Design and automation of a COSMIC measurement procedure based on UML models

Gabriele De Vito[1] · Filomena Ferrucci[1] · Carmine Gravino[1]

"The final publication is available at link.springer.com". https://link.springer.com/article/10.1007/s10270-019-00731-2

Abstract

Context. Many organizations are adopting the COSMIC method to size software products for estimating and controlling their development costs and performances. Using a functional size measurement method requires specialized expertise and can be time-consuming. *Objectives*. Since UML is the de facto industrial modeling language standard for object-oriented systems, it is very useful to understand how to exploit UML models for measuring software systems and for developing tools that can automatically derive the COSMIC size from them. This paper provides an answer to these needs. *Method.* We present a measurement procedure to derive the COSMIC functional size from UML software artifacts and a tool, named J-UML COSMIC, for the automation of the procedure. Based on the observation that different development processes are characterized by the use of different UML models, the tool has been designed to work with different UML artifacts (such as use case models, package diagrams, component diagrams, class diagrams, activity diagrams, and sequence diagrams) and to adapt to the specific employed process. To assess the measurement procedure and J-UML COSMIC, we have carried out two case studies and compared the measurement results provided by the tool with the ones obtained by experts applying the standard COSMIC method. *Results.* Using the proposed measurement procedure the tool is able to identify from UML software models all the COSMIC concepts and data movements identified by the experts. Moreover, the tool allows us to obtain incremental accurate measurements when new models are considered or existing ones are detailed. *Conclusions*. The designed approach is able to automatically measure the functional size starting from UML artifacts and providing higher accurate results when more data is available.

Keywords Functional size measurement · Automation tool · COSMIC-ISO 19761 · Unified modeling language

1 Introduction

Functional size measurement (FSM) methods measure the software size by quantifying the "functionality" provided to the users and are nowadays widely applied in the industrial field for many purposes. Indeed, functional size can be exploited to estimate the effort required to develop a software system, that is a key factor for making a bid, planning the development activities, allocating resources adequately, and so on [1]. Furthermore, functional size can be used for benchmarking [2], comparing business processes and performance metrics to industry best practices from other companies, for portfolio management, enabling measurement and objective evaluation of investment scenarios [3], and for analyzing the performance of software contractors.

Function point analysis (FPA) [4] was the first FSM method introduced in 1979 for measuring transactional systems. Since then, several variants of the method have been provided (e.g., MarkII and NESMA) with the aim of improving the measurement procedure or extending their applicability to different domains [5]. FPA and its variants can be considered as the first generation of FSM methods, distinguishing them from COSMIC [7], which is considered a second-generation FSM method, due to several specific characteristics. COSMIC was the first FSM approach designed to comply with the standard ISO/IEC14143/1 [6]. Based on fundamental principles of measurement theory and software

Communicated by Dr. Alan Hartman.

Carmine Gravino
gravino@unisa.it

Filomena Ferrucci
fferrucci@unisa.it

[1] University of Salerno, Via Giovanni Paolo II, 132, Fisciano, (SA), Italy

engineering, COSMIC was also defined to be suitable for a broader range of application domains.

From the software development point of view, unified modeling language (UML) is the de facto industry standard modeling language for object-oriented systems [22]. UML is used to describe analysis and design solutions in a concise way, understandable to a wide audience. Consequently, applying the COSMIC method to UML artifacts has attracted several researchers with the common goal of automating the COSMIC measurement procedure, e.g., [8–20,23,25–34]. These investigations are motivated by the fact that functional size measurement is still a manual process which turns out to be time-consuming and needs specialized experts. Functional size measurement automation is really a challenge when the artifacts to measure are expressed as textual descriptions of user requirements. However, when requirements are captured by models the measurement automation becomes feasible.

In the literature, we find two kinds of proposals for measurement automation: those designed for specific model-driven approaches (e.g., [35]) and those based on some UML models (e.g., [9]) in some cases specialized for specific domains (e.g., [23]). This paper focuses on UML models and aims to advance the existing approaches providing a general purpose solution characterized by the following characteristics. First of all, it is applicable to any domain (not specific to a single domain, as for example Web applications [19]). Moreover, it is able to adapt to any development process based on standard UML models. To this aim, we wide the sets of models taken into account in the literature and provide "escalation" rules for exploiting the models that are used in the adopted development process. The approach proposed in this paper comprehensively takes advantage of the proposals made so far and introduces rules for activity diagrams. Furthermore, it is able to provide incremental accurate measurements when new models are considered or existing ones are detailed. Finally, the procedure is supported by a software tool that enables:

1. automatic identification of the main COSMIC elements (functional users, triggering events, objects of interest, data groups, and functional processes);
2. automatic measurement of UML models;
3. recording the results of the textual requirements analysis (such as use cases, scenarios, user manual, etc.), of the UML models analysis, and of the automatic measurement (e.g., identified functional processes) to create a database to be exploited for benchmarking and controlling.

To evaluate the measurement procedure and the tool, we have considered two case studies and compared the measurement results obtained with the tool with those obtained by experts.

The paper is organized as follows. Section 2 summarizes the main concepts of COSMIC functional size, while Sect. 3 reports on the main related work highlighting features and shortcomings of the existing approaches and summarizing the characteristics of the proposal. Section 4 presents the proposed COSMIC measurement procedure. Section 5 provides the description and the design of the functional size measurement automation tool. Section 6 presents the case studies carried out to evaluate the proposed measurement procedure and tool. Section 7 provides conclusions and several directions for future work.

2 COSMIC

The COSMIC approach is composed of a set of models, principles, rules, and processes applicable to the functional user requirements (FURs) of a given piece of software [24]. In the following, we recall the main aspects of the approach as defined in the COSMIC Measurement Manual [24]. In particular, the Software Context Model introduces the principles and the concepts needed to identify the FURs of the piece of software to be measured. The Generic Software Model has to be applied to the FURs to identify the components of the functionality that will be measured. The concepts of the Software Context Model are the following:

- The **Purpose** of a measurement defines why a measurement is required.
- The **Scope** of a measurement selects the set of FURs to be included in a specific functional size measurement exercise.
- The **FURs** describe what the software will do in terms of tasks and services.
- A **Layer** is a partition resulting from the functional division of a software architecture.
- A **Functional user** is a (type of) user that is a sender and/or an intended recipient of data in the FURs.
- The **Boundary** is defined as a conceptual interface between the software being measured and its functional users.

The following concepts of the Generic Software Model are applied to the FURs of each separate piece of software:

- **A Functional Process** is an elementary component of a set of FURs comprising a unique, cohesive, and independently executable set of data movements. It is triggered by a data movement from a functional user that informs the piece of software that the functional user has identified a triggering event. It is complete when it has executed all that is required in response to the triggering event. Each functional process consists of (a set of) sub-processes.

- A **Functional Sub-Process** may be a data movement (i.e., Entry, Exit, Read, and Write).
- A **Triggering Event** is an event that causes a functional user of the piece of software to initiate (trigger) one or more functional processes.
- A **Data Group** is a distinct, non-empty, unordered, and non-redundant set of data attributes describing a complementary aspect of the same object of interest.
- A **Data Attribute** is the smallest piece of information, within an identified data group, carrying a meaning from the perspective of the software's FURs.

In the measurement phase, the data movements (i.e., Entry, Exit, Read, and Write) of each functional process have to be identified and used to obtain a size measurement of the software. They are defined in [24] as follows:

- An **Entry (E)** data movement moves a data group from a functional user across the boundary into the functional process where it is required.
- An **Exit (X)** data movement moves a data group from a functional process across the boundary to the functional user that requires it.
- A **Read (R)** data movement moves a data group from the persistent storage within each of the functional process that requires it.
- A **Write (W)** data movement moves a data group lying inside a functional process to the persistent storage.

Each data movement is counted as 1 CFP (COSMIC Function Point), the COSMIC measurement standard. Thus, the size of a piece of software within a defined scope is obtained by summing the sizes of all the functional processes identified. The minimum size of any functional process is 2 (1E and 1W), when nothing is shown to the functional user. For more details about the COSMIC method, readers are referred to the COSMIC Measurement Manual [24].

3 Related work

In this section, we summarize the main related work. In particular, Sect. 3.1 focuses on existing COSMIC measurement procedures for UML models. In Sect. 3.2 we briefly recall the COSMIC measurement procedures proposed for model-driven development. Subsection 3.3 aims to illustrate some approaches that consider different levels of granularity by relating the abstraction levels of the measured artifacts and functional size, as this aspect is one of the main contributions of the approach presented in this paper.

3.1 COSMIC measurement procedures for UML models

We start by illustrating the main related work; then, we summarize the characteristics and shortcomings of each approach and discuss the main contributions presented in this paper.

A mapping between the COSMIC concepts and UML is introduced in [8]. In particular, the COSMIC concept of functional process is mapped onto the UML use case. The (types of) data movements (Read, Write, Entry, or Exit) are considered UML scenarios. Data groups are mapped onto UML classes in class diagrams, and data attributes onto the attributes of these classes. The software boundary is represented by a use case diagram, and the COSMIC concept of user as a UML actor. Finally, two COSMIC concepts have no mapping onto UML, namely layer and triggering events. The application of the proposed mapping rules is illustrated in [8] through a case study of a Management Information System (MIS), whose functional size is first computed exploiting use cases and then using scenarios. It is worth mentioning that the two functional measurements lead to different results, due to different levels of granularity of the measured artifacts. The approach presented in [8] is refined in [9] using sequence diagrams. A functional process is mapped onto a sequence diagram, and for each message in the diagram a data movement is introduced.

A process to automate the counting of the functional size using a CASE tool such as Rational Rose [26] was first proposed in [25]. In the same line, [10] proposes to automate the COSMIC functional size measurement using the Rational Unified Process [21] and Rational Rose to implement it. Based on the results of [8] and [9], the authors establish a mapping between the RUP concepts and the COSMIC concepts. The difference lies mainly in the use of a new UML stereotype to map the concept of triggering event. However, the concept of layer, having no correspondence in UML, must be manually identified. The proposed tool was assessed on a single case study.

In [23], the authors propose a system called μcROSE, which is based on the use of statechart diagrams and calculates the functional size of real-time systems. However, the work is heavily dependent on hard-coded rules to map the different objects of interest onto the COSMIC ones. The authors also present a brief review of their work carried out by an expert, who tested their system using a single case study, and show different measurement errors. The authors of [11] present a prototype (MetricXper) that automates the measurement starting from specifications created with Rational Rose [27]. This prototype uses XMI files to export the MOF (Meta Object Facility) meta-models and stores all the related measurement concepts in a specific database.

The authors of [12] propose a tool to measure the COSMIC functional size of embedded software using a new UML

profile that captures all the necessary information to apply COSMIC and to estimate the code size [13].

