functional Size Measurement; COSMIC; Simplified measurement processes, Use Cases

I. INTRODUCTION

Functional Size Measurement (FSM) methods have been widely applied for software sizing in order to estimate and control development costs and performance and to plan schedules [3][4]. Among the appealing features of FSMs is their independence from the development decisions taken in the operational artifacts of the software to be measured. In particular, many organizations are adopting COSMIC that is considered a 2nd generation FSM since it addresses some limitations of previous methods, indeed it was the first FSM method developed to conform to the standard ISO/IEC14143/1, it is based on fundamental principles of software engineering and measurement theory, and it is applicable to a wide range of software types (business, real-time, and infrastructure or hybrids of these).

The basic idea underlying COSMIC is that the number of data movements from/to persistent storage and users can provide a meaningful sight of the system size. Thus, the COSMIC size of software consists of the number of those data movements. To precisely identify all data movements measurer has to identify the data groups thus performing some kind of data analysis. This is time consuming and requires detailed functional specifications that normally are not available in the requirements elicitation phase but become complete in the later stage of a software project. In practice, however, project managers often need functional size at the beginning of the project life cycle. In addition, due to limited time and resources they need rapid sizing without using all the required details to apply COSMIC.

To define a simplified COSMIC measurement to be applied to requirements elicitation documents we need to identify a software "functional concept", that can be identified in the available documentation and is characterized by a sufficiently high level of abstraction, but not enough to affect the accuracy of the result.

Simplified COSMIC measurement processes have been proposed (see for instance the section on “early or rapid approximate sizing” in [2]) identifying the Functional Process as the "functional concept" to which assign a scale factor, representing the functional size when FURs are sufficiently detailed to allow for an accurate measurement. However, the main critical aspect of this approach is that incorrect calibrations of the scale factor lead to inaccurate approximations. The aim of the paper, therefore, is to define an approximate COSMIC approach, named the Quick/Early method that does not require scale factors, but only the availability of appropriately documented use cases. In fact, use cases are widely used to model software systems, because represent the system requirements in terms of functionality (FURs) and environment (actors) and can be modeled right from the beginning phase of requirements elicitation. In the paper, we introduce a template to appropriately document use cases and a method to quickly size them.

The remainder of the paper is organized as follows. Section II overviews the COSMIC method and its simplified versions, showing limitations and difficulties with using them. Section III describes some characteristics of use cases and illustrates the use cases model template that supports the simplified measurement process. Section IV describes the Quick/Early method. Section V shows some examples. Section VI accounts for related work. While Section VII draws some conclusions and outlines future work.

II. COSMIC AND ITS MAIN SIMPLIFIED APPROACHES

A. Overview of COSMIC and difficulties with using it

The COSMIC [1] method is based on the application of a set of models, rules, and procedures to a given piece of software, as it is defined from its Functional User Requirements (FURs). The COSMIC method requires that:

1. The functional processes of the software being measured be identified;
2. The data groups mentioned in the user requirements be identified;
3. For each functional process, the unique data movements involving each identified data group be counted. Data movements are classified into Entries and Exits (i.e., I/O movements) and Reads and Writes (to persistent storage). The sum of data movements determines the size expressed as *CFP*.

To recognize objects of interest and data groups, therefore, it is almost inevitable some kind of data analysis, namely: noun/verb, class, responsibility and collaborators (CRC), entity/relationship, class diagrams and relational data analysis to identify conceptual classes and relationships among them, activity diagrams and sequence diagrams analysis to identify the data flows in a functional process, etc.. These techniques are usually performed to understand the structure of the persistent data and, therefore, to identify objects of interest related to Read and Write data movements. However, consistent with the aims of the COSMIC method, we need to apply the data analysis principles to the structures of the transient data to identify objects of interest related to Entry and Exit data movements. This means analyzing the available software documents which describe the data flow, and, in some cases (especially if the identified data structures are not normalized), reapply relational data analysis. Consequently, performing this kind of analysis is an expensive and lengthy process. Another challenge of the COSMIC method is the accurate identification of all COSMIC concepts. This is especially when the software project has a single layer (as understood in the COSMIC manual [1]), or the measurement scope is the entire project. Based on these assumptions, we aim to avoid special Data Analysis, and, therefore, to simplify the COSMIC standard process merely identifying what is strictly required.

