Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A measurement method for sizing the structure of UML sequence diagrams

Asma Sellami [a], Hela Hakim [a,*], Alain Abran [b], Hanêne Ben-Abdallah [c]

[a] *Computer Science Engineering Department, University of Sfax, Tunisia*
[b] *Software Engineering and Information Technology Department, École de Technologie Supérieure, University of Quebec, Canada*
[c] *Department of Information Systems, Software Engineering, King Abdulaziz University, Saudi Arabia*

A R T I C L E   I N F O

Article history:
Received 2 May 2014
Received in revised form 16 October 2014
Accepted 6 November 2014
Available online 27 November 2014

Keywords:
COSMIC ISO 19761
UML sequence diagram
Combined fragment
Functional size measurement
Structural size measurement
Different levels of granularity

A B S T R A C T

Context: The COSMIC functional size measurement method on UML diagrams has been investigated as a means to estimate the software effort early in the software development life cycle. Like other functional size measurement methods, the COSMIC method takes into account the data movements in the UML sequence diagrams for example, but does not consider the data manipulations in the control structure. This paper explores software sizing at a finer level of granularity by taking into account the structural aspect of a sequence diagram in order to quantify its structural size. These functional and structural sizes can then be used as distinct independent variables to improve effort estimation models.
Objective: The objective is to design an improved measurement of the size of the UML sequence diagrams by taking into account the data manipulations represented by the structure of the sequence diagram, which will be referred to as their structural size.
Method: While the design of COSMIC defines the functional size of a functional process at a high level of granularity (i.e. the data movements), the structural size of a sequence diagram is defined at a finer level of granularity: the size of the flow graph of their control structure described through the *alt*, *opt* and *loop* constructs. This new measurement method was designed by following the process recommended in Software Metrics and Software Metrology (Abran, 2010).
Results: The size of sequence diagrams can now be measured from two perspectives, both functional and structural, and at different levels of granularity with distinct measurement units.
Conclusion: It is now feasible to measure the size of functional requirements at two levels of granularity: at an abstract level, the software functional size can be measured in terms of COSMIC Function Point (CFP) units; and at a detailed level, the software structural size can be measured in terms of Control Structure Manipulation (CSM) units. These measures represent complementary aspects of software size and can be used as distinct independent variables to improve effort estimation models.

1. Introduction

Effectively managing software projects largely depends on the development of a realistic estimation of project attributes including effort, schedule, resources, cost, etc. In addition, predicting the effort at the beginning of a software project (requirements analysis and design phases) helps to adequately plan any forthcoming activities (construction, test, deployment). Most effort estimation models, techniques and tools use the software size as an input on which the estimate is based [2,3]: indeed, as the size of software increases, so does the development effort.

Historically, source lines of code (SLOC) have been used as the size measurement, but they are highly sensitive to programming languages and style and it is difficult to estimate lines of code early in a project. To overcome these drawbacks, functional size measurement (FSM) methods are technology independent and can be derived from the requirement specifications early in the software development life cycle. Functional size can be used to forecast software project effort, to derive productivity ratios or to do benchmarking.

Software functional size can be measured using various ISO FSM methods such as: COSMIC (ISO 19761 [4]), IFPUG (ISO 20926 [5]), MKII (ISO 20968 [6]), NESMA (ISO 24570 [7]), and FISMA (ISO 29881 [8]). For example, the COSMIC method can be used to measure the UML sequence diagrams which document the use cases representing the user functional requirements. These UML sequence diagrams are used in the requirement analysis, architectural design and detailed design phases.

* Corresponding author.

http://dx.doi.org/10.1016/j.infsof.2014.11.002
0950-5849/© 2014 Elsevier B.V. All rights reserved.

Unlike other ISO FSM methods, COSMIC does not take into account the structural aspect of sequence diagrams, i.e. the behavioral control structure (e.g. number of packages, subsystems, files, classes, subroutines). Such functional details require of course additional coding, testing and verification effort; hence there could be benefits in considering them when sizing software requirements for estimation purposes.

ISO 14143-1 [9] defines the software functional size as "the size derived by quantifying the functional user requirements (FUR)". FURs represent "user practices and procedures that the software must perform to fulfill the user's needs, and exclude quality requirements and any technical requirements" (ISO 14143-1 [9]).

Given that UML [10] has been adopted as a *de facto* standard modeling language, researchers have explored the use of the COSMIC method for measuring the functional size of UML diagrams with a focus on particular diagrams in a specific version of UML:

- the use case diagram: Bévo et al. (UML1.0) [11]; Berg et al. (UML2.0) [12]; Sellami and Ben-Abdallah (UML2.0) [13]; Lavazza and Robiolo (UML2.0) [14];
- the class diagram: Bévo et al. (UML1.0) [11]; Berg et al. (UML2.0) [12]; Sellami and Ben-Abdallah (UML2.0) [13]; Nagano and Ajisaka (UML2.0) [15];
- the sequence diagram: Lavazza and Robiolo (UML2.0) [14]; Lavazza et al. (UML2.0) [16]; Lévesque et al. (UML 1.4/UML 2.0) [17]; Sellami and Ben-Abdallah (UML2.0) [13]; and
- the component diagram: Sellami et al. (UML2.3) [18]; Lavazza and Robiolo (UML2.0) [14].

These studies explored the functional size of UML diagrams at a high level of granularity and without taking into account the details expressed at a finer level: for instance, [13,14,16] do not include either the specific details in the combined fragments of these diagrams which represent the behavioral aspects of software, or their usefulness in more extensively representing the user requirements.

To tackle the challenge of a more comprehensive measurement of the size of a sequence diagram, this paper proposes to measure the software control structures. Indeed, the control structure of software has a direct impact on the efforts needed for its development; for example, a large control structure will require more effort for thorough testing. Despite the impact and usefulness of the control structure of software in representing system details, none of the ISO-recognized FSM methods investigates their size.

This paper provides two main contributions. First, it completes our previous work [13] with a new measurement method for the structural size of sequence diagrams at a finer level of granularity, i.e. the size of their control structures described through the *alt*, *opt*, or *loop* constructs. Secondly, it evaluates the comprehensive measurement method through case studies. Note that, even though UML sequence diagrams are investigated in the work herein reported, the concepts discussed in this paper are applicable to measurement rules with any type of functional specification's notation when used in the context of comprehensive software requirements in response to sophisticated needs as observed in industry.

The remainder of this paper is organized as follows: Section 2 presents an overview of related work on UML sequence diagrams and the COSMIC FSM method; Sections 3 and 4 respectively present a set of procedures for sizing sequence diagrams at a high level of granularity using the COSMIC rules, and for the measurement of the structural size of sequence diagrams at a finer level of granularity; Section 5 illustrates these procedures with the *"Rice Cooker"* case study; Section 6 illustrates the impact of structural size through sample case studies from the literature; and finally, Section 7 summarizes the work presented and outlines future work.