In [14], UML use case models are exploited to support the COSMIC measurement. The authors argue that, regardless of the complexity of data groups, it is more interesting to base the measurement process on data attributes. They present an approach based on the use of detailed scenarios to simplify functional size measurement. They also highlight the redundancy problem and suggest to use inclusion, extension, and inheritance relationships of use case models in order to avoid repetitions during the measurement procedure.

The approach proposed in [15] exploits use cases and UML sequence diagrams to measure the functional size. It takes into account not only data movements but also data manipulations. In this way, it becomes easier to calculate the functional size by summing all the messages in the sequence diagrams. There seem to be no tools to implement this procedure and adding data manipulations to the measurement imply a great deal of manual work.

A specific technique to measure UML use case diagrams is presented in [16]. The authors assert that use case diagrams represent functional processes, since they describe the behavior of the software from the point of view of the functional users. The authors use a specific format to document the use cases, and the total size of the use case diagram is given by the sum of the functional size of each scenario composing it. A set of rules is applied to take into account the inclusion, extension, and inheritance relationships between use cases. The authors of the proposal claim that UML sequence and class diagrams are related to the use case diagram. This implies that the proposed model to measure the functional size of use case diagrams can also be applied to measure UML sequence and class diagrams.

Under the assumption that UML allows us to specify user requirements, [17] proposes a "requirements" space to show that FURs can be supported by UML at different levels of granularity. This space consists of definitions of a number of UML diagrams that are consistent at each level. In this proposal, actors and system boundary can be specified using the use case diagrams. Moreover, class diagrams are used to measure data structures and attributes. Finally, activity diagrams are exploited to represent functional processes based on the concept of flow of events. The measurement phase is based on these diagrams which describe the different use cases. For each use case, the authors of the proposal show how to calculate the number of the data movements, but they do not make clear the rationale for the choice of this kind of diagrams.

In [28], the authors present step-by-step guidelines to manually derive UML artifacts, such as use case diagrams and sequence diagrams, from the requirements and then apply a set of rules to measure them using COSMIC. However, no details of the tool used to automate the counting procedure are provided.

In [18], the authors discuss how UML can be used to support the COSMIC method. The proposed approach is illustrated through a case study belonging to the embedded and real-time domain (the "rice cooker controller" case, originally designed by the COSMIC group [29]). The authors also use a mapping between all the COSMIC concepts and the corresponding UML concepts. In particular, use case diagrams are used to represent the boundary of the software to measure, business users and other external elements exchanging data with the application. Moreover, component diagrams are used to identify persistent data groups and interfaces between the application and external components. The COSMIC concept of a triggering event is mapped onto a UML element within the component diagram. The proposal highlights that that sequence diagrams are important to specify functional processes and data movements and shows how exploiting use case and component diagrams can help to identify non-functional users. The authors conclude that UML can be used to build models that are relatively easy to measure. Lavazza and Bianco [18] also demonstrate the independence of problem and solution domains through the rice cooker controller case but only from a theoretical point of view. Another study by Ungan and Demirors relates abstraction levels of the measured artifacts and functional size [30]. They start from the observation that none of the functional size measurement methods proposed in the literature explicitly position themselves in problem or solution domains and propose an approach to distinguish problem and solution domains for a software project. In particular, they present a software size measurement methodology for the solution domain which is based on sizes determined from software design models. In particular, the proposed approach exploits sequence diagrams. A tool automating the measurement is also proposed.

A COSMIC approach aimed at predicting the effort to develop Web Applications is described in [19]. The method focuses on counting data movements and captures the specific aspects of dynamic Web applications, which are characterized by data movements exchanged with Web servers. The method is based on two measures to be applied to analysis and design documents aiming at providing estimates in the early stage. In the analysis phase, the authors adopt the approach described in [7,8], while, to determine the size from the design documents, they extend the proposal presented in [31] and [32] and define a set of rules that allow us to measure Web applications using the information contained in the class diagrams. The diagrams exploit stereotypes, constraints, and tagged values to indicate the components that are specific to the Web as suggested in [33].

A literature survey of papers that studied the application of COSMIC functional size measurement using UML models is presented in [34]. This work also attempts to propose

a framework for supporting the automation of measurement, mainly based on the use of a specific UML profile for defining functional size measurements rules. The underlying idea is that software engineers annotate their UML models with information pertaining to COSMIC functional size measurement. However, the authors of the proposal do not provide details and declare their intention to fully specify their idea in ongoing research [34].

With the aim of summarizing contributions as well as shortcomings of the proposals described above, for each paper Table 1 shows the following information:

- the focus of the proposal;
- the UML artifacts taken as input;
- the UML artifacts measured;
- the COSMIC concepts not mapped onto UML models;
- whether a tool for automation is provided;
- whether a validation is performed;
- whether different levels of granularity are considered.

In the table, we have also added an entry characterizing the present proposal to highlight the differences with respect to the existing approaches. As we can see, many of the existing proposals focus on use cases and sequence diagrams, while a few exploit class diagrams and component diagrams.

None of the existing studies take into account the fact that different development processes can exploit different models. Similarly, different models could be available in different phases of the development process. Thus, the COSMIC measurement procedure should be able to adapt to different processes and phases. As an example, if class diagrams are available the measurement procedure could exploit them to identify the objects of interest with higher precision. If class diagrams are not available the measurement procedure could exploit other diagrams taking into account their level of granularity, i.e., first the sequence diagrams and if they are not available the activity diagrams. The majority of the approaches described above focus on the measurement starting from given UML artifacts produced in a given software development phase. Moreover, just a few papers provide an automated approach to calculate the functional size in terms of COSMIC and even fewer allow us to apply the measurement procedure to different domains.

Finally, we observe that the results of the application of the proposed COSMIC measurement procedures, and especially how these results have been validated, are not provided in a satisfactory manner.

We have designed the proposed approach and tool (J-UML COSMIC) with the aim of overcoming the shortcomings of previous proposals. Thus, our proposal is characterized by the following features:

- complete mapping of COSMIC concepts onto UML models,
- automation of the measurement procedure,
- incremental refinement of measurement exploiting UML artifacts at different levels of granularity,
- validation of the measurement procedure through case studies based on a comparison of measurements performed by experts,
- applicability to different UML artifacts from different domains.

These features are synthesized in the last entry of Table 1.

3.2 COSMIC measurement procedure for model-driven development approaches

Model-driven development methods are characterized by specific visual notations and model transformation rules. The visual notations are often inspired by UML and are used to design software applications through suitable conceptual models. The transformation rules are used to automatically generate the corresponding source code starting from the designed conceptual models using a development tool. It is worth noting that the model-driven approaches are characterized by a low granularity level since details in the conceptual models are needed to generate software applications. Some COSMIC measurement procedures have been proposed for different model-driven approaches focused on different types of applications (e.g., Web applications, real-time applications, MIS). In particular, [35] proposes the OO-Method COSMIC Function Points (OOmCFP) procedure to automate the functional size measurement of object-oriented MIS applications generated transforming OO-M conceptual models. The paper also presents a tool that supports the proposed procedure. A COSMIC-based procedure for Web applications modeled with Hypermedia Object-Oriented (OO-H) method is proposed in [36] and [37]. The authors define mapping rules to automatically derive the functional size of OO-H models. They show that a correct mapping between the COSMIC concepts and those of the models used together with suitable measurement rules are fundamental elements to create a COSMIC-based procedure for model-driven development approaches. The authors of [38] and [39] present a COSMIC procedure for sizing embedded and real-time software when requirements are expressed using Simulink [40]. The procedure is based on the mapping between the COSMIC concepts and those of the Simulink model and on the identification of rules to extract relevant information stored in Simulink files. Thus, it provides the basis for automating the measurement of FURs documented with Simulink.

Table 1 Summary of existing COSMIC measurement procedures based on UML models

Ref.	Focus	Input UML artifacts	UML artifacts measured	COSMIC concepts not mapped onto UML	Automated?	Validated?	Doma
[8]	Mapping between COSMIC concepts and UML	Use cases, scenario, class diagrams	Use cases, scenarios	Layer, peer components, triggering events	No	No	Gene
[9]	Mapping between COSMIC concepts and UML, and granularity aspect of the uses cases proposed in [8]	Use cases, sequence diagrams, class diagrams	Sequence diagrams	Layer, peer components, triggering events	No	No	Gene
[25]	Automation of the counting. However, there is no details of the mentioned tool. It uses the mapping described in [9]	Use cases, sequence diagrams, class diagrams	Sequence diagrams	Layer, peer components, triggering events	Yes	Yes	Gene
[10]	Automated counting of the size at different level of granularity	Use cases, scenario, detailed scenarios	Use cases, scenario, detailed scenarios	Layer, peer components, triggering events	Partial	Partial	Gene
[12]	How to model the COSMIC Generic SW Model using UML (and automated counting) and how to use COSMIC Function Points to estimate the software code size of software components	Component diagrams, class diagrams	Component diagrams	Layer, peer components, triggering events	Partial	Yes	Embe

Table 1 continued

Ref.	Focus	Input UML artifacts	UML artifacts measured	COSMIC concepts not mapped onto UML	Automated?	Validated?	Domain
[14]	Resolving the functional redundancy issue by appropriate use case organization providing a template for use case specification useful to count the functional size	Use cases	Use cases	No mapping	No	No	General
[15]	Automated counting of the size	Use cases, sequence diagrams	Sequence diagrams	No mapping, except for functional process, functional user, and data movement	Yes	No	General
[16]	Fine-grain measurement for the UML use case diagrams, sequence diagrams and class diagrams	Use cases, class diagrams, sequence diagrams	Use cases, class diagrams, sequence diagrams	No mapping, except for Data Movements	No	No	General
[17]	Fine-grain measurement for the UML use case diagrams, sequence diagrams, and class diagrams	Use cases, class diagrams, activity diagrams	None	No Mapping	No	Yes	General
[18]	How UML can be used to support the COSMIC method	Use cases, component diagrams, sequence diagrams	Sequence diagrams	Layer, peer components	No	Partial	Embedded and R

Table 1 continued

Ref.	Focus	Input UML artifacts	UML artifacts measured	COSMIC concepts not mapped onto UML	Automated?	Validated?	Do
[19]	Predicting Web application development effort	Use cases, sequence diagrams	Sequence diagrams	Same as [9,27,32]	No	Yes	W
[23]	Automatically measuring the functional software size for Rational Rose Real-Time model (RRRT), using COSMIC-FFP	Class (RRRT concepts), extended finite state machine	Extended finite state Machine	Triggering event, functional users	Yes	Yes	Re
[28]	Defining a COSMIC procedure to estimate the size of OO systems from high-level specifications using OO-Method Requirement Model	Use case, sequence diagrams	Sequence diagrams	Peer components, Triggering event	Yes	Yes	Ge
[34]	Design of a specific UML profile to support COSMIC measurement using UML models	None	None	No mapping	No	No	Ge
Current proposal	Defining a COSMIC measurement procedure from UML models and its automation	Use cases, package diagrams, component diagrams, class diagrams, activity diagrams, sequence diagrams	Use cases, package diagrams, component diagrams, class diagrams, activity diagrams, sequence diagrams	None (all the COSMIC concepts are mapped)	Yes	Yes	Ge

3.3 Focus on granularity levels for functional size measurement

As we can see from Table 1, a few proposals for deriving COSMIC size have been concerned with different model granularity levels. In this subsection, we recall some approaches that relate abstraction levels of the measured artifacts and functional size even though they do not focus on COSMIC. In particular, a few works have addressed the issue of applying FPA to FURs with different levels of granularity [46].