B. Simplified COSMIC methods

Four simplified COSMIC measurement processes have been proposed by COSMIC Consortium [2] to early or rapid approximate sizing, namely: *Average Functional Process* (*AFP*), *Fixed Size Classification* (*FSC*), *Equal Size Bands* (*ESB*) and *Use Case and Average* (*AUC*). These versions identify the functional process as the "*concept*" (at a level of abstraction higher than the data movements) to which you assign a scale factor that is the expected functional size when FURs are sufficiently detailed to allow an accurate measurement. It is worth noting that functional processes have at least two data movements and theoretically no upper bound. If on one hand the latter feature is an advantage compared with other FSMs (especially for "*data intensive*" software) on the other hand, this means that there is no average functional process from which we can obtain a scale factor to measure software in the early phase. But the accuracy of scale factors strongly depends on the local calibration process. This aspect represents the main critical aspect of these simplified methods.

Thus the aim of the paper is to propose a simplified/approximate COSMIC method that does not require a local calibration process either a data analysis (because it is time consuming and needs qualified resources) and is based on the information that is available at the requirements elicitation phase (use case models). This means we cannot precisely identify data groups and then the types of data movements. But if we focus on events and actions (in terms of data, parameters or variable) involving the functional user, the system and the external components, we can identify data movements as incoming and outgoing data, parameters or variables. Obviously, this might determine missing or wrongly added data movements, in particular when they involve external components included (or not) in the measurement scope. In general, however, counting these kinds of movements can result in a sufficiently accurate functional size estimation.

III. USE CASES TEMPLATE

Use cases allow to model requirements, especially behavioral or functional requirements splitting them into consistent units named use cases, having a meaning for actors. There is no UML standard about formulating use cases. However, we can outline a use case as shown in Table I. This template has a dual aim: the first one is to define the "*modus operandi*" to properly document a use case model; the second one is to implement a tool to support more adequately and specifically the COSMIC measurement.

TABLE I. OUTLINE OF A USE CASE

Preamble		
Use case	<name>	Starts with verb
Actors	<actor1, actor2,...>	Show who (user, external system...) interacts with the system by requiring services and who is interested in this use case and what he wants.
Preconditions	<conditions>	Show what must be true at the beginning of the use case.
Extension point	<condition, UCE>	Indicates that this Use Case extends use case *UCE* when "*condition*" is true.
Generalization of	UCG	Indicates that *UCG* is the parent of this use case.
Main scenario		
<1.>	<actor|system><action>|<include><use case UCx>	
....		
<n.>	<actor|system><action>|<include><use case UCx>	
Alternative scenario n. x <Event, starting on step k>		
<k(a).>	<actor|system><action>|<include><use case UCx>	
...		
Error scenario n. x <Event, starting on step e>		
<e(a).>	<actor|system><action>|<include><use case UCx>	
...		

A use case is composed of a preamble and some scenarios. The preamble contains general information about the use case, namely ***name***, ***actors***, ***preconditions***, ***extension point***, ***and***

generalization of. As for generalization there are some options for the inheritance shown in Table II

TABLE II. SYNTAX FOR OPTIONS IN USE CASES

Option	Example
Inherits without changes	3. (3.) The client types the search criteria
Inherits and renumbers	6.2(6.1) The system informs the customer that did not found products that meet the specified criteria
Inherits and redefines	1.(o1.) The customer selects the button labeled "product lookup"
Inherits, redefines and renumbers	5.2 (o5.1) The system displays a page showing no more than five books
Adds new behaviour	6.3 The system redisplays the search page titled "book lookup"

The scenarios show the action sequences (events) described by the use case. Each sequence step should have the following structure:

<n.><actor|system><action>|<include><UCx>

where

- ***n.*** is the order of the step within the sequence;
- ***the action event*** is "*what is happening*" and is structured as follows:

  <actor|system><action>

  where the ***actor*** or ***system*** performs a specified action or function,

  otherwise, ***<include><UCx>***

  which ***includes*** the behavior of another use case named ***UCx***. This allows to "*reuse*" an existing use case, avoiding rewriting unnecessarily actions sequences (events). When included use case ends, control will return back to the "*caller*" use case.

Each use case can be documented by three kinds of scenarios:

- ***Main scenario(MS)***, which realizes the use case postconditions in a successful way;
- ***one (or more) Alternative scenario (AS)***, which performs the use case postconditions, but in an unusual way, branching or interrupting the main sequence of events; this kind of scenario is an extension of another scenario (typically the main scenario) and is positioned at points named extension points within the main sequence of events;
- ***one (or more) Error scenario (ES)***, which doesn't perform the use case postconditions and is positioned at points named extension points within the main sequence of events.

Anyway, the MS scenario must be performed independently from the AS and ES scenarios. On the other hand, the MS scenario can specify alternative action sequences (AS) to be performed if one of its actions cannot be performed. The AS scenario is performed only once the triggering event occurs, then the MS scenario resumes its execution. In addition, if an error occurs, the MS execution will be stopped and the necessary ES scenario will be performed.