2. Related works

2.1. UML sequence diagrams

UML sequence diagrams are used for dynamic modeling. They focus on the behavior within a system and the interactions among its objects [19]. A sequence diagram can take many forms, each with a corresponding level of granularity:

(1) simplified: interactions among actors with the system represented as a black box (i.e. high level of granularity);
(2) detailed: interactions among objects inside the system as a white box (i.e. fine level of granularity); or
(3) a combination of simplified and detailed (i.e., fine level of granularity).

With the publication of the UML2.0 version, it became possible to distinguish subsets of interactions in the sequence diagrams, such as combined fragments or interaction fragments. A combined fragment contains interaction operands (interaction fragment and messages) and it is defined by the interaction operators.

More specifically, there are 13 interaction operators defined in UML 2.0 [20] describing the control flow within a sequence diagram: *alt*, *opt*, *loop*, *break*, *par*, *critical*, *ignore*, *consider*, *assert*, *ref*, *neg*, *weak sequencing*, and *strict sequencing*. For example:

- The *alt* (alternative) indicates that the combined fragment represents a behavioral choice with at most one of the operands chosen. The chosen operand must have an explicit or implicit guard expression. At this point of interaction, the guard condition should be verified as true (Fig. 1(a)).
- The *opt* (optional) operator indicates that the combined fragment represents a behavioral choice, and that either the (unique) operand happens or nothing happens (Fig. 1(b)).
- The *loop* operator indicates that the combined fragment represents a repeated behavior where the *loop* operand will be repeated a number of times that depending on the guard of the *loop*. The guard may include a lower and an upper number of iterations of the *loop* as well as a Boolean expression. The *loop* will terminate after the verification of the number of iterations and/or the Boolean expression (Fig. 1(c)).

In this paper, the *alt*, *opt*, and *loop* fragments are collectively referred to as the control structure of a sequence diagram (Fig. 1).

2.2. Cosmic measurement of sequence diagrams

Researchers have studied the use of COSMIC to measure the functional size of various UML diagrams in CFP (COSMIC Function Point) units. For instance, Lévesque et al. [17] used both the use case and the sequence diagrams, and suggested adding a number of conditions related to UML 2.0. They classified the functional processes into two groups: data movement types, which correspond to messages; and data manipulation types, which correspond to the conditions associated with the error messages in a sequence diagram. However, their measurement approach for the sequence diagram is at a high-level of granularity.

Lavazza and Bianco [16] defined a mapping between all COSMIC concepts and UML sequence diagrams to allow the modeler, on the one hand, to build measurement-oriented models and, on the other hand, to allow the measurer to apply the COSMIC measurement rules more easily. This application of COSMIC on the sequence diagram provides for measurement only at a high level of granularity (i.e. without considering the interactions in the combined fragments).

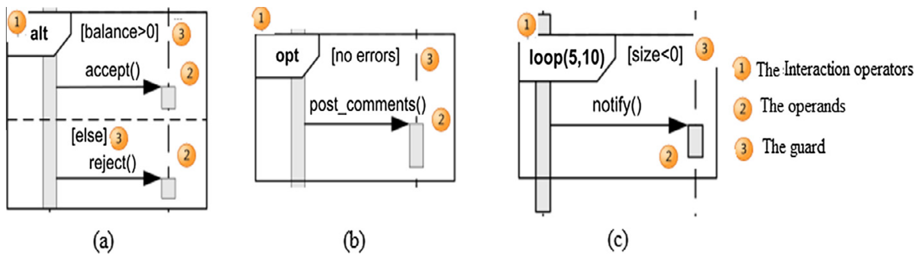


Fig. 1. Illustrative example of *alt*, *opt* and *loop* combined fragments.

Table 1
COSMIC measurement proposals of UML sequence diagrams.

Criteria	Research		
	Lévesque et al. [17]	Lavazza and Bianco [16]	Sellami and Ben-Abdallah [13]
UML version	UML1.4/UML2.0	UML 2.0	UML 2.0
Level of granularity	High level	High level	High level

Sellami and Ben-Abdallah [13] presented an approach to measure the functional size of use case diagrams and stressed the importance of the semantic links among the various UML diagrams in any measurement process. In particular, they proposed to use the functional size of the use case diagram as a reference for sizing the sequence and class diagrams, i.e. a size measure of these diagrams is representative of the size of use case diagrams. However, this initial proposal was limited to deriving measures from an external viewpoint, that is, the interactions between an actor and the system as a black box (simplified form at a high level of granularity).

As summarized in Table 1, the related works consider the sequence diagram at a high level of granularity, measuring only the interactions among objects inside the system as a white box, and not the interactions among objects in the combined fragments (referred to here as a fine level of granularity). Consequently, a sequence diagram with simple interactions between objects and another one with a control structure (i.e. combined fragments *alt*, *opt*, *loop*) have the same functional size in CFP units; this is the case despite the difference in the size of their control structures. This measurement approach at a high level of granularity does not quantify the size of the control structure even though it is expected that the effort to implement a simple or a combined structure will differ. In other words, this high-level measurement approach does not allow FSM-based estimation models to take the control structure information into account for estimating project effort.

3. An FSM procedure for the sequence diagram at the current level of granularity of the cosmic method

3.1. Mapping cosmic to sequence diagram concepts

Table 2 shows the mapping between the COSMIC concepts and those of the sequence diagram instantiated with Jacobson's stereotypes. Note that only the layer concept present in the COSMIC method is not present in the set of sequence diagram concepts, where a layer is a partition resulting from the functional division of a software system level and a sequence diagram corresponds to a functional process level.

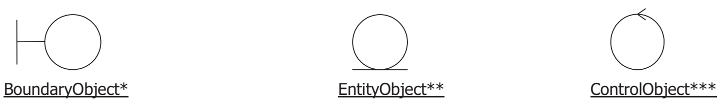
3.2. Modeling rules

To measure the functional size of a sequence diagram at the current high level of granularity of the COSMIC ISO 19761 standard, we specify two rules (**InterR1** and **InterR2**) to handle the interactions among the objects of a sequence diagram. These rules are based on the decomposition of a sequence diagram into two types according to Jacobson's stereotypes [20]:

(i) a user interface (UI) or physical device <<boundary>> object (Fig. 2(a)); and
(ii) at least one object that can be the <<entity >> object (Fig. 2(b)) or the <<control>> object (Fig. 2(c)).

Table 2
Mapping COSMIC concepts to UML sequence diagram concepts.

COSMIC concepts		UML Sequence diagram concepts
System boundary		Virtual line separating a UI/physical device and an object(s)
Functional user		A UI/physical device
Functional process		Messages exchanged between objects in a sequence diagram
Types of data movement	Entry	Messages exchanged from a UI/ physical device to an object
	Exit	Messages exchanged from an object to a UI/physical device
	Read/Write	Messages exchanged between two objects in the system
Groups of persistent data		A UI/physical device and an object
Event		Incoming message to a given object: Messages exchanged between a UI/physical device and objects
Layer		Not applicable



*objects that interface with system actors (e.g. a user or external service) windows, screens and menus are examples of boundaries that interface with users

** Objects representing system data, often from the domain model

***objects that mediate between boundaries and entities object. These serve as the glue between boundary elements and entity elements, implementing the logic required to manage the various elements and their interactions

Fig. 2. Jacobson's stereotypes [Roque 2009].