In [47], the authors highlight that the choice of a suitable granularity level of the requirements should be carried out based on the goal of the requirements specification, e.g., providing a broad vision of the software (in the earlier phase to identify the scope) or providing a detailed vision of the software (to develop it). An example of possible levels of granularity are those introduced by [50]. The author asserts that the interactions in a use case scenario can be broken up into smaller and smaller interactions, representing an actor attempting to achieve a goal. So, it is natural to break up goals into smaller and smaller goals. According to [50], the level of goals are strategic, user, and subfunction. Bearing this in mind, applying Function Points Analysis (FPA) means to measure requirements which are refined in terms of finer subfunction goals.

The authors of [48] describe how some UML diagrams can be used for determining the number of Function Points at different levels. In particular, high-level use cases and class diagrams can be exploited for calculating the number of Function Points applying the Indicative FPA approach, which is an approximation of the FPA method [49]. On the other hand, expanded use cases, activity diagrams, interactions diagrams, and system-level sequence diagrams can be used for calculating the number of Function Points applying the High-level FPA approach, which is another approximation of the FPA method [49]. Finally, detailed use cases, activity diagrams, interaction diagrams, sequence diagrams/collaboration diagrams, and class diagrams can be used for calculating the number of Function Points with the standard FPA method.

4 Design of the measurement procedure

In the following, we present the set of mapping and measurement rules for the proposed COSMIC measurement procedure of UML software requirements. This procedure is applicable to different UML artifacts which can have different levels of granularity (such as use case models, package diagrams, component diagrams, class diagrams, activity diagrams, sequence diagrams). This flexibility allows us to refine the measurement when new models are introduced or existing ones are detailed. Table 2 shows, for each COSMIC element, the priority order considered for the UML artifacts when applying the proposed measurement procedure and tool. So, for example, a persistent data group can be identified by a class diagram or by a sequence diagram when the class diagram is not available. Finally, if no class and sequence diagrams are available the measurement procedure can exploit activity diagrams.

4.1 Mapping rules

Table 3 summarizes the mapping rules we defined to map each COSMIC concept onto UML elements and diagrams. They were defined taking into account the COSMIC Software Context Model and the Generic Software Model. In the following, we provide the description of these rules and the rationale behind them.

Rules R1–R5, R16, and R25 allow identifying the COSMIC concept of a layer, if the system is organized in layers. On the other hand, if the measurement scope contains peer components, we need to apply Rules R6, R17, and/or R26 to identify these components. The applicability of these rules depends, of course, on the available artifacts, namely: component diagrams, package diagrams, and use cases. Whenever the component diagrams that model the logical architecture of the system exist, we can identify layers and peer components using them.

One of the most widely used patterns to define the logical architecture is the pattern layer, which can be realized using the component diagrams. Each level in a diagram is a COSMIC layer (Rule R25), and each component within a layer can be considered a peer component (Rule R26). An alternative representation of this pattern (and of this kind of architecture) is obtained using package diagrams. In these diagrams, UML packages represent the layers, and unidirectional dashed arrows represent the relationships between them (Rule R16). Each package in a given layer cooperates with other packages within the same layer without hierarchical dependencies; in other words, they can be considered peer components (Rule R17). However, if there are no package or component diagrams, or it is not possible to identify layers or peer components, we need to analyze use case models. Rule R1 maps the COSMIC concept of a layer onto the Subject of a use case model. In fact, use cases describe the system at any level. For each set of use cases that specifies a particular system level, we can define a specific context: the Subject. Each level of the system has its own Subject and its own set of actors with which it interacts and is contained in a use case model. Each use case is related to a specific system level and to a specific Subject; therefore, it corresponds to one or more use cases that concern a lower system level in the hierarchy (Rule R2). The relationship between the use cases that belong to a higher level (“superordinated”), and those

Table 2 COSMIC concepts and UML artifacts exploited

COSMIC concept	UML artifacts priority when applying the measurement procedure
Layer	Package, component diagram use cases
Peer component	Package, component diagram use cases
Functional user	Component diagram, sequence diagram, activity diagram, use cases
Boundary	Component diagram, use cases
Triggered event	Component diagram, sequence diagram, activity diagram, use cases
Persistent data group	Class diagram
	Sequence diagram
	Activity diagram
Transient data group	Class diagram, component diagram
	Sequence diagram
	Activity diagram
Data attribute	Class diagram
Object of interest	Class diagram
	Sequence diagram
	Activity diagram
Functional process	Class diagram
	Sequence diagram
	Use cases
	Activity diagram
Data movement	Sequence diagram, activity diagram

belonging to lower layers ("subordinated"), is a "refinement" relationship (Rule R3). Functional services such as database management systems, operating systems, or device drivers should normally be considered to be located in separate layers (Rule R4). If it is not possible to establish partitions, we can consider the system composed of a single layer (Rule R5).

Rules R7, R8, R27, R33, R34, and R49 allow identifying the COSMIC concept of a functional user. In particular, we considered as functional users, all the actors associated with the use cases in a use case diagram (Rule R7). They are outside the system and can be administrators, humans, hardware controllers, other pieces of software, etc., but the operating system cannot be considered a functional user (Rule R8). If there is a component diagram that refines the use cases by introducing the groups of data managed by the application, and making the interfaces between the application and the external elements explicit, then we can identify the functional users by means of Rule R27. This rule identifies the functional users as the components of this diagram, outside the application boundary and directly connected to the system, using the realized interfaces. Rules R33–R34 can be applied to the activity diagrams, if they refine a use case diagram. Rule R33 identifies swimlanes that begin the functional flow as functional users. Rule R34, instead, identifies timers and clock ticks (timed batch process, periodic signals, timers, etc.) triggering the functional processes as functional users. Finally, Rule R49 can be applied to the sequence diagrams, refining a specific use case diagram. This rule maps the COSMIC concept of a functional user onto the actor/agent that begins the sequence of messages in the sequence diagram.

Rules R9, R10, and R28 allow identifying the COSMIC concept of boundary. Rule R9 asserts that a boundary exists between the actors and their associated use cases. Moreover, a boundary exists between each pair of peer components within the same layer, and between each couple of layers (Rule R10). On the other hand, a boundary can be represented explicitly as a UML component when component diagrams are used to refine the use cases and to introduce a group of data managed by the application (Rule 28).

Rules R11, R32, and R48 map the COSMIC concept of level of granularity, while Rules R12, R29, R35, R36, and R50 allow identifying the COSMIC concept of triggering event. Rule R12 is applicable to use case diagrams and asserts that each association between actors and use cases is a candidate to be a triggering event. If there is a component diagram that refines a use case diagram, introducing the group of data managed by the application, we can map the COSMIC concept of triggering event onto the operations provided by the interfaces realized by the system and invoked spontaneously by an active external component (Rule R29). Rules R35–R36 are applicable to the activity diagrams which refine the use cases diagrams. In particular, Rule R35 maps the triggering event onto the starting node of this kind of activity diagram,

Table 3 Mapping rules

COSMIC concept	Rule	UML diagram/artifact	UML element
Layer	R1–R5	Use case	Model subject of the use case model
	R16	Package diagram	Package
	R25	Component diagram	Component
Peer component	R6	Use case model	Horizontal partition of the Subject
	R17	Package diagram	Package
	R26	Component diagram	Component
Functional user	R7–R8	Use case diagram	Actor
	R27	Component diagram	Component
	R33–R34	Activity diagram	Swimlane, accept time node
	R49	Sequence diagram	Actor/object boundary
Boundary	R9–R10	Use case diagram	Boundary between actors and associated use cases
	R28	Component diagram	Component
Level of granularity	R11	Use case model	Granularity of the use case
	R32	Activity diagram	Granularity of the activity diagram
	R48	Sequence diagram	Granularity of the sequence diagram
Triggering event	R12	Use case model	Association
	R29	Component diagram	Implemented interfaces
	R35–R36	Activity diagram	Start node, accept time node
	R50	Sequence diagram	Initial message
Data group	R19–R21	Class diagram	Class
	R30–R31	Component diagram	Operations of the interfaces or parameters of these operations, classes inside the application boundary
	R39–R40	Activity diagram	Object node with stereotype datastore
	R52–R53	Sequence diagram	Class/object with stereotype entity or table , group of parameters of a specific message
Data attribute	R22–R23	Class diagram	Attributes
Functional process	R13–R15	Use case model	Use case
	R24	Class diagram	Operations
	R37	Activity diagram	Activity diagram
	R51	Sequence diagram	Sequence diagram
Object of interest	R18	Class diagram	Class
	R38	Activity diagram	Object node

while Rule R36 maps it onto the Accept Time nodes. Finally, Rule R50 is applicable to the sequence diagrams and asserts that we can map the triggering event onto the first message in the diagram sent by the functional user to the system.

Rules R18 and R38 allow identifying the COSMIC concept of object of interest. Indeed, the class diagrams are the finest grained UML artifacts from which it is possible to precisely identify the objects of interest. If class diagrams are not available, we can identify the objects of interest in the activity diagrams (Rule R38).

Rules R19–R21, R30–R31, R39–R40, and R52–R53 can be used to map the COSMIC concept of data group. As with the objects of interest, the class diagrams are the best artifact to identity the data groups. Each class is a candidate to be a data group (Rule R19), except the abstract classes (Rule R20). Rule R21 states that each class labeled with the stereotype entity or the stereotype table is a candidate to be a persistent data group. If class diagrams are not available, we can map the data groups using the sequence diagrams (Rules R52 and R53). A further way to identify the data groups is the analysis of the activity diagrams (Rules R39–R40). In particular, a persistent data group is an object node whose outgoing flow corresponds to the incoming flow of an object node labeled with the stereotype datastore , since this represents the persistent data managed by the application (Rule R39). Otherwise, it is a transient data group (Rule R40). Finally, we can identify the data groups within the component diagrams (Rules R30–R31).