IV. THE QUICK/EARLY METHOD

The template described in the previous section shows the functional components immediately identifiable by a use case, i.e., the sequences of events. Events within a use case involve the functional user, the system and the external components (for instance persistent storage) in terms of actions and preconditions, i.e., performed functions. In the event that these functions require parameters/input or output data, we can reasonably assert that we have a "*data movement*". Take for example the sequence: "*The user asks the system to display his personal data*". The action, i.e., the function, performed by the user is: "*asks to display*". This function, however, requires some parameters to be performed (i.e., "*his personal data*"), which are "*what*" the system will process. In other words, the functional user "*move data*" to the system (i.e., "*his data*"). Conversely, let us examine the sequence: "*the system displays a confirmation dialog to the user*". The action performed by the system is "*displays a confirmation dialog*". In this case, however, the function performed doesn't require parameters, and then there are no "data movements" from the system to the functional user.

Therefore, it is possible to simplify the COSMIC standard process, analyzing only "functions" related to the sequences of events and preconditions in the use cases models. Finally, to avoid redundancy, it is sufficient to analyze "*include*", "*extend*" and "*generalize*" relationships between use cases:

- counting included use cases only once,
- counting extended use cases only when referenced in the preamble,
- counting generalized use cases only if referenced and counting only overridden or added action sequences.

The obtained result is a functional size less accurate, but more rapid to be performed, whose precision is directly proportional to the level of granularity of the analyzed use cases model. By a formal point of view, to compute the functional size of use case ***U***, we proceed as follows:

$$FSM(U) = FSM_{SEQ}(PC) + FSM_{SCEN}(MS) + \sum FSM_{SCEN}(AS_i) + \sum FSM_{SCEN}(ES_i) \quad (1)$$

where

- ***PC*** are the preconditions in the use case *U* preamble;
- ***MS*** is the main scenario;
- AS_i is the ith alternative scenario;
- ES_i is the ith error scenario;
- $FSM_{SEQ}(A)$ is the counting function of the action sequences ***A***, and matches the incoming and outgoing

number of data groups, parameters or variables needed to perform ***A***:

- from the user to the system,
- from the system to the persistent storage or to an external component,
- from an external component to the system.

- $FSM_{SCEN}(S) = \sum FSM_{SEQ}(A_i)$ where A_i is the ith action sequence (event) in the scenario S.

In addition, error messages/notifications from the system to the user "*should*" be counted once and only once, unless they do not contain data or parameters "*computed*" starting from other objects of interest. ***FMS(U)*** must result in at least 2 CFP, according the COSMIC manual [1].

V. EXAMPLE

To illustrate the application of the approach, we apply it on some sample use cases shown in Tables III-V.

TABLE III. USE CASE EXAMPLE : CLOSE REGISTRATION

Use case	Close Registration
Actors	Registrar
Preconditions	The Registrar selects the "close registration" activity from the Main Form.
Extension point	N/A
Generalization of	N/A
Main scenario (MP)	
1. The Registrar selects "Close Registration" command. 2. The system checks to see if registration is still in progress. 3. For each opened course offering, the system checks if three students have registered and at least one professor has signed up to teach the course offering. 4. The system closes the course offering. 5. For each student enrolled in the cancelled course offering, the system will modify the student's schedule. The first available alternate course selection will be substituted for the cancelled course offering. If no alternates are available, then no substitution will be made. 6. The system will notify all students, by mail, of any changes to their schedule. 7. The system sends a transaction to the billing system for each student enrolled in the course offering.	
Error scenario (ES1) <Registration in progress, 3>	
3(a). The system displays an error Message.	
Error scenario (ES2) < Less Than Three Students in the Course Offering, 4>	
4(b). The system displays an error Message.	
Error scenario (ES3) <No professor for the course offering, 4>	
4(c). The system displays an error Message.	
Error scenario (ES4) <Billing System Unavailable, 4>	
4(d). The system displays an error message after retrying transactions.	

TABLE IV. USE CASE EXAMPLE : ENQUIRE ON A SCHEDULE

Use case	Enquire on a schedule
Actors	Student
Preconditions	The Student selects the "maintain schedule" activity in the Main Form.
Extension point	N/A
Generalization of	N/A
Main scenario (MP)	
1. The Student selects "Enquire on my schedule". 2. The system retrieves the Student's current schedule. 3. The system displays the Student's current schedule.	
Error scenario (ES1) <Course Registration Closed, 3>	
3(b). The system displays an error message.	
Error scenario (ES2) <No schedule Found, 3>	
3(b). The system displays an error message.	