Table 3
Message types in a UML sequence diagram and their associated size *FS*() in CFP units.

Message type according to UML2.0 [20]	Symbol	Data Movement	CFP units
Sending message	→	E or X	1
Call procedure message (create and delete) (synchronous or asynchronous)	→▯	W or R	1
Return message	⇢	Not applicable	0
Reflexive message (call)	↺		
Reflexive message (return)	⇠		

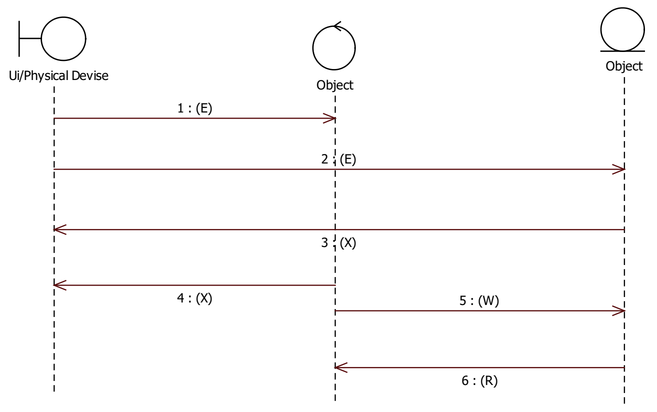


Fig. 3. COSMIC data movement types in a sequence diagram – rules **InterR1** and **InterR2**.

- **InterR1:** If the sender of a message is a UI/physical device and the receiver is an object, then the data movement is considered as an Entry (E), even if the message requires data from the receiver. Otherwise, if the message is transmitted between two other objects, it will be either a Write (W) data movement or a Read (R) data movement (Fig. 3).
- **InterR2:** If the sender of a message is an object and the receiver is a UI/physical device, then the data movement type is an eXit (X), except when the message requires data from the receiver; in which case, it will be an Entry (E) (this is a general rule on the data requested by a functional user [21]) (Fig. 3).

3.3. Sizing a sequence diagram in CFP units

At the current high level of granularity of the COSMIC FSM method, the functional size of a sequence diagram depends on the number and types of messages exchanged. Table 3 presents the types of messages for the example illustrated in Fig. 3 along with their corresponding measurement results in CFP units. Note that, being local to an object, reflexive call and return messages do not result in any visible data movement. Thus, as shown in Table 3, these messages do not contribute to the CFP size of the sequence diagram.

Given the CFP values based on the message type, the classic COSMIC functional size of a sequence diagram is determined by adding up, arithmetically, the functional sizes of the diagram's individual data movements (i.e. interactions among its objects). The value of 1 CFP is assigned to each data movement. More precisely, the COSMIC functional size (FS) of a sequence diagram *Sq* is calculated according to Eq. (1):

$$COSMIC\_FS(Sq) = \sum_{i=1}^{n} FS(m_i) \quad (1)$$

where n is the number of messages m_i in the sequence diagram *Sq*, and $FS(m_i)$ is the CFP value of the message m_i calculated according to the message type as indicated in Table 3.

4. Measurement of the structural size of a sequence diagram at a fine level of granularity

This section first presents the generic design of the proposed measurement method for the structural size of sequence diagrams, followed by the proposed measurement procedure itself.

4.1. The generic design of the structural size of sequence diagrams

The process recommended in Software Metrics and Software Metrology [1] is used in this section for the design of a new measurement method. This process consists of four steps:

(1) determination of the measurement objectives;
(2) characterization of the concept to be measured;

(3) design or selection of the meta-model to be used in the measurement procedure; and
(4) definition of the numerical assignment rules.

Each of the above four steps is presented next in detail for the design of a new method to measure the size of the control structure of the sequence diagrams.

4.1.1. Step 1: Determination of the measurement objectives

This step fixes answers to the following questions:

(a) *What is to be measured?* In this case, the "structural size" of a sequence diagram is to be measured.
(b) *What is the attribute to be measured and its corresponding entity?* The attribute to be measured is the "size" of the "control structure" of the sequence diagram, and the entity is the "control structure" of the sequence diagram. These structures are shown in the control flow graph (see Step 4).
(c) *What is the point of view of the measurement?* The viewpoint is that of the developers (e.g. designers, programmers, testers, etc.).
(d) *What are the intended uses of the measurement results?* The measurement results are intended to be used for managing and estimating the effort in software projects.

4.1.2. Step 2: Characterization of the concept to be measured

The concept of the structural size will be considered as a common, generic concept of sequence manipulation in sequence diagrams. It can be divided into several levels of more specific concepts, such as: conditional control structures (S_{cc}) which include *alt* and *opt* fragments; and iterative control structures (S_{ic}), which include *loop* fragments.

4.1.3. Step 3: Design of the meta-model

A set of characteristics selected to represent software or pieces of software, together with the set of their relationships, constitute the meta-model of the software to which the proposed measurement method will be applied [1]. Therefore, a meta-model can be used to describe not only the entities and attributes (as measurement concepts) to be taken into account in the measurement method, but also the roles and relationships of these concepts.

Fig. 4 shows the meta-model of the entity "control structure" selected to be measured and adapted from the <<enumeration>> of the combined fragments described in the Appendix. This entity of the sequence diagram can be decomposed into streams and flows controls, modeling one or more conditional control structures and one or more iterative control structures. While a conditional control structure describes *alt* and *opt* fragments represented by an alternative and choice flow graph, an iterative control structure describes the *loop* fragments represented by an iterative flow graph.

4.1.4. Step 4: Definition of the numerical assignment rules

Defining numerical assignment rules requires that the measurement unit(s) be specified. According to the International Vocabulary of Metrology–VIM [22], they are designated by conventionally assigned names and symbols.

In our context, we define the numerical size 1 CSM (for control structure manipulation) as the size of one guard condition in a control structure in a sequence diagram. A guard condition describes the condition controlling the execution flow in the sequence diagram; in other words, it is a concept used to structure the message exchanges. Informally, the guard condition size represents the number of additional execution branches incurred by the guard condition value. As formalized in the next section, it depends on the control structure type.

4.2. Measurement procedure for sizing the control structure of a sequence diagram

To implement the design of the proposed measurement method, we propose the following three-step measurement procedure:

1. Map the combined fragments of the sequence diagram into a flow graph.
2. Identify and size the conditional and iterative control structures in the flow graph.
3. Measure the structural size of the sequence diagram expressed in CSM units.

4.2.1. Mapping the combined fragments of the sequence diagram into a flow graph

Table 4 presents the mapping of the concepts of combined fragments to the aspects of a flow graph.