Table 4 Sequence diagram measurement rules

Data movements	Rule	Sequence diagram element
1 Entry (E)	R54	Message labeled with the stereotype signal , connecting the functional user and the System with input direction
1 Exit (X)	R54	Message labeled with the stereotype signal , connecting the system and the functional user with output direction
1 Write (W)	R55	Message labeled with the stereotype service
	R57	Message labeled with the stereotype connect
1 Read (R)	R56	Message labeled with the stereotype query

Data attributes are identified from the class diagrams since they are the finest grained UML artifacts to use for this purpose (Rules R22–R23). Rule R22 allows mapping the attributes of a class identified as an object of interest onto data attributes. Rule R23 states that each data group can contain only attributes belonging to the same object of interest.

At the end of the mapping phase, we need to identify the functional processes and the use case model to be used. A functional process lives in a use case and is triggered by an external event with respect to the system (Rule R13) and is identified from the functional user's point of view (Rule R14). In addition, each use case associated to the functional user is a candidate to be a functional process (Rule R15). Each sequence diagram (Rule R51) and each activity diagram (Rule R37) within the context of this use case can be considered a functional process. Moreover, each operation of a class is a candidate to be a functional process (Rule R24), in particular, the operations belonging to the realized interfaces.

4.2 Measurement rules

According to the COSMIC measurement method, the functional size of the software is given by the sum of all the data movements obtained for the detected functional processes, where each data movement counts as 1 CFP. Thus, we defined the set of measurement rules relating to "measurable" UML diagrams: sequence and activity diagrams.

4.3 Sequence diagram measurement rules

Sequence diagrams describe the steps of the functional processes and reflect the size of the dynamic capabilities of the software. In fact, a sequence diagram shows the internal view of a system for a Use Case. To assign a quantitative value that represents the size of a sequence diagram, we defined the set of measurement rules shown in Table 4. These rules allow us to identify the four kinds of data movements in a sequence diagram. In the following, we provide a description of these rules and the rationale behind them.

Rule R54 allows counting Entry (E) or Exit (X) data movements. Indeed, it refers to messages that represent interactions between an actor (external entity type) and the system. The application of this rule to the diagram shown in Fig. 1 produces 1E and 1X due to the messages "introduce_sale_data" and "show_total," respectively.

For Write (W) data movements, we provide two rules: Rule R55 and Rule R57. The former covers interactions where the receiver class represents an object that changes its state when the interaction occurs. Three kinds of changes can occur: (1) *new* when an object is created; (2) *update* when an object's state is modified; (3) *destroy* when an object is removed or destroyed. Rule 57 is used to establish a structural relation between the sender and the receiver objects in the interaction. As an example, by using Rule R55, 3W can be identified in the sequence diagram shown in Fig. 1 since it contains 3 messages ["create_sale()," "create_sale_line()," and "change_stock()"] labeled with the stereotype service . Another W can be obtained applying Rule R57, due to the message "select_one_item".

Rule R56 allows obtaining Read (R) data movements from messages used to represent queries on objects or classes. The application of this rule to the diagram shown in Fig. 1 allows us to count 2R due to the messages labeled "get_item_info()" and "get_sub_totals(),' respectively.

4.4 Activity diagram measurement rules

If there are no sequence diagrams modeling a given functional process, we can use the activity diagram modeling that process (if it exists) to assign a quantitative value that represents the size of an activity diagram. We defined the set of measurement rules shown in Table 5.

Like sequence diagrams, these rules allow us to identify the types of data movements in the activity diagrams, but unlike sequence diagrams, we introduced two new stereotypes entry and exit to identify Entry and Exit data movements, respectively.

Rules R41–R42 allow counting Entry (E) data movements. In particular, R41 refers to an action that creates a signal instance from its inputs and transmits it to the target object (in this case the system), where it may cause a state machine transition or the execution of an activity. The application of this rule to the diagram shown in Fig. 2 produces 1E, due to the Signal Send State labeled "submit account's data".

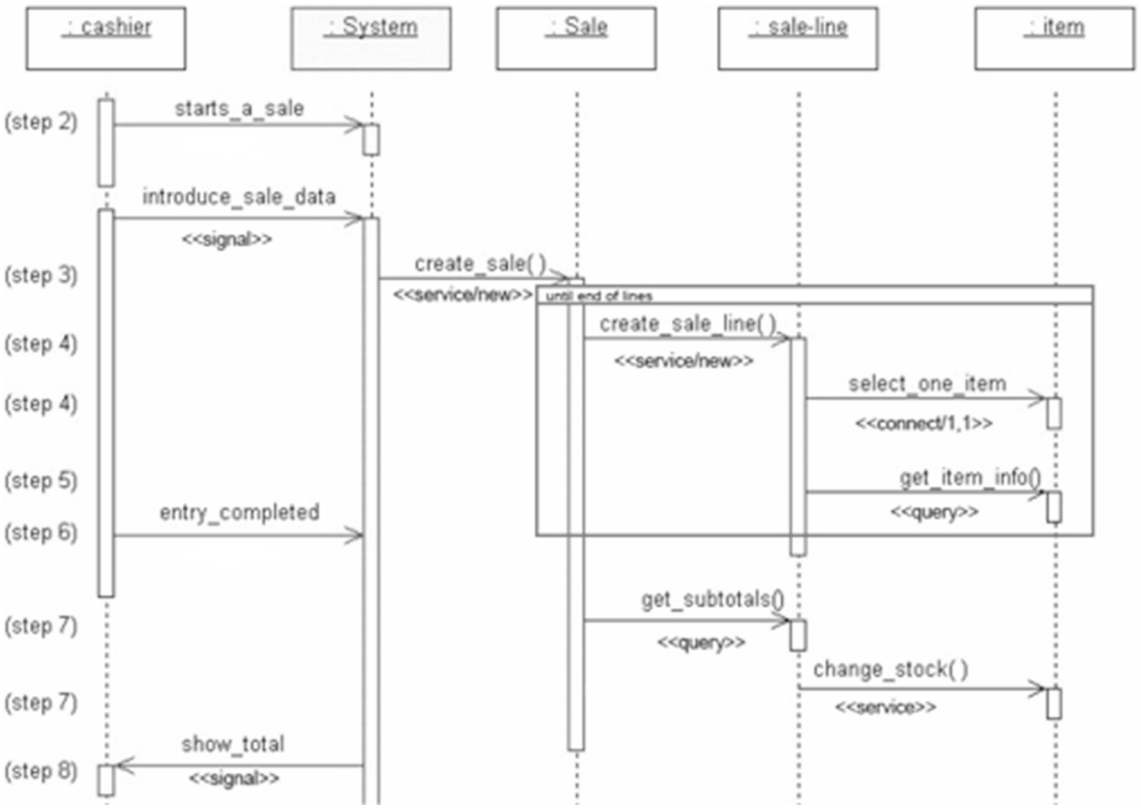


Fig. 1 Sequence diagram for sale items use case

Table 5 Activity diagram measurement rules

Data movements	Rule	Activity diagram element
1 Entry (E)	R41	Signal Send State node labeled with the stereotype entry , submitted by the functional user to the system
	R42	ActionState node labeled with the stereotype entry
	R43	AcceptTime node
1 Exit (X)	R44	Signal Send State node labeled with the stereotype exit , submitted by the system to the functional user
	R45	ActionState node labeled with the stereotype exit
1 Write (W)	R46	Incoming transition to an object node labeled with the stereotype datastore
1 Read (R)	R47	Outgoing transition from an object node labeled with the stereotype datastore

Dealing with ActionState node, the application of R42 to the diagram shown in Fig. 3 produces 1E since there is 1 ActionState node labeled with stereotype entry , namely "enter information for the professor". Furthermore, according to the COSMIC measurement method, R43 allows associating each clock tick or timing event with an Entry. So, from Fig. 4 we can obtain 1E by applying R43.

Analogously, Rules R44 and R45 allow counting Exit (X) data movements. The application of R44 to the diagram shown in Fig. 2 produces 2X since there are 2 Signal Send States in the diagram, namely "send success notification" and "send error notification," while R45 allows identifying 2X, due the ActionState nodes labeled with stereotype exit , "send error message" and "send new professor ID," respectively.

Finally, Rules R46 and R47 count 1W and 1R, respectively, for each incoming (Rule R46) and outgoing (Rule R47) transitions to/from object nodes labeled with the stereotype datastore . Indeed, a datastore is a stereotype for an object which stores objects persistently. The application of R46 and R47 to the diagram shown in Fig. 3 allows us to count 1R and 1W.

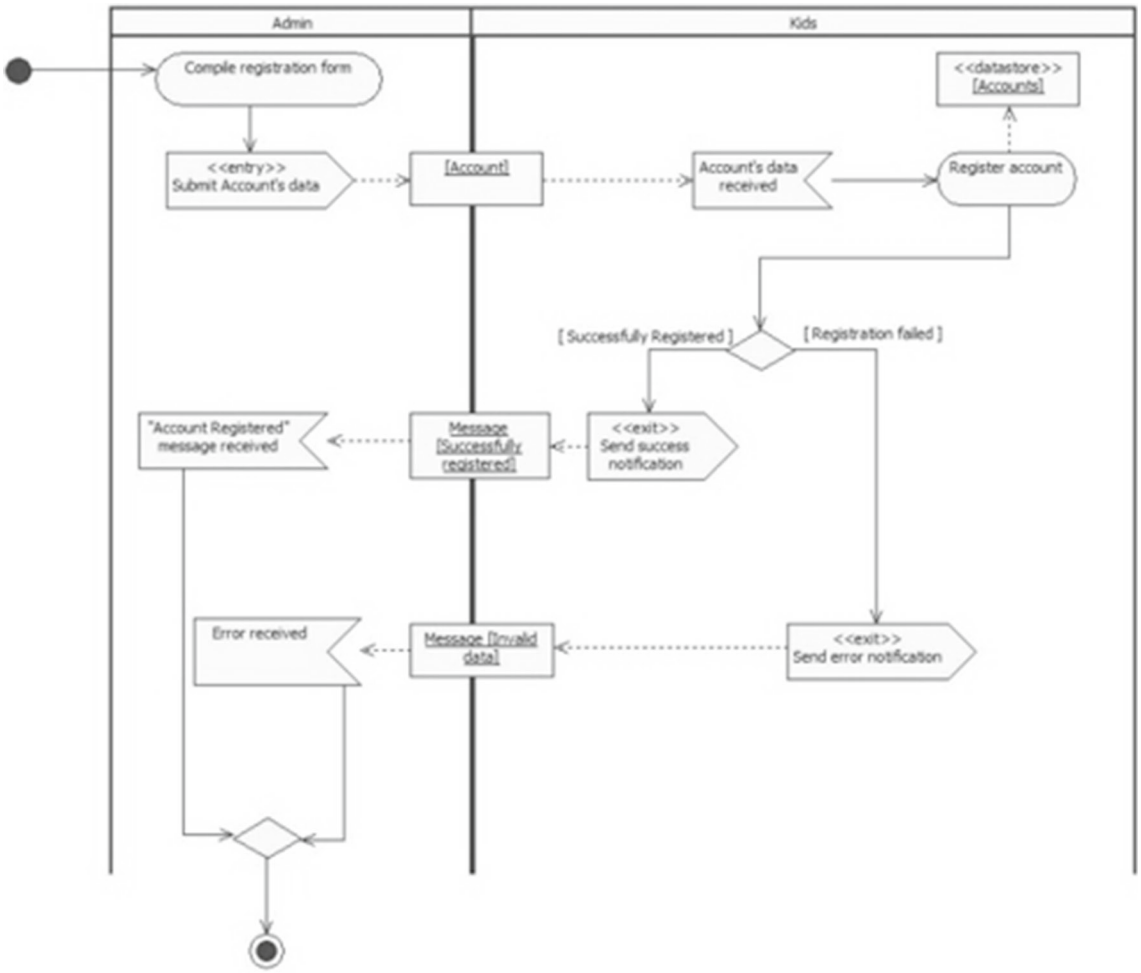


Fig. 2 Activity diagram for register account use case

5 The automation of the measurement procedure

While there is a large offering of software tools to support the development process, measurement of functional size had until recently been almost exclusively a manual process, therefore, time-consuming and subject to human errors. This section presents the approach to address the problem and the description of the tool developed for automating the proposed measurement procedure, named J-UML COSMIC.