TABLE V. USE CASE EXAMPLE : DELETE A SCHEDULE

Use case	Delete a Schedule
Actors	Student
Preconditions	The Student selects the "maintain schedule" activity in the Main Form.
Extension point	N/A
Generalization of	N/A
Main scenario (MP)	
1. Include "Enquire on a schedule". 2. The Student selects "Delete my schedule" command. 3. The system prompts the Student to verify the deletion. 4. The student confirms the deletion. 5. The system deletes the schedule. 6. The system displays a notification message.	
Error scenario (ES1) <Course Registration Closed, 3>	
3(a). The system displays an error message.	

A. Applying the Quick/Early method

Let's apply now Quick/Early, using the rules described in section IV.

1) Functional size of "Close Registration" use case

The preconditions in the preamble require no data or parameter to be validated. In fact, the user selects a simple link, therefore $FSM_{SEQ}(PC) = 0$. In addition, there is no Alternative scenario, then $\sum FSM_{SCEN}(AS_i) = 0$.

Let's analyze the Main scenario (MP). Step 1 has the form: <actor|system><action> in which *action* needs some kind of data/parameters to be performed, going from the actor to the system: *the current registration data*. Obviously here we are not interested to know what kind of entity is involved or what kind of data movements, but only if exists a data movement. Therefore, $FSM_{SEQ}(A_1) = 1$. The action in step 2 needs to move data (*current registration data*) from the persistent storage to the system, therefore, $FSM_{SEQ}(A_2) = 1$. Action in step 3 requires data (*course offerings list*) from the persistent storage using Course Catalog System, then $FSM_{SEQ}(A_3) = 1$. In these

situations, a more formal measurement shows that being Course Catalog System external to the measurement scope, we should count two more data movements (1 Exit and 1 Entry). The action in step 4 needs to store data in the persistent storage (i.e. *Course offerings*), then $FSM_{SEQ}(A_4)$ =1. Likewise, the action in step 5 requires to update data in the persistent storage (i.e. *Schedule item data*), therefore $FSM_{SEQ}(A_5)$ =1. The action in step 6 needs to sent data by email (i.e. *Schedule item data*), then $FSM_{SEQ}(A_6)$ =1. Finally, action in step 7 need data to be performed (i.e. *Invoice item* to send to the Billing System), then $FSM_{SEQ}(A_7)$ =1. To summarize, $FSM_{SCEN}(MS) = 7$.

Let's analyze the Error scenarios. In this case, we have four Error scenarios, in which each *action* is an error notification. Therefore, $FSM_{SEQ}(A_{3a}) = FSM_{SEQ}(A_{4b}) = FSM_{SEQ}(A_{4c}) = FSM_{SEQ}(A_{4c}) = 1$. But, keeping in mind that messages and notifications need to be counted only once, then $FSM_{SEQ}(A_{3a})$ =1, while $FSM_{SEQ}(A_{4b}) = FSM_{SEQ}(A_{4c}) = FSM_{SEQ}(A_{4c}) = 0$. Therefore, applying (1):

$$FSM("Close\ Registration") = 0 + 7 + 0 + 1 = 8.$$

2) Functional size of "Enquire on schedule" use case

Likewise the use case above, the preamble preconditions, verify a simple link selection, therefore there are no data or parameters required, then $FSM_{SEQ}(PC) = 0$. In addition, there is no Alternative scenario, then $\sum FSM_{SCEN}(AS_i) = 0$.

Let's analyze the Main scenario (MP). Step 1 has the form: <actor|system><action> in which *action* needs some kind of data/parameters to be performed, going from the actor to the system (for instance *student data*). In fact, student wants the system enquires on ***his schedule***. Therefore, $FSM_{SEQ}(A_1)$ =1. The action to be performed in step 2 moves data from the system to the persistent storage (i.e., *schedule item data*), therefore, $FSM_{SEQ}(A_2)$ =1. Finally, the last step shows data to the user ("*schedule item data*"), then $FSM_{SEQ}(A_3)$ =1. To summarize, $FSM_{SCEN}(MS) = 3$.

Let's analyze the Error scenarios. In this case, we have two Error scenarios, in which step 3(a) and 3(b) are error message notifications. Therefore, $FSM_{SEQ}(A_{3a}) = FSM_{SEQ}(A_{3b}) = 1$. But, messages and notifications must be counted only once, this implies that $\sum FSM_{SCEN}(ES_i) = 1 + 0 = 1$. Then, applying (1):

$$FSM("Enquire\ on\ a\ schedule") = 0 + 3 + 0 + 1 = 4.$$

3) Functional size of "Delete a schedule" use case

The preamble preconditions require no data to be verified (the user selects a simple link), consequently $FSM_{SEQ}(PC) = 0$. In addition, there is no Alternative scenario, then $\sum FSM_{SCEN}(AS_i) = 0$.