Fig. 5 shows the various flow graphs representing the three types of combined fragments: (a) the *alt* fragment, (b) the *opt* fragment, and (c) the *loop* fragment.

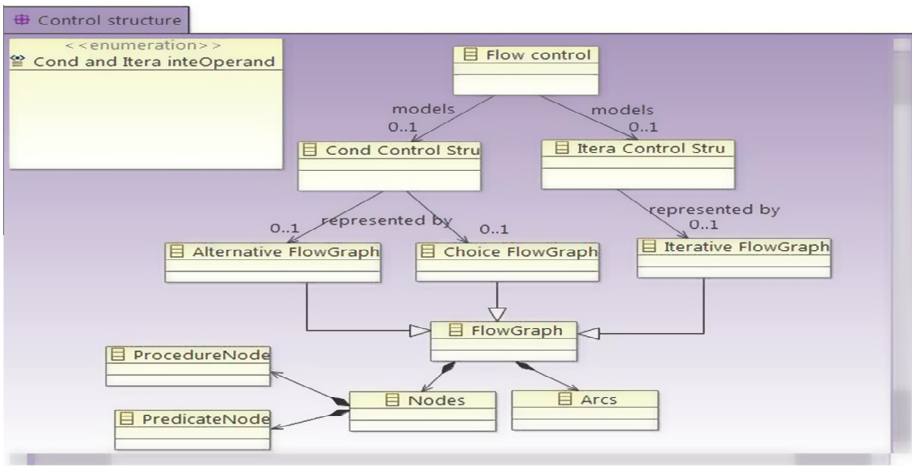


Fig. 4. Meta-model of the "control structure" of the sequence diagram.

Table 4
Rules for mapping a sequence diagram to a flow graph.

Combined fragments (*alt*, *opt*, and *loop*) in a sequence diagram [20]	Aspects of a flow graph [23]
Interactions among objects (messages)	Procedure nodes (statements for procedure ●)
Guard condition	Predicate nodes ◉ (or end nodes)
Relationship between messages and guards that represent a control flow	An arc oriented to represent a link connecting two nodes (edges)

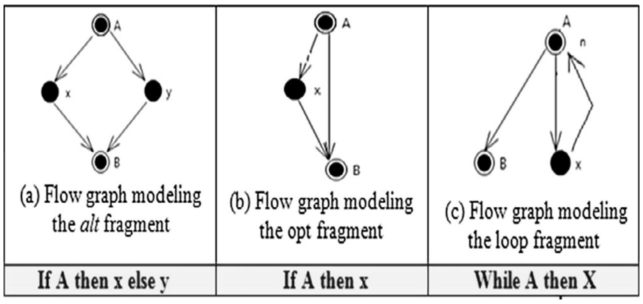
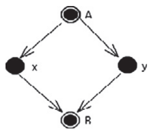
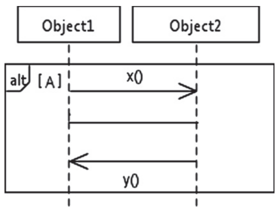
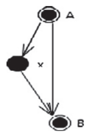
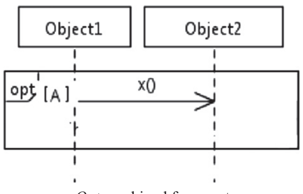
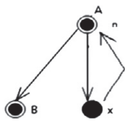
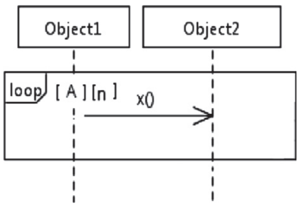

A
x
y
B
(a) Flow graph modeling the *alt* fragment
(b) Flow graph modeling the opt fragment
(c) Flow graph modeling the loop fragment
n
If A then x else y
If A then x
While A then X

Fig. 5. Flow graphs modeling the *alt*, *opt* and *loop* fragment.

Table 5
Mapping of the control structure with the sequence manipulation (interaction among objects) in a sequence diagram.

Control structure	Sequence manipulation
 Conditional control structure1 (switch)	 *Alt* combined fragment
 Conditional control structure2 (switch)	 *Opt* combined fragment
 Iterative control structure (iteration)	 *Loop* combined fragment

Note that the "guard condition" is represented by a predicate node associated with an arc defined as a switch coming out of this node (Fig. 5(a) and (b)), or an iteration going into this node (Fig. 5(c)).

In this context, the control structures illustrate the interactions in terms of sequence manipulations defined as a switch in a conditional control structure or an iteration of instructions in an iterative control structure:

- A switch is a deviation from a sequential execution, representing the various possible paths for the code/scenarios of execution and it is provided by an alternative *alt* or choice *opt* combined fragment.
- An iteration is the execution of the same set of instructions one or more times, and it is provided by *loop* combined fragments with the number of iterative executions labelling the arc going into the predicate node (see Table 5).

4.2.2. Identifying and sizing the conditional and iterative control structures in the flow graph

The identification and measurement of nodes representing control structures is handled in three steps:

1. For each node, compute its *in-degree* and *out-degree*.
2. Classify the nodes according to their types: For each node *x*,
   If out-degree $(x) \geqslant 2$ then it is a predicate node,
   else if *out-degree* $(x) = 1$ then it is a *procedure* node,
   otherwise it is an *end* node.

Furthermore, according to our mapping rules, each *predicate* node *x* has the following properties:

If *x* has a labelled incoming arc, then *x* is the root of a subgraph representing an iterative control structure (*loop*); we call *x* a *loop predicate* node,

else if *out-degree*$(x) = 2$ and one arc out of *x* connects it to an *end* node, then *x* is the root of a subgraph representing an *opt* conditional control structure; we call *x* an *opt predicate* node,otherwise *x* is the root of a subgraph representing an *alt* conditional control structure; we call *x* an *alt predicate* node.

3. Size the predicate nodes by applying the following graph-based structural size function:

$$GSS(x) = \begin{cases} 1, & \textit{if } x \text{ is an } \textit{opt} \text{ predicate node} \\ Outdegree(x), & \textit{if } x \text{ is an } \textit{alt} \text{ predicate node} \\ iter(x) * GSS(y), & \textit{if } x \text{ is a } \textit{loop} \text{ predicate node} \\ 0, & \textit{otherwise} \end{cases} \tag{2}$$

where

- *x* is a node,
- *iter(x)* is a function that returns the number of iterations—it is the label of the arc incoming to the predicate node *x;* and
- *y* is the non-end node representing the beginning of the *loop* fragment content.

With the above graph-based sizing of classified nodes, the structural size of a sequence diagram can now be computed based on its flow graph.

4.2.3. Measuring the structural size of a sequence diagram expressed in CSM units

The proposed structural size (SS) measurement of a sequence diagram is iteratively computed based on its control structures as represented in its flow graph.