5.1 Design and development of J-UML COSMIC

The design of the proposed tool is based on the consideration that there are different UML modeling tools, commercial and open source, which allow us to export their project files in different formats. So, importing UML models from heterogeneous tools and environments implies that J-UML COSMIC must have an adaptable conversion mechanism and a flexible data model. Some design goals guided the design and development of the tool, such as accuracy, repeatability, and reproducibility. Indeed, the tool is motivated by the need to perform fast and accurate measurements to alleviate the time consumption and errors produced by a manual measurement process. On the other hand, we wanted to be able to record manually performed analysis results (for instance, using the standard COSMIC method, or using the Early and Rapid methods [41]). We also took into account extensibility and modifiability aspects to allow the addition of new features, or the support for other tools/platforms. Moreover, it is possible to extend the scope of the tool to other measurement methods.

The architecture of J-UML COSMIC is based on the pattern layer. This architectural style provides the logical structure of the system into layers and the communications among them. The architecture of the tool, as shown in Fig. 5, consists of: (1) user interface (presentation layer), that handles user interactions with the system, (2) business logic layer (BL), where there are application services, (3) technical services, where persistence and other technical services are

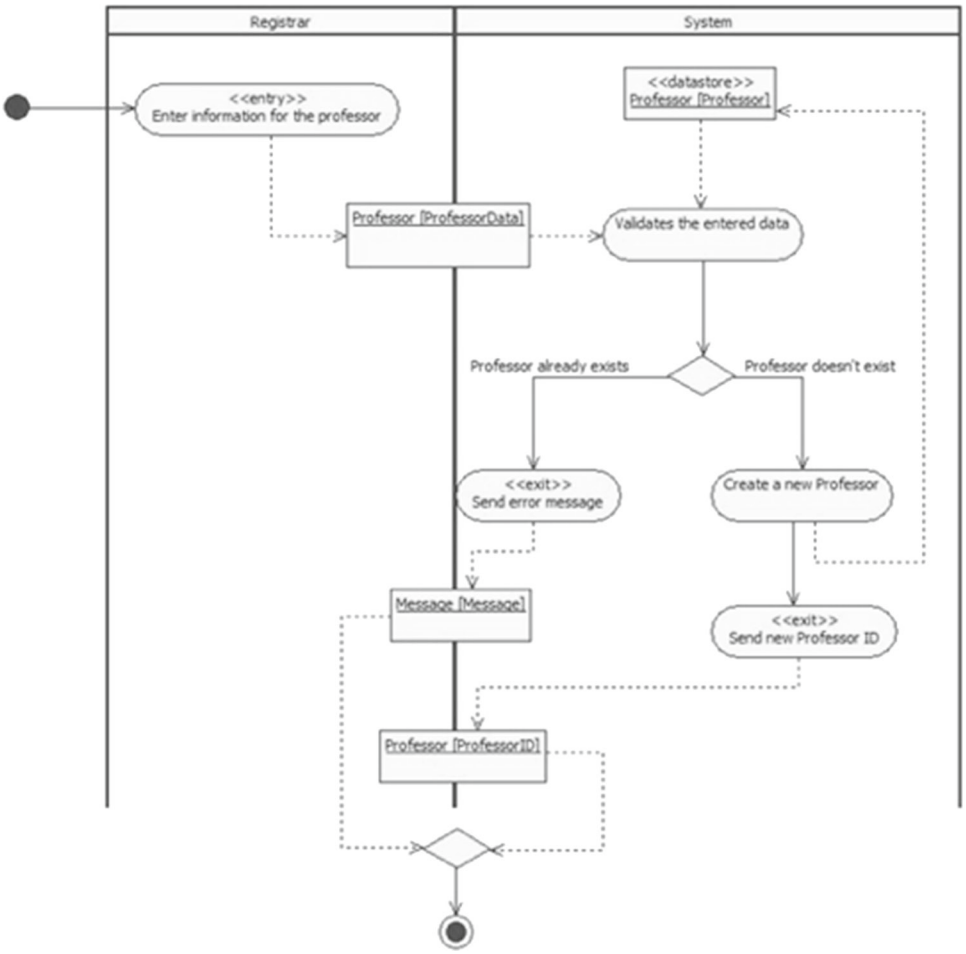


Fig. 3 Activity diagram for add a professor use case

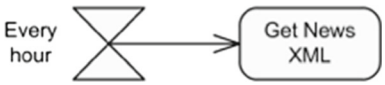


Fig. 4 Activity diagram for get XML news use case

managed, such as logging, and finally (4) the value object layer, where we can find UML, COSMIC, and Configuration value objects.

To develop the system we used several patterns, each for a specific design aspect, namely: MVC, facade, singleton, bridge, template method, and cache.

Finally, J-UML COSMIC was designed and developed in Java to support the following measurement methods: the standard COSMIC method, the COSMIC approximations average functional process, fixed size classification, equal size bands, average use case as defined in [41], Quick/Early as defined in [42], and activity and sequence diagrams measurement, i.e., the procedure presented in the previous sections. Moreover, it also has a function that allows verifying if a functional process has at least two data movements as requested by the COSMIC manual. These verification procedures further differentiate this work from other proposals.

5.2 Main features of J-UML COSMIC

J-UML COSMIC is characterized by a simple and intuitive user interface, typically used in desktop applications. We describe some features below.

The *project conversion* feature allows us to convert projects, that are developed using a specific platform or UML modeling tool, into the J-UML COSMIC conceptual model. As we can see in Fig. 6, the user can select the converter type (among those registered in the system) to perform the conversion operation. In the current version of the tool, we implemented only the StarUML project file converter.

The *configuration management* feature allows us to make changes to the tool configuration such as: registering new converters; registering new components to manage the business logic relative to the Strategy Phase and/or the Mapping Phase, making it easier to maintain the evolution of the tool when the measurement procedure changes; registering new measurement methods to compute the project functional size (see Fig. 7).

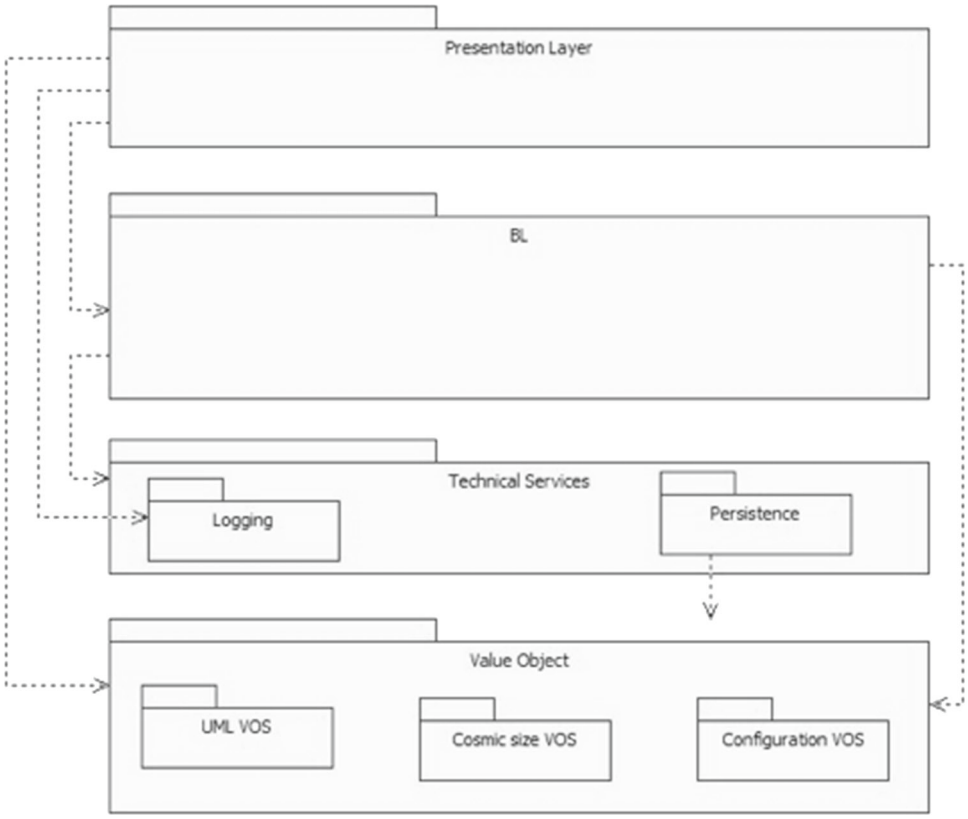


Fig. 5 J-UML COSMIC architecture

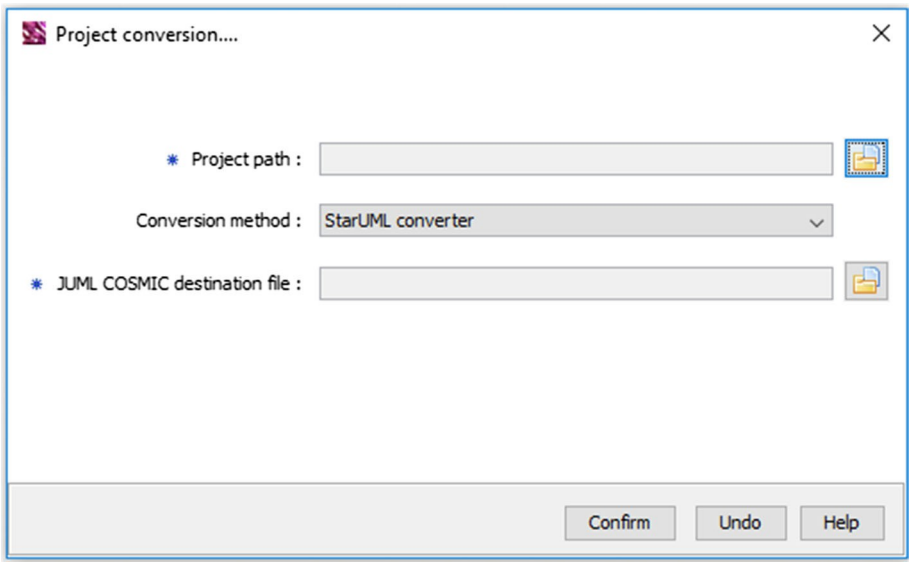


Fig. 6 J-UML COSMIC *conversion* feature

Open and Manage Project is the main functionality and is organized in three main areas:

- the *project explorer* which allows us to explore, in a hierarchical fashion, the UML models and elements and the project measurements;
- *the property inspector*, which displays the properties of each element in the project;
- *the workbench*, which represents the working area in which J-UML COSMIC displays as tabs:
  - the project summary (see Fig. 8),
  - the UML diagrams selected by the user (see Fig. 9),
  - the measurement data (see Fig. 10).