Let's analyze the Main scenario (MP). Step 1 has the form <include><UCx>, and *UCx (use case "Enquire on a schedule")* was already measured, then $FSM_{SEQ}(A_1)$ =0. In step 2 the action to be performed needs to move data ("*student data*"), therefore, $FSM_{SEQ}(A_2)$ =1. Actions in steps 3 and 4 need no data to be executed because the first action is a confirmation request, and the second one is a repetition of action in step 2 (already counted). Therefore, $FSM_{SEQ}(A_3) = FSM_{SEQ}(A_4) = 0$. In step 5 actions to be performed need some kind of data/parameters, from the persistent storage ("*schedule item data*"), therefore, $FSM_{SEQ}(A_5)$ =1. Finally, the action in the last step needs to move some data (i.e., *notification message*) from the system to the student, then $FSM_{SEQ}(A_6)$ =1. To summarize, $FSM_{SCEN}(MS) = 3$.

Let's analyze the Error scenarios. In this case, we have only one Error scenario, having only one step: 3(a) in which *action* is a notification, then a data movement. This implies that, $FSM_{SEQ}(A_{3a})$ =1. But messages (see action A_6) must be counted only once, therefore $FSM_{SEQ}(A_{3a})$ =0. Then, applying (1):

$$FSM("Delete\ a\ schedule") = 0 + 3 + 0 + 0 = 3.$$

Summarizing, the total functional size of all FURs expressed in the uses cases described above is 8+4+3= 15 CFP.

B. Applying the COSMIC method

We apply the standard COSMIC method to highlight possible differences with the proposed approximation. We need to perform the mapping phase to identify all objects of interest, data groups and then data movements in term of Entry, Exit, Read, and Write. Applying data analysis we can identify the objects of interest and data groups listed in Table VI.

TABLE VI. LIST OF OBJECTS OF INTEREST AND DATA GROUPS

Object of interest	Data groups
User	User Data
Student	Student data
Schedule item	Schedule item data, Invoice item, Student schedule changes message
System	Command
Message	Message
Course offering	Course offering data

For all functional processes in the previous steps, all data movements of a data group must be identified to obtain the results reported in Tables VII-IX giving a total of 16 CFP, only 1 more CFP compared to the value obtained by the Quick/Early method. This difference is due to the data movements exchanged between the Course Catalog System and the system to be measured. In fact, it is external to the measurement scope, therefore we must add two more data movements: 1 Exit going from the system to the Course Catalog, and 1 Entry going from the Course Catalog to system.

TABLE VII. "ENQUIRE ON A SCHEDULE" DATA MOVEMENTS

Sub-process description	Data group	Data movement Type	CFP
Student enters 'enquire on a schedule' command	'Enquire' command	E	1
The system retrieves the Student's current schedule	Schedule Item data	R	1
The system display the Student's current schedule	Schedule Item data	X	1
Display error message	Message	X	1
			4

TABLE VIII. "CLOSE REGISTRATION" DATA MOVEMENTS

Sub-process description	Data group	Data movement Type	CFP
Registrar enters 'Close Registration'	Close command	E	1
Read if a Registration is in progress	System	Not enough details	
Display error message	Message	X	1
Obtain Course offering data from the Course Catalog	Course offering	1 x X 1 x E	2
Read Schedule items to obtain the course selections	Schedule item data	R	1
Check that at least three students signed up: if not cancel course and check alternatives	Data manipulation	N/A	
Send info for a billing transaction for each student accepted for each course	Invoice item	X	1
Update the course offerings on the Course Catalog	Course offering data	X	1
Update each student's schedule	Schedule item data	W	1
Send info on any schedule changes to students through mail	Student schedule changes	X	1
			9

TABLE IX. "DELETE A SCHEDULE" DATA MOVEMENTS

Sub-process description	Data group	Data movement Type	CFP
The student enters the deletions for selected item(s)	Schedule item data	E	1
The system prompts to verify the deletion	System command	N/A	
The student confirms the deletion	Schedule item data	N/A (Repeat of Entry above)	
The system updates the student's schedule	Schedule item data	W	1
Display error message	Message	X	1
			3

C. *Quick/Early Accuracy on the C-Registration System*

We applied Quick/Early method on the case study described in [26], which contains a very detailed COSMIC analysis about the Management Information System C-Registration, whose functional size is equal to **102 CFP** [26]. Using Quick/Early we obtained the size estimate equal to **95 CFP**. Table X shows the details for each functional process and some summary measures. The accuracy of the obtained estimate was assessed using summary metrics such as MAR, MdAR, SAR which represent the mean, median, and sum of absolute errors, respectively. The absolute error is given by:

$$ABS = |SIZEreal - SIZEapp|$$

where SIZEreal is the standard COSMIC size and SIZEapp is the approximate size given by Quick/Early. Based on the relative error:

$$MRE = |SIZEreal - SIZEapp| / SIZEreal$$

we can calculate the Prediction at level *l* that measures the percentage of the approximations whose MRE is less than *l*% and *l* is usually set at 25%. It can be defined as

$$Pred(l) = k/N$$

where N is the total number of functional processes and k is the number of functional processes whose MRE is less than or equal to *l*.