Let *Sq* be a sequence diagram, *GSq* be its flow graph S_{ic} be the set of loop predicate nodes in *GSq* and S_{cc} be the set of nodes in *GSq* representing conditional (*alt* and *opt*) control structures. The structural size of *Sq* is computed in three steps:

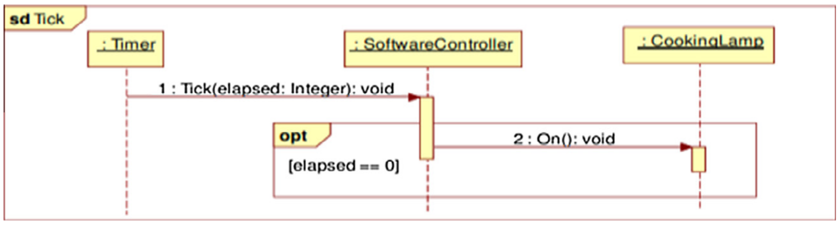


Fig. 6. Sequence diagram of the *Clock Tick* process [16].

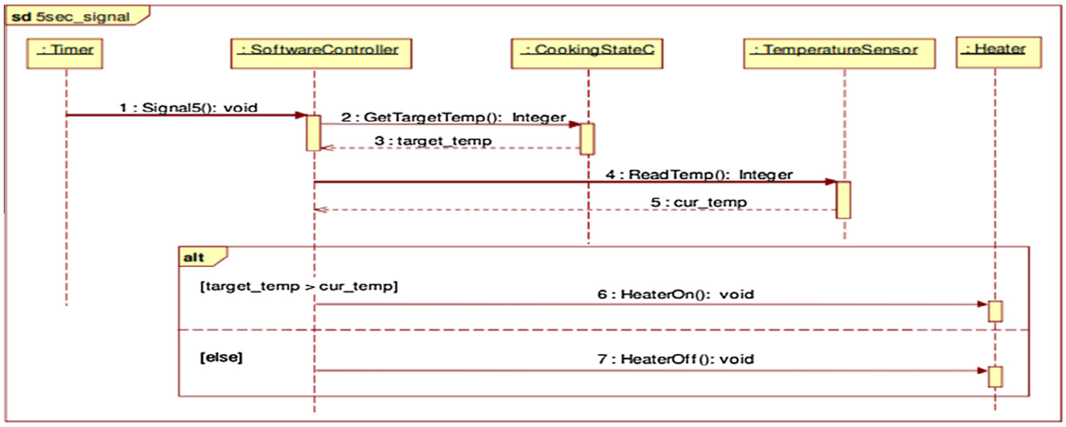


Fig. 7. Sequence diagram of the 5sec_signal process [16].

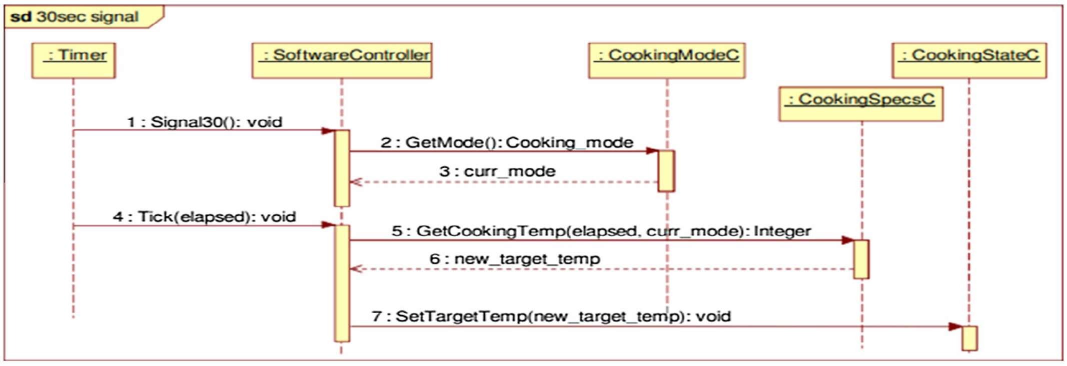


Fig. 8. Sequence diagram of the 30sec_signal process [16].

1. The size of all the iterative control structures is calculated:

$$SS_{ic} = \sum_{x \in Sic} GSS(x) \quad (3)$$

During this first sizing step, we mark all predicate nodes that are counted when following the outgoing arcs of the nodes *x* to size the inside fragment of the *loop*. Let *M* be the set of marked nodes.

2. The size of all conditional control structures whose nodes were not already counted in the first step is calculated:

$$SS_{cc} = \sum_{x \in Scc \setminus M} GSS(x) \quad (4)$$

3. Finally, the structural size of the sequence diagram is calculated:

$$SS(Sq) = SS_{ic} + SS_{cc} \quad (5)$$

5. Illustrative example: rice cooker

5.1. Functional user requirements

To illustrate the application of the proposed measurement method and procedures, the case study of the sequence diagrams of the real time software “Rice Cooker”, as described in [16], is used. Here, the focus is on the sequence diagrams of the software controller of this case study.

Due to space limitations, we will limit ourselves to the measurement results of the sequence diagrams with both simple and combined (*alt*, *opt* and *loop*) interactions. Figs. 6–8 illustrate the sequence diagrams of the three functional processes of this case study: *Clock Tick*, *5sec_signal*, and *30sec_signal*.

5.2. Functional sizing of the rice cooker in cosmic CFP units

The first step in measuring the *FS(Sq)* (see Section 3.1) requires an instantiation of this diagram with Jacobson’s stereotypes [20].

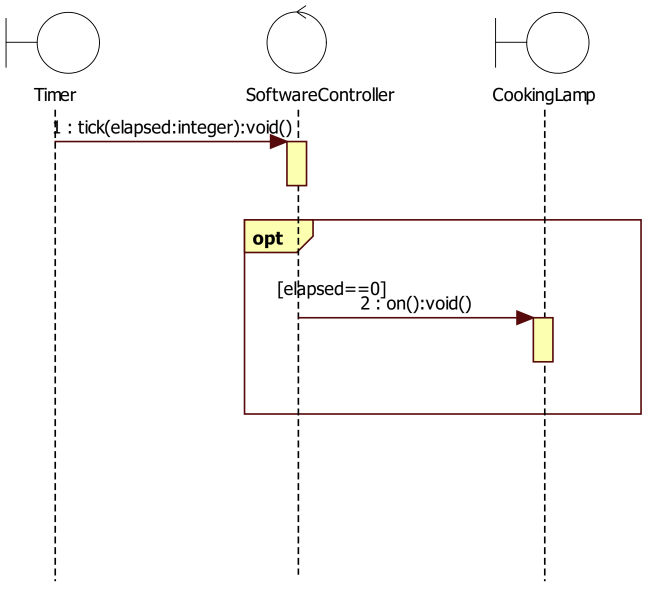


Fig. 9. Sequence diagram of *Clock Tick* presented with Jacobson's stereotypes.