For supporting the registration of new project measurement results, we implemented a wizard which allow us to:

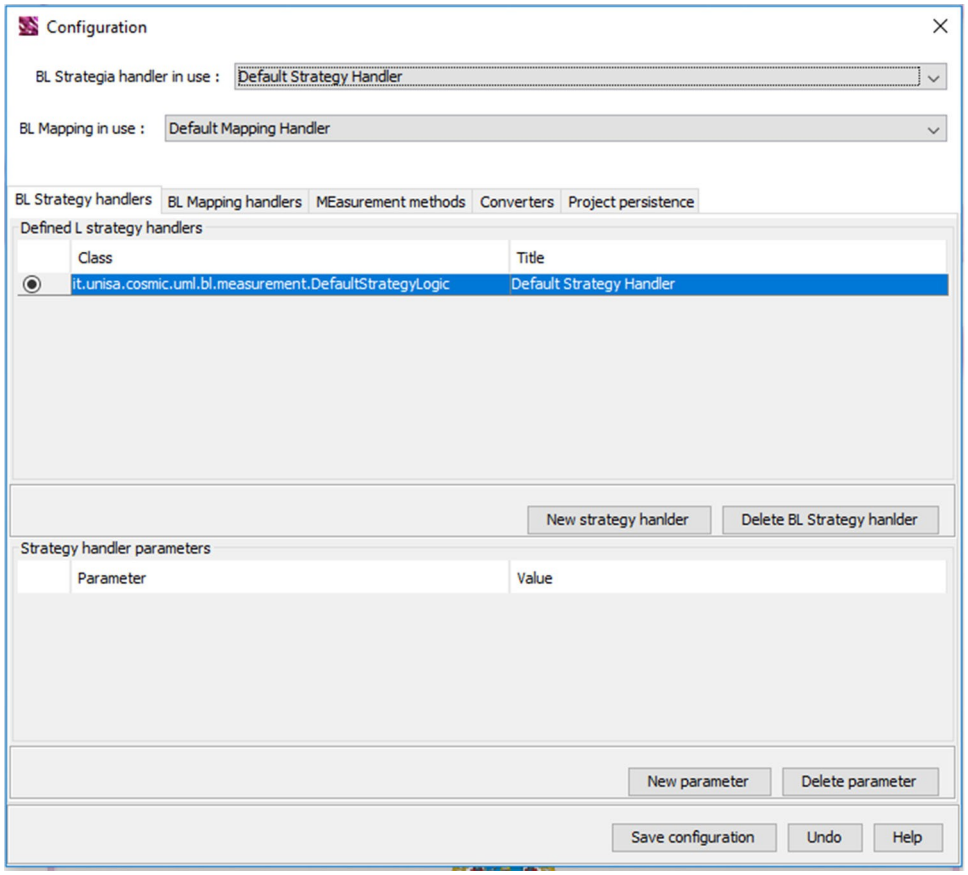


Fig. 7 J-UML COSMIC *configuration management* feature

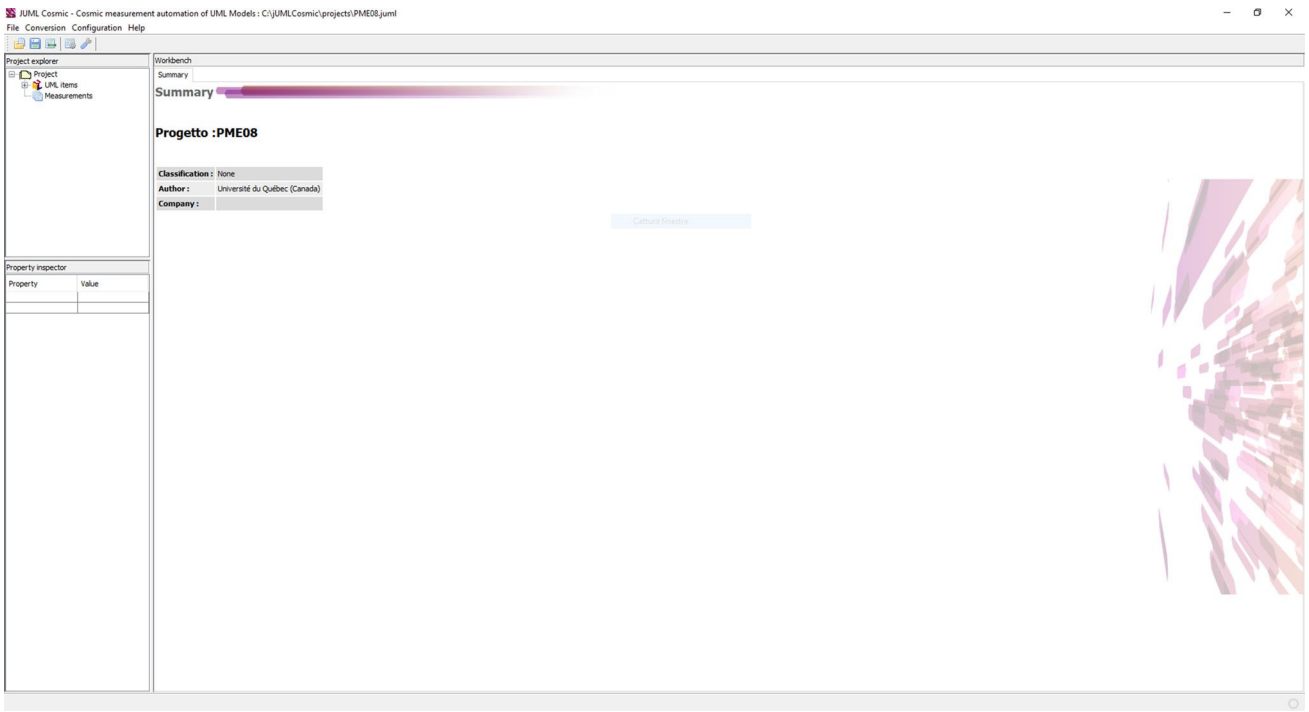


Fig. 8 J-UML COSMIC project summary tab

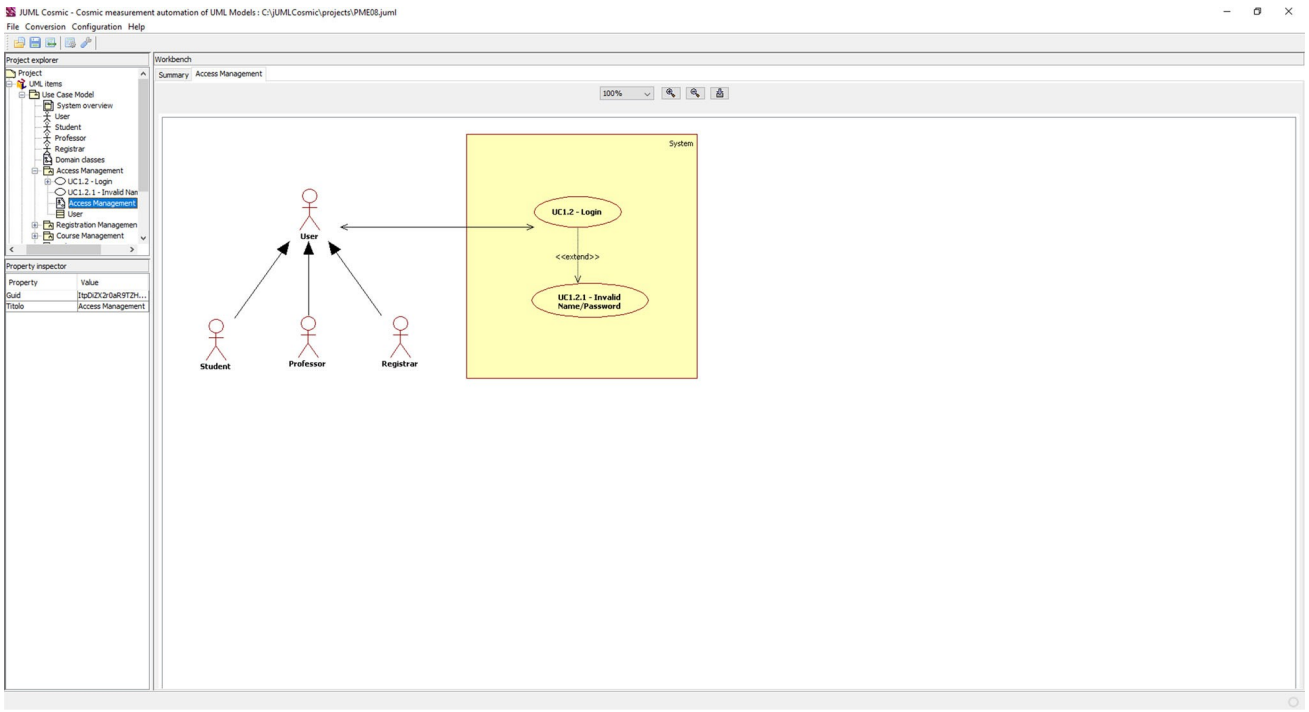


Fig. 9 J-UML COSMIC UML diagram tab

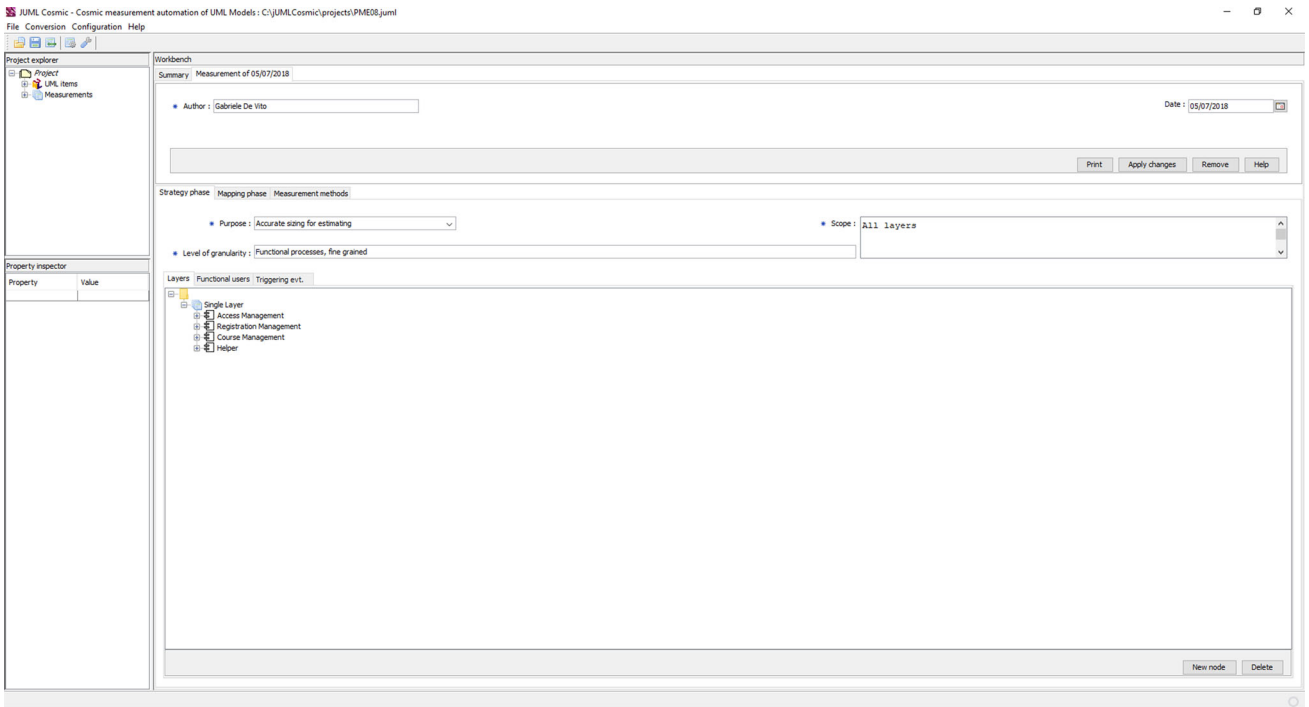


Fig. 10 J-UML COSMIC UML project measurement tab

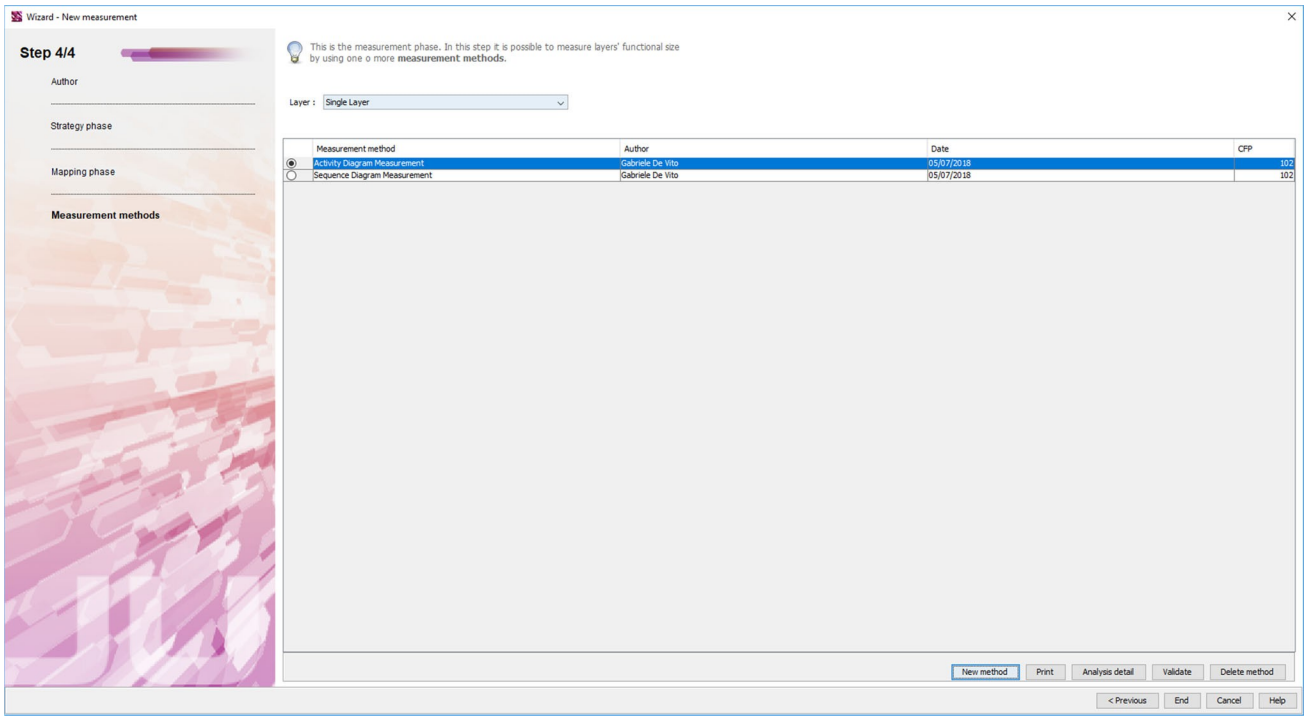


Fig. 11 Step 4 in J-UML COSMIC measurement wizard

1. register general project data;
2. handle the Strategy Phase;
3. handle the Mapping Phase;
4. register one or more measurement methods (for instance automatic measurement of activity and sequence diagrams, COSMIC standard, etc.).

In the first step, the user has to enter the author and the date of the measurement. In the second step, having specified the purpose and the scope of the measurement, the system automatically identifies the candidate layers, and for each layer the peer components, and for each peer component the UML packages to measure; and the functional users.

The J-UML COSMIC user can manually change the proposed information or can confirm it, entering the level of granularity of the FURs to measure. In the third step, the system automatically identifies the candidates for the triggering events; the functional processes; the objects of interest; and the data groups. Again, the user can manually change the proposed information. In the final step (see Fig. 11), the user must select the measurement method among those recorded in the configuration, to get the functional size of the project.