In this preliminary study, we found that MAR is 0.41, MdAR is 0, and SAR is 7. Moreover, 94% of the estimated values fall within 25% of MRE and 76% within 5%. We can say that for this case study the accuracy of Quick/Early is good.

TABLE X. QUICK/EARLY ACCURACY

	CFP		Error	
Functional process	***Quick Early***	***COSMIC***	***ABS***	***MRE***
Login	3	3	0	0
Add a professor	5	5	0	0
Enquire on a Professor	4	4	0	0
Modify a Professor	3	3	0	0
Delete a Professor	3	3	0	0
Select Courses to teach	9	9	0	0
Add a Student	4	4	0	0
Enquire on a Student	4	4	0	0
Modify a Student	3	3	0	0
Delete a Student	3	3	0	0
Create a Schedule	13	13	0	0
Enquire on a Schedule	4	4	0	0
Modify a Schedule	10	13	3	0.23
Delete a Schedule	3	4	1	0.25
Close Registration	8	9	1	0.11
Submit grades	12	12	0	0
View Report Card	4	6	2	0.33
TOTAL	**95**	**102**	**7**	**0.92**

Accuracy Summary Measures				
MAR	**MdAR**	**SAR**	**Pred(0.25)**	**Pred(0.05)**
0.41	**0**	**7**	**0.94**	**0.76**

VI. RELATED WORK

A. *Early and rapid sizing for the COSMIC method*

Besides the methods provided in Advanced & Related Topic COSMIC [2] and described in section II there are other related work in the literature.

In [6] the authors analyze two approximate measurements: the first based on the number of COSMIC functional processes and the second based on AFP proposed in [2]. This work, similarly to [7], [8], [9], [10] and [11] focuses on to predict the effort to develop Web Applications.

[12] proposes different approaches to rapid estimate MIS applications, based on the number of physical tables and the number of tables read/written, which can be easily calculated from the database schema and application interfaces. Obviously this approach cannot be an early method but only a quick method.

In [13] the authors present a fully manual approach to quickly approximate the functional size using the COSMIC method from the textual requirements, without extracting the artifacts of the COSMIC models. This approach classifies projects already developed in fuzzy classes, looking for similarities between concepts belonging to the same class, and finally allows discover the similarities between software components, to estimate CFP by analogy with past projects.

In [14] the authors extends the approach described in [13] and assert that COSMIC is not compatible with Agile processes, because COSMIC requires that the requirements are formalized and decomposed to a level of granularity in which the interactions between user and system are visible to the measurer. This analysis is time consuming, and then avoided by Agile processes, leaving the subjective judgment, which often leads to errors, as the last option. The authors proposed an approach to approximate CFP starting from informally written textual requirements, demonstrating its applicability in Agile processes. In addition, the authors discuss the preliminary results of an experiment to automate the process using supervised text mining. This supervision requires experts to create the historical database, manually measuring the functional size using COSMIC on textual requirements. The approach, however, has not been tested on requirements at a variable level of quality. According to the authors, this means that the approach is company specific, where the textual requirements stored in the historical database should be written in the same format with the same quality and style.

B. Early and rapid sizing for other FSMs

There are several studies related to this paper which proposed approaches to approximate and quick size other FSMs. Two approaches to simplify Function Point Analysis are Early & Quick Function Point (EQFP) [13] and the simplified version of NESMA [15].

EQFP purpose is estimating the functional size before the requirements analysis has been completed. The main idea is that different pieces of software can be measured at different level of detail. In this way, using coarser grained components than the Base Functional Component (BFC) of FPA (i.e., EI, EO, EQ, ILF, and EIF), EQFP produces an approximated size expressed in Function Points. In particular, EQFP ranking data and processes in different groups according to a hierarchy [13], where some groups contain the BFC of FPA and other groups are aggregates of BFC (i.e., Macro Processes, General Processes) or data groups (i.e., multiple data groups).

The first approximation suggested by NESMA (named Indicative NESMA) is based on the identification of Internal Logical Files (ILF) and External Interface Files (EIF) without considering, for example, the contribution of transactional functions (i.e., EI, EQ and EO) of the FPA, and applying predefined weights [15].The second approximation proposed by NESMA (named Estimated NESMA) uses all the BFC of the FPA, without determining the identified components complexity. By default, a low complexity is assigned to ILF and EIF, while an average complexity is assigned to EI, EQ and EO.