Table 6
Functional size of the Rice Cooker in COSMIC CFP units – at a high-level of granularity.

Functional processes (*Sq*)	Functional size – FS (Sq)
Clock Tick	2 CFP
5sec_signal	5 CFP
30sec_signal	5 CFP
Total	12 CFP

For this, the *Clock Tick* functional process shown in Fig. 6 is transformed into Fig. 9. Afterwards, the **InterR1** or **InterR2** modeling rules described in Section 3.2 are applied.

The other step (*cf.* Section 3.3) yields a functional size of 2 CFP for *FS(Sq):*

- 1 CFP for the *Clock tick* (elapsed: integer) message whose type is (*E*), and
- 1 CFP for the message *On()* whose type is (*X*).

In an analogous way, the functional sizes of the 5sec_signal (Fig. 7) and 30sec_signal (Fig. 8) are measured. A summary of the measurement results of each of the three functional processes is presented in Table 6. The total functional size is 12 CFP.

5.3. Structural sizing of the rice cooker in CSM units

The structural sizes of the sequence diagrams of the Rice Cooker case study are measured at a finer level of detail by applying the three steps described in Section 4.2.

Tables 7 and 8 present the measurement steps of two fragments (*alt* and *opt*) for the *5sec_signal* and *Clock Tick* functional processes (i.e. mapping the *alt* and *opt* fragments on the flow graph).

By applying Eq. (5), the SS of the sequence diagrams describing the functional processes *Clock Tick* and *5sec_signal* is equal to the sum of the sizes of their iterative (Eq. (3)) and conditional (Eq. (4)) control structures. In this example, each sequence diagram contains a single iterative or conditional structure, except for the *30sec_signal* functional process, which does not contain either of these structures.

Table 9 summarizes the measurement results for each of the sequence diagrams of the three functional processes of the Rice Cooker example.

5.4. Discussion

As observed in Table 9, the structural size of each of the sequence diagrams related to the *Clock Tick* and *5sec_signal* functional processes is equal to 1CSM and 2CSM respectively; this means that these functional processes include sequence manipulation sub-processes which describe a control behavior structure in the *alt*, *opt* or *loop* combined fragment. In the classical measurement of these sub-processes at a high level of granularity, the distinct sizes of their behavior structures are not taken into account when only the data movements they are associated with are considered. Therefore, measurement in terms of CFP units does not quantitatively reveal the details included in data movements.

For example, the structural size of the sequence diagram on the *30sec_signal*, which is equal to 0 CSM, will indicate that the effort of

Table 7
Structural sizing: Measurement steps of the *alt* fragment.

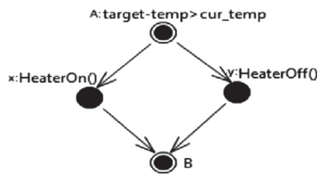
Measurement procedure	Combined fragments *Alt* in the sequence diagram of "*5sec_signal*" process
Step 1 (Section 4.2.1) Mapping the combined fragments into the flow graph	 A:target-temp>cur_temp x:HeaterOn() y:HeaterOff() B (b) A flow graph modeling the *alt* fragment <u>Step 1:</u> the identified nodes are: *A*, *x*, *y*, *B* *in-degree* (*A*) = 0; *out-degree* (*A*) = 2 *in-degree* (*x*) = 1; *out-degree* (*x*) = 1 *in-degree* (*y*) = 1; *out-degree* (*y*) = 1 *in-degree* (*B*) = 2; *out-degree* (*B*) = 0
Step 2 (Section 4.2.2) Identifying and sizing the conditional and iterative control structures in the flow graph	<u>Step 2:</u> *A* is an alt predicate node; *x* and *y* are procedure nodes; *B* is an end node
Step 3 (Section 4.2.3) Measuring the structural size of a sequence diagram in CSM units	<u>Step 3:</u> *GSS*(*A*) = 2 S_{cc} = {A} and $S_{ic} = \{\}, M = \{\}$ $\boldsymbol{SS_{cc}} = \sum_{\boldsymbol{x}\in\ \boldsymbol{Scc}\setminus M} \boldsymbol{GSS}(x) = 2\text{CSM} \quad (4)$ $\boldsymbol{SS}(\boldsymbol{Sq}) = \boldsymbol{SS}_{ic} + \boldsymbol{SS_{cc}} \quad (5)$ $\boldsymbol{SS}(\boldsymbol{Sq}) = 0 + 2 = 2\text{CSM}$

Table 8
Structural sizing: Measurement steps of the *opt* fragment.

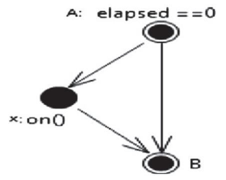
Measurement procedure	Combined fragments *Opt* in the sequence diagram of the "*Clock Tick*" process
Step 1 (Section 4.2.1) Mapping the combined fragments into the flow graph	 (a) A flow graph modeling the *opt* fragment
Step 2 (Section 4.2.2) Identifying and sizing the conditional and iterative control structure in the flow graph	Step 1: the identified nodes are: *A*, *x*, *B* *in-degree* (*A*) = 0; *out-degree* (*A*) = 2 *in-degree* (*x*) = 1; *out-degree* (*x*) = 1 *in-degree* (*B*) = 2; *out-degree* (*B*) = 0 Step 2: *A* is an opt predicate node; *x* is a procedure node; *B* is an end node
Step 3 (Section 4.2.3) Measuring the structural size of a sequence diagram in CSM units	Step 3: *GSS*(*A*) = 1 $S_{cc} = \{A\}$ and $S_{ic} = \{\}, M = \{\}$ $\boldsymbol{SS_{cc}} = \sum_{x \in \boldsymbol{Scc} \setminus M} \boldsymbol{GSS}(x) = 1\text{CSM}$ (4) $\boldsymbol{SS}(\boldsymbol{Sq}) = 0 + 1 = 1\text{CSM}$ (5)

Table 9
Structural size of the Rice Cooker in CSM units – at a fine level of granularity.

Functional Processes (*Sq*)	*Structural Size – SS* (*Sq*)
ClockTick	1 CSM
5sec_signal	2 CSM
30sec_signal	0 CSM
Total	3 CSM

developing a functional process with 0 CSM should be less than that of developing functional process with *n* CSM (where *n* > 0).

6. Size impact analysis

To further illustrate the expected usefulness of the proposed measurement method for effort estimation, this section presents two additional smaller case studies: Washing machine [24] and Airline reservation [25]. These two case studies involved the development of software systems. The goal of the first study was to develop embedded software modeled with one class diagram and eight sequence diagrams. The second case study represents an example of a management information system modeled with one use case diagram describing ten functionalities: manage staffing preparations, maintain equipment, manage flights, load/unload equipment and ringing, manage ringing, manage devices, take charge booking, make reservation, claim banding absence, claim banding damaged. Each of these functionalities (use cases) is further detailed through one sequence diagram. Both case studies have simplified as well as detailed forms of sequence diagrams.