When the user confirms all the entered data, the system performs the measurement process, i.e., the necessary analyses to identify the data movements. This process is carried out automatically for supported methods (e.g., the COSMIC procedure proposed here), or interacting with the user in manual mode (e.g., to apply the standard COSMIC method).

For each project, the user can add multiple measurements, view and edit the detailed data, and print reports, containing general (see Fig. 12) or detailed (see Fig. 13) measurement information.

The general report displays the COSMIC concepts identified from the underlying UML artifacts by the automation tool. It contains three sections, one for each COSMIC phase: strategy, mapping, and measurement.

In the strategy section, it reports:

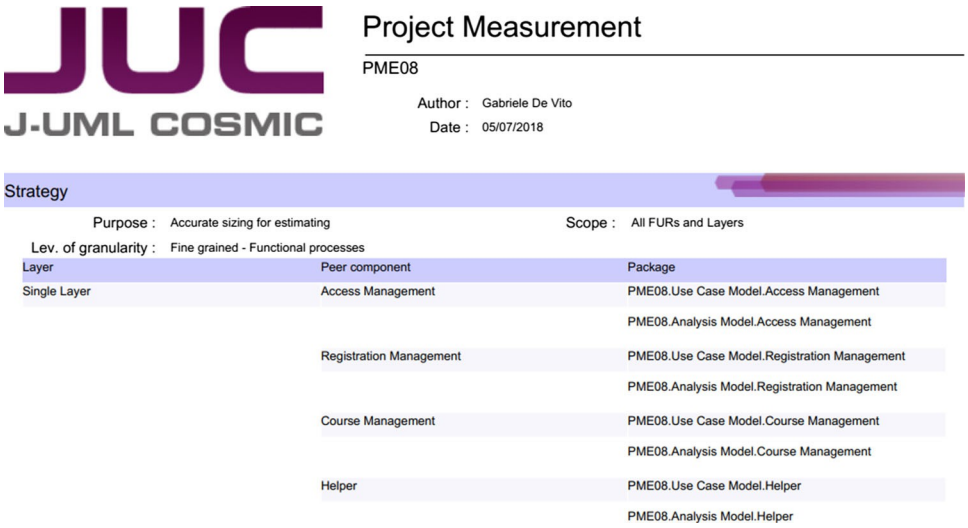
- the purpose, the scope, and the level of granularity of the measurement. For instance, as shown in Fig. 12, the purpose of the measurement is "Accurate sizing for estimating" and the scope is "All FURs and layers";
- the functional users;
- the features and the triggering events.

The mapping sections outlines:

- the objects of interest;
- the data groups and their attributes;
- the functional processes.

Finally, the measurement section gives us a summary of the measurement methods used in the project (i.e., sequence diagrams) and their CFPs.

As for the activity or sequence diagram reports, they outline the detailed identification of the data movements for



J-UML COSMIC

Project Measurement

PME08

Author : Gabriele De Vito

Date : 05/07/2018

Strategy

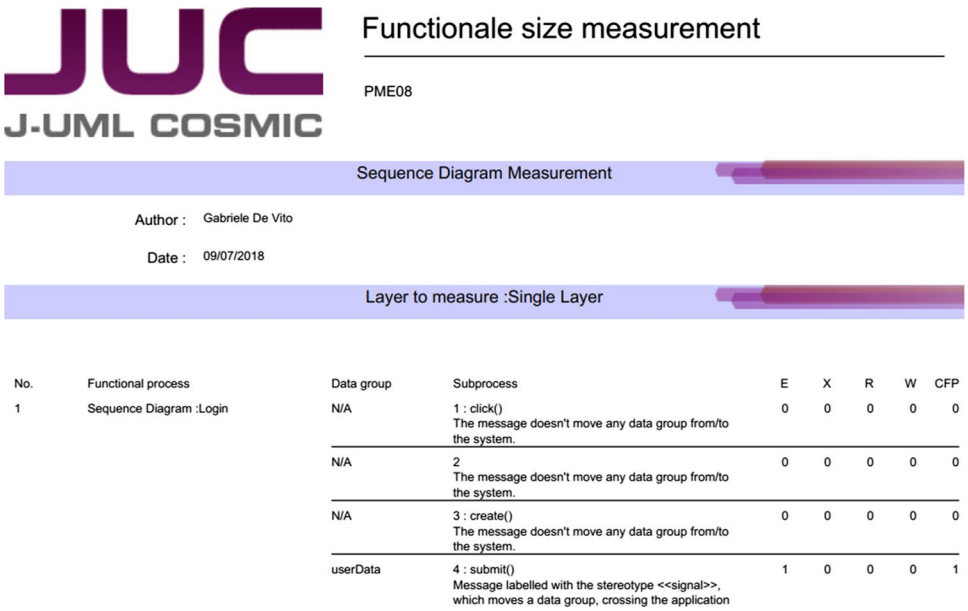
Purpose : Accurate sizing for estimating