EQFP and the simplified NESMA approaches have been applied to measure both traditional and real-time applications in order to assess the accuracy compared to FPA [16]. The result of this analysis shows that EQFP produces an estimate of the functional size of MIS applications, while the NESMA approaches are sufficiently accurate for MIS and real-time domains.

An initial proposal, very close to EQFP, was presented in [17], which describes Early Function Points (EFP) and Extended Function Point (XFP) methods, which aim to achieve, faster, the number of Function Points in the initial phase of the development life cycle. EFP suggests applying FPA to the available information to produce estimates compatible with IFPUG 4.0. XFP extends EFP applying factors, based on the level of reuse and classification algorithms, in terms of their complexity. EXP is not compatible with IFPUG 4.0, but [17] shows that it is able to produce estimates more accurate than EFP.

In [18] two dimensional metrics based on use cases (such as transactions and entities) are presented to estimate the effort in two case studies based on information derived from four academic projects and from four industrial projects. The studies compare the estimates obtained using these two metrics and show that the "*transactions*" metric can provide better effort estimates than the "*entity objects*" metric.

Related work can be considered those that analyze the different contributions to the total size of BFC (i.e., UFP for IFPUG [19] or CFP for COSMIC) in the effort estimation. In particular, Abran et al. was the first to analyze the contribution of individual BFC to the effort [20]. The results show that the effort correlation is different for each BFC within a data set.

Empirical studies based on ISBSG samples [21] showed that regression models which use two or more BFC (i.e., EI, EO, EQ, ILF, EIF for IFPUG method and Entry, Exit, Read and Write for COSMIC) as independent variables instead of the total fsu size (i.e., UFP for IFPUG and CFP for COSMIC) have more accurate estimates [22][23].

The study presented in [24] [25] are further applications of [22] and [23], using a data set of Web applications developed by a single Italian company. The result confirmed that using a subset of BFC types allows obtaining effort estimates close to the total functional size.

VII. CONCLUSIONS

Simplified FSM methods are often used when a project manager needs an estimate of the functional size of the software application to be built before the requirement specification phase is completed, or when the cost or time allowed for measurement are limited. Based on these considerations we introduced the simplified method named Quick/Early. This method allows us to estimate and quickly measure FURs, even in the early stage, requiring only the availability of an appropriately documented use cases model.

Because identifying data groups, and then "*type of data movements*" (Entry, Exit, Read and Write) in the available documentation is not always possible, this method simplifies the COSMIC standard process, requiring only the counting of "*data movements*". In this way it is possible to obtain a functional measure less accurate but more rapid. Being a central model, the use cases model is implicitly related to any other UML models. Thus, the measurement of this model can provide a reference measurement, and can be sufficiently accurate, as shown in the case study illustrated in the paper.

The next research phase should focus on the practical experimentation of the proposed approach with a large number of projects. In particular, it is essential to analyze the results obtained applying Quick/Early on projects of different domains (MIS, embedded and real-time, etc...), in order to demonstrate its general validity and perform more accurate comparative analysis with the approaches described in [2]. It will be also useful to realize experiments and case studies involving software analysts and project managers to verify the effectiveness of the proposed use cases template both for formulating requirements and to quickly and accurately measure them.

REFERENCES

[1] A. Abran, J.-M. Desharnais, S. Oligny, D. St-Pierre and C. Symons, "The COSMIC Functional Size Measurement Method Version 3.0.1, Measurement Manual, The COSMIC Implementation Guide for ISO/IEC 19761:2003," Common Software Measurements International Consortium (COSMIC), Montreal, 2009.

[2] A. Abran, B.Londeix, M. O'Neill, L. Santillo, F. Vogelezang, J.-M. Desharnais, P. Morris, T. Rollo, C. Symons, A. Lesterhuis, S. Oligny, G. Rule, H. Toivonen, "The COSMIC Functional Size Measurement Method, Version 3.0, Advanced and Related Topics,", 2007.

[3] A. Albrecht, "Measuring Application Development Productivity," in Procs. of the Joint SHARE/GUIDE/IBM Application Development Symposium, 1979, pp. 83–92.

[4] IFPUG, "International Function Point User Group - www.ifpug.org."

[5] Software Measurement European Forum 2009, "Functional size measurement . accuracy versus costs – is it really worth it?," May 2009, www.dpo.it/smef2009/presentazioni/d104.pdf.