Table 10 presents the measurement results related to the functional and structural sizes of the three case studies. From these measurement results, it can be observed that there are many more sequence manipulations in the Airline reservation case study (14 CSM) than in the Rice Cooker (3 CSM) and Washing machine (6 CSM) case studies; the same holds true in terms of CFP units.

Table 10
Functional and structural sizes of the three case studies.

Case study	Functional size (FS)	Structural size (SS)	Ratio (SS/FS)
Rice cooker	12 CFP	3 CSM	0.25 CSM/CFP
Washing machine	6 CFP	6 CSM	1.0 CSM/CFP
Airline reservation	16 CFP	14 CSM	0.88 CSM/CFP

It can also be observed that while the functional size of the Airline reservation case study is only a third larger than the Rice Cooker size ((16CFP − 12CFP) ÷ 12CFP = 0.33), its structural size is 2.5 times larger ((14CSM – 3CSM) ÷ 3CSM = 3.66). This clearly indicates that the difference in their structural size is proportionally larger than the difference in their functional size, and that should be accounted for to refine effort estimation models.

To do so, when the detailed information is available about the control structure of the sequence diagrams, the size of any functional process can be presented as a tuple representing two attributes: its functional size (FS) and its structural size (SS). These attributes can be appropriately correlated by a function *f* (e.g., ratio) defined with the corresponding proper measurement units to adequately assess their impact on effort. For example, the Rice Cooker case study can be measured with the following tuple: (12CFP, 3CSM) which gives a corresponding ratio of the structural size with respect to the functional size equal to *f(SS,FS)* = 3CSM/12CFP = 0.25 CSM/CFP. This ratio can be used, for instance, to appropriately distribute the testing resources among the various testing techniques. If we consider two different ratios, we can say whether these ratios are equal or not, and which is bigger (such as the concept of distance in the scientific field).

In summary, the above small set of relatively small case studies (Rice Cooker, Washing machine, Airlines reservation) shows: how to apply the COSMIC method and the proposed method; how to quantitatively evaluate the size of the sequence diagram at different levels of granularity; and that there is sufficient evidence to substantiate that the detailed level can improve the accuracy of the effort estimate. More specifically, through these small case studies, it can be observed that, as the structural size increases (i.e. the sequence manipulation increases), it is reasonable to expect that more development effort will be required. Thus, both the functional and structural size measurements can be useful for a more comprehensive quantitative comparison of functional processes based on both their functional and structural requirements.

In spite of its small size, we believe that the experimental evaluation shows the applicability of our measurement method as well as its sound design. Indeed, the unsound design of a measurement method influences its application as well as the use of the

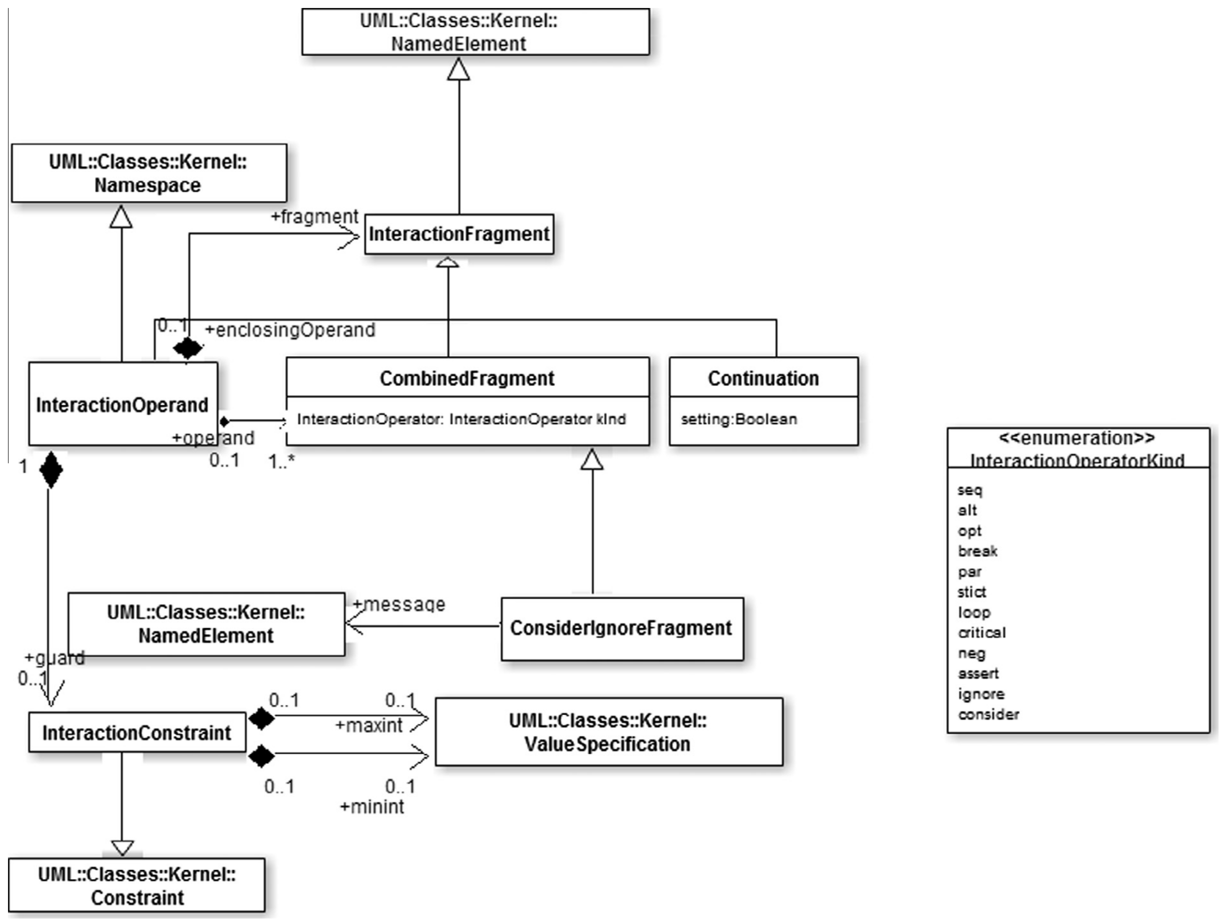


Fig. A. Combined fragments Package – from [OMG].

measurement results. In our case, this threat was mitigated by following the design of a measurement method from the measurement context model proposed by Abran [1]. In addition, we used the terminology related to the VIM metrology concepts [22] to avoid problems related to subjectivity.

The generalizability of the herein presented results may be limited: on the one hand, the number of case studies is small, and on the other hand this research may not have captured all aspects of the combined fragments of a sequence diagram. Hence, a larger set of case studies is needed to be able to generalize the findings.

7. Discussion and future work

The objective of this paper was to design an improved measurement of the size of the UML sequence diagrams by taking into account the data manipulations represented by the control structure of the sequence diagram, which was referred to as their structural size. This paper has proposed measurement procedures for sizing sequence diagrams at two distinct levels of granularity based on the UML2.0 sequence diagrams:

- The classical measurement of the functional size with the COSMIC international standard – ISO 19761 method measures at a high level of granularity.
- The measurement of the functional control structure of the sequence diagram is at a finer level of granularity.

Three small case studies illustrated that as the structural size increases (i.e. the sequence manipulation increases), it is reasonable to expect that more development effort will be required. However, experimentally investigating such a size-effort relationship was outside the scope of this paper therefore, further research is needed including collecting effort data from completed projects and measuring both the functional size and the structural size of the software requirements developed for such projects.

The design of the proposed measurement method and related measurement procedure was defined according to [1]: this allowed for the identification of the measurable concepts and constructs, the measurement principle and the description of the numerical assignment rules.

Thanks to the proposed method, the sequence diagram size can be measured with reference to its functionality in terms of COSMIC Function Points (CFP units) and structurally in terms of Control Structure Manipulations (CSM units): on the one hand, COSMIC CFP provides the functional size based on the data movements in a sequence diagram at a high level of granularity; on the other hand, the CSM provides the structural size based on the sequence manipulations in a sequence diagram at a finer level of granularity.

These sizes, at different levels of granularity, have distinct measurement units. They represent complementary aspects of the size of software and could be used as distinct variables in estimation models. Further empirical research will be needed to quantitatively determine their contribution to the size-effort relationship in various empirical contexts.

Furthermore, changes to functional requirements which affect only the sequence manipulations and not the data movements can now be captured quantitatively with CSM units and the impact of these changes on estimating software development effort can now be accounted for.

Of course, the proposed measurement procedure has been described as a manual procedure, and this can be time-consuming and error prone. To improve the speed and precision of measurement, further work is required to extend UML CASE tools to support automation of the measurement of both the functional and the structural sizes of the sequence diagrams.

In addition, to offer comprehensive measurement for planning and controlling software project management, further research work is also needed to explore relevant measurement concepts within the other UML diagrams.

Finally, further exploration of the way in which measurements and their units differ at these two levels of granularity (i.e. CFP and CSM) is required, in order to gain additional insights into their similarities and differences. This knowledge will make it possible to identify the conditions under which they could be combined through valid mathematical operations representing complementary quantification of the pieces of software being measured.

Appendix A

See Fig. A.

References

[1] A. Abran, Software Metrics and Software Metrology, John Wiley & Sons, Inc. IEEE Computer Society Press, 2010.

[2] B.W. Boehm, Software Cost Estimation with COCOMO II, Prentice Hall, 2000.

[3] P.R. Hill, Practical Software Project Estimation: A Toolkit for Estimating Software Development Effort & Duration, McGraw-Hill, Osborne, 2010.

[4] ISO/IEC19761, Software Engineering – COSMIC: A Functional Size Measurement Method, International Organization for Standardization, ISO, Geneva, 2011.

[5] ISO/IEC20926, Software and Systems Engineering – Software measurement – IFPUG Functional Size Measurement Method Geneva, International Organization for Standardization, ISO, 2009.

[6] ISO/IEC20968, Software Engineering – Mk II Function Point Analysis – Counting Practices Manual, International Organization for Standardization, ISO, Geneva, 2002.

[7] ISO/IEC24570, Software Engineering – NESMA Functional Size Measurement Method Version 2.1 – Definitions and Counting Guidelines for the Application of Function Point Analysis, International Organization for Standardization, ISO, Geneva, 2005.

[8] ISO/IEC29881, Information Technology – Software and Systems Engineering – FiSMA 1.1 Functional size Measurement Method, International Organization for Standardization, ISO, Geneva, 2008.

[9] ISO/IEC14143-1, Information Technology – Software Measurement-Functional Size Measurement–Part 1: Definition of Concepts, International Organization for Standardization, ISO, Geneva, 2007.

[10] OMG, Documents Associated With Unified Modeling Language (UML), V2.4.1, 2011. <http://www.omg.org/spec/UML/2.4.1/>.

[11] V. Bévo, G. Lévesque, A. Abran, Application de la méthode FFP à partir d'une spécification selon la notation UML, in: 9th International Workshop Software Measurement (IWSM 1999), Lac Supérieur, Canada, 1999.

[12] K.v.d. Berg, T. Dekkers, R. Oudshoorn, Functional size measurement applied to UML-based user requirements, in: 2nd Software Measurement European Forum (SMEF 2005), Rome, Italy, 2005, pp. 69–80.

[13] A. Sellami, H. Ben-Abdallah, Functional size of use case diagrams: a fine-grain measurement, in: 14th International Conference on Software Engineering Advances (ICSEA 2009), Porto, Portugal, 2009.

[14] L. Lavazza, G. Robiolo, Introducing the evaluation of complexity in functional size measurement: a UML-based approach, in: International Symposium on Empirical Software Engineering (ESEM 2010), Bolzano-Bozen, Italy, 2010.

[15] S. Nagano, T. Ajisaka, Functional metrics using COSMIC-FFP for object-oriented real-time systems, in: 13th International Workshop on Software Measurement (IWSM), Montreal, Canada, 2003.

[16] L. Lavazza, V.D. Bianco, A case study in COSMIC functional size measurement: the rice cooker revisited, in: International Conference on Software Process and Product Measurement (IWSM'09/Mensura'09), Amsterdam, The Netherlands, 2009.

[17] G. Lévesque, V. Bévo, T.H. Cao, Estimating software size with UML models, in: C3S2E conference (C3S2E '08), Montreal, Canada, 2008.

[18] A. Sellami, M. Haoues, H. Ben-Abdallah, Analyzing UML activity and component diagrams: an approach based on COSMIC functional size measurement, in: 8th International Conference on Evaluation of Novel Software Approaches (ENASE '13), Angers, France, 2013.

[19] J. Gabay, D. Gabay, UML 2 Analyse et conception, Dunod, France, 2008.

[20] P. Roque, UML2 pour la pratique, Eyrolles, 2009.

[21] A. Abran, J.-M. Desharnais, S. Oligny, D. St-Pierre, C. Symons, COSMIC Measurement Manual – Version 3.0.1, – Measurement Manual – The COSMIC Implementation Guide for ISO/IEC 19761: 2003, COSMIC Group, 2009.

[22] VIM, International Vocabulary of Metrology – Basic and General Concepts and Associated Terms (VIM), Geneva, International Organization for Standardization, ISO, 2007.

[23] N.E. Fenton, S.L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, second ed., PWS Publishing Co, London, 1997.

[24] A. Parada, E. Siegert, L.d. Brisolara, Generating Java code from UML Class and Sequence Diagrams, in: Workshop de Sistemas Embarcados, Florianópolis, 2011.

[25] A. Ben-Abda, Modélisation d'un système d'information d'une compagnie aérienne, University of Sfax, Tunisia, 2008.