[6] L. De Marco, F. Ferrucci, and C. Gravino, "Approximate COSMIC size to early estimate Web application development effort," SEAA & DSD Euromicro Conferences 2013 Santander, Spain, September 2013.

[7] E. Mendes, S. Counsell, and N. Mosley, "Comparison of Web Size Measures for Predicting Web Design and Authoring Effort," IEE Proceedings-Software, vol. 149, no. 3, pp. 86-92, 2002.

[8] G. Costagliola, S. Di Martino, F. Ferrucci, C. Gravino, G. Tortora, and G. Vitiello, "A COSMIC-FFP approach to predict web application development effort," Journal of Web Engineering, vol. 5, no. 2, pp. 93-120, 2006.

[9] S. Di Martino, F. Ferrucci, and C. Gravino, "Estimating Web Application Development Effort Using Web-COBRA and COSMIC: An Empirical Study," Euromicro Conference on Software Engineering and Advanced Applications. ACM press, 2009, pp. 306-312.

[10] S. Abrahão, L. De Marco, F. Ferrucci, C. Gravino, and F. Sarro, "A COSMIC measurement procedure for sizing web applications developed using the OO-H method", Workshop on Advances in Functional Size Measurement and Effort Estimation, 2010, pp. 2:1–2:8.

[11] F. Ferrucci, C. Gravino, and S. Di Martino, "A Case Study Using Web Objects and COSMIC for Effort Estimation of Web Applications," in Procs. of Euromicro Conference on Software Engineering and Advanced Applications. IEEE press, 2008, pp. 441–448.

[12] F. Ferrucci, C. Gravino, and G. Moretto, "A Lean Approach to Estimate the Functional Size of Operating Applications", SEAA & DSD Euromicro Conferences 2013 Santander, Spain 4-6 September

[13] L. Santillo, M. Conte, and R. Meli, "Early & quick function point: Sizing more with less," in Procs. of the 11th IEEE International Software Metrics Symposium, . METRICS '05, 2005, pp. 41.

[14] sI. Hussain, L. Kosseim, and O. Ormandjieva: "Approximation of COSMIC functional size to support early effort estimation in Agile", Data & Knowledge Engineering 85 (2013) 2–14.

[15] NESMA, "Definitions and Counting Guidelines for the Application of Function Point Analysis," 1997. [Online]. Available: www.nesma.org.

[16] R. Meli, "Report on using simplified function point measurement processes," in International Conference on Software Engineering Advances, pp. 367–372.

[17] R. Meli, "Early and extended Function Point: A new method for Function Points estimation", in Procs of the IFPUG-Fall Conference, 1997.

[18] G. Robiolo, and R. Orosco, "Employing use cases to early estimate effort with simpler metrics", Innovations Syst. Softw. Eng., vol. 4, 2008.

[19] ISO, I. 14143-1:2007, "Information technology – Software measurement – Functional size measurement – Part 1: Definition of concepts," 2007. [Online]. Available: http://www.iso.org.

[20] A. Abran, B. Gil, and E. Lefebvre, "Estimation models based on functional profiles," Procs. International Workshop on Software Measurement, Shaker Verlag, pp. 195–211.

[21] ISBSG, "ISBSG Data Repository r10, January 2007, URL: www.isbsg.org," 2007. [Online]. Available: www.isbsg.org.

[22] L. Buglione and C. Gencel, "Impact of Base Functional Component Types on Software Functional Size Based Effort Estimation," In Procs. of Software Process and Product Measurement, International Conferences, LNCS 5891 Springer, 2008, pp. 75–89.

[23] C. Gencel and L. Buglione, "Do base functional component types affect the relationship between software functional size and effort?", Software Process and Product Measurement, 2008.

[24] F. Ferrucci, C. Gravino, and L. Buglione, "Estimating web application development effort using COSMIC: Impact of the base functional component types," , Sof. Measurement European Forum, 2010, 103-116.

[25] L. Buglione, F. Ferrucci, C. Gencel, C. Gravino, and F. Sarro, "Which COSMIC base functional components are significant in estimating web application development? - a case study,", IWSM/MetriKon/Mensura 2010. Alain Abran, Gnter Bren, Juan J. Cuadrado-Gallego, Reiner R. Dumke., Shaker Verlag, 2010, pp. 205–224.

[26] A. Khelifi and A. Abran , Proposed Measurement Etalon: "C-Registration System". Research Laboratory of École de technologie supérieure - Université du Québec (Canada). www.cosmicon.com/portal/public/CRS_RUP_Case_%20Study_version_Jan_04_2007_web_%20version_update_feb_2008.pdf

[27] D. Conte, H.E Dunsmore:, and V.Y.Shen: "Software engineering metrics and models". The Benjamin/Cummings Publishing Company, Inc., 1986.