Scope : All FURs and Layers

Lev. of granularity : Fine grained - Functional processes

Layer	Peer component	Package
Single Layer	Access Management	PME08.Use Case Model.Access Management
		PME08.Analysis Model.Access Management
	Registration Management	PME08.Use Case Model.Registration Management
		PME08.Analysis Model.Registration Management
	Course Management	PME08.Use Case Model.Course Management
		PME08.Analysis Model.Course Management
	Helper	PME08.Use Case Model.Helper
		PME08.Analysis Model.Helper

Fig. 12 J-UML COSMIC measurement general report



J-UML COSMIC

Functionale size measurement

PME08

Sequence Diagram Measurement

Author : Gabriele De Vito

Date : 09/07/2018

Layer to measure :Single Layer

No.	Functional process	Data group	Subprocess	E	X	R	W	CFP
1	Sequence Diagram :Login	N/A	1 : click() The message doesn't move any data group from/to the system.	0	0	0	0	0
		N/A	2 The message doesn't move any data group from/to the system.	0	0	0	0	0
		N/A	3 : create() The message doesn't move any data group from/to the system.	0	0	0	0	0
		userData	4 : submit() Message labelled with the stereotype <<signal>>, which moves a data group, crossing the application	1	0	0	0	1

Fig. 13 J-UML COSMIC sequence and activity diagrams measurement report

each functional process. For instance, as shown in Fig. 13, the sub-process no. 4 moves the data group "User data" into the software, crossing the boundary, because it is a message labeled with the stereotype <<signal>> sent by the functional user to the system and, therefore, we can identify 1E.

6 Accuracy of the measurement procedure and tool

We performed two case studies to show the application of the proposed measurement procedure and tool and to compare the automatic measurement results with the manual measurement carried out by experts who exploited the functional requirement specifications. In particular, we first applied our proposal to the "Web Advice Module case study" produced by members of the COSMIC working group of NESMA since it is a well-known example showing how to apply the COSMIC method to Web applications (see Sect. 6.1). Then, we have considered the "Course Registration case study" which was initially written by Adel Khelifi and then reviewed by members of the COSMIC group like Alain Abran, Charles Symons, and Jean-Marc Desharnais (see 6.2).

Verification of the measurement accuracy was done in different phases aligned with the verification protocol described in [43]:

- in the first phase we compared the COSMIC concepts manually identified with those identified using our measurement procedure and tool;
- in the second phase we compared the total functional sizes in CFP manually obtained with those produced by the measurement procedure and tool;

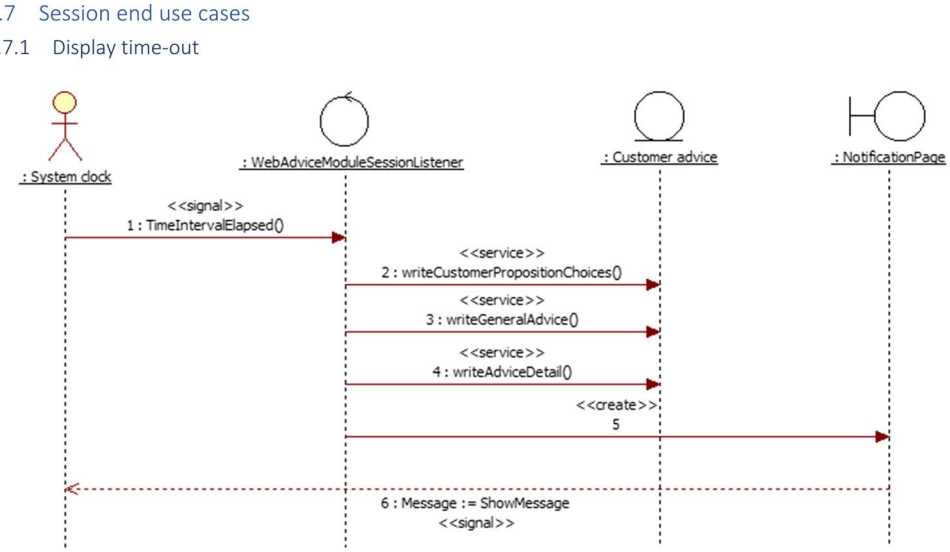


Fig. 14 The sequence diagram display time-out of web advise module

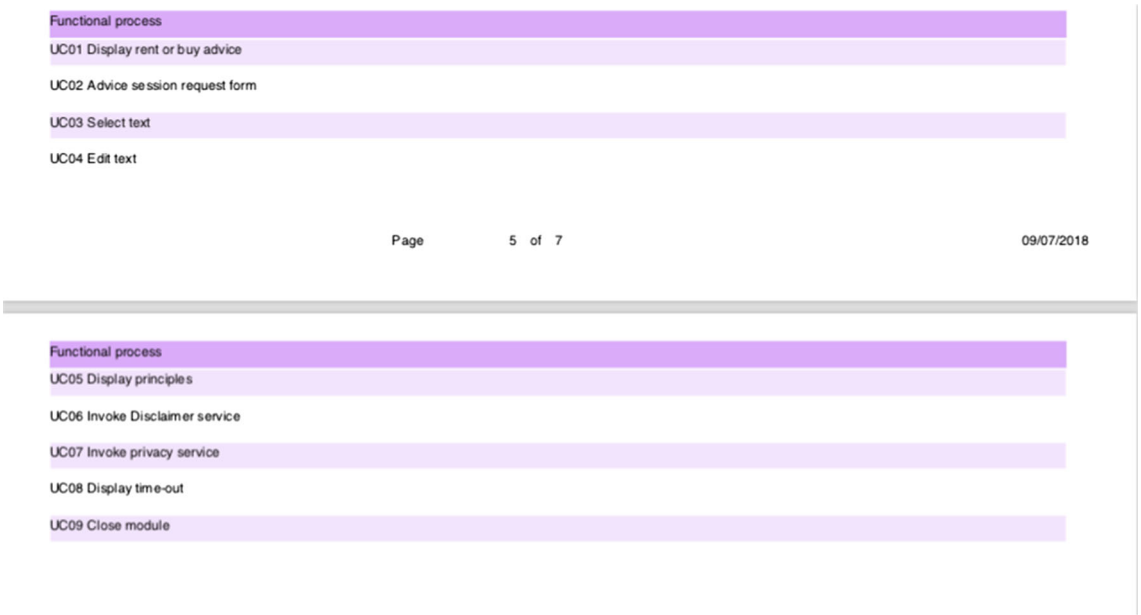


Fig. 15 An excerpt of the general report obtained for web advise module

– in the third phase, for each functional process, we verified sub-processes, data groups, and then data movements manually identified with those automatically identified by our measurement procedure and tool.

The two studies are described below.

6.1 First study: web advice module

The Web Advice Module case study gives insights and detailed examples on how to size a Web-based business application, especially if Web services are used in the business application domain [44]. This is a domain where first generation functional size measurement methods struggled to size web services as separate components within a service-oriented architecture. Even if the analyzed application is small (just 42 CFP), it is pretty complicated and it helps the measurement professionals in ensuring coherent interpretation of the COSMIC principles.

All the detailed information about the study can be found on the Web appendix,[1] where we made available:

– the Web Advise Module documentation and manual measurement from the COSMIC community;
– the UML diagrams exploited for the measurement;

[1] See https://docenti.unisa.it/004724/risorse?categoria=348&risorsa=2376.

Fig. 16 An excerpt of the detailed measurement information for web advise module

Layer to measure :Single Layer

No.	Functional process	Data group	Subprocess	E	X	R	W	CFP
8	Sequence Diagram :UC08 Display time-out	null	1 : TimeIntervalElapsed() Message labelled with the stereotype <<signal>>, which moves a data group, crossing the application boundary. Therefore, applying the R54 rule, we can identify an Entry.	1	0	0	0	1
		Customer advice	2 : writeCustomerPropositionChoices() Message labelled with the stereotype <<service>>, which writes data in the persistent storage. Applying the R55 rule, we can identify a Write.	0	0	0	1	1
		Customer advice	3 : writeGeneralAdvice() Message labelled with the stereotype <<service>>, which writes data in the persistent storage. Applying the R55 rule, we can identify a Write.	0	0	0	1	1
		Customer advice	4 : writeAdviceDetail() Message labelled with the stereotype <<service>>, which writes data in the persistent storage. Applying the R55 rule, we can identify a Write.	0	0	0	1	1
		N/A	5 The message doesn't move any data group from/to the system.	0	0	0	0	0
		Message	6 : Message := ShowMessage Message labelled with the stereotype <<signal>>, which moves a data group from the system to the functional user. Therefore, applying the R54 rule, we can identifyan eXit.	0	1	0	0	1
9	Sequence Diagram :UC09 Close module	null	1 : Click() Message labelled with the stereotype <<signal>>,	1	0	0	0	1

Page 9 of 10 09/07/2018

Layer to measure :Single Layer

No.	Functional process	Data group	Subprocess	E	X	R	W	CFP
			which moves a data group, crossing the application boundary. Therefore, applying the R54 rule, we can identify an Entry.					
		N/A	2 : closeSession() The message doesn't move any data group from/to the system.	0	0	0	0	0
		Customer advice	3 : writeCustomerPropositionChoices() Message labelled with the stereotype <<service>>, which writes data in the persistent storage. Applying the R55 rule, we can identify a Write.	0	0	0	1	1
		Customer advice	4 : writeGeneralAdvice() Message labelled with the stereotype <<service>>, which writes data in the persistent storage. Applying the R55 rule, we can identify a Write.	0	0	0	1	1
		Customer advice	5 : writeAdviceDetail() Message labelled with the stereotype <<service>>, which writes data in the persistent storage. Applying the R55 rule, we can identify a Write.	0	0	0	1	1
				7	16	7	10	42

Page 10 of 10 09/07/2018

– the general report obtained with J-UML COSMIC;
– the report with the detailed measurement information obtained with J-UML COSMIC.

Figure 14 shows one of the UML diagrams considered in the study: the sequence diagram *Display Time-out*. The documentation provided by the COSMIC community states that: *"the Web Advice Module uses existing functionality of the Web server, which uses the system clock of the operating system. If the inactivity time exceeds the threshold the system clock notifies the Web server, which triggers the Web Advice Module to show the inactivity message and delete the customers data"*.

Figure 15 shows an excerpt of the general report produced by J-UML COSMIC showing the identified functional processes, while the last 2 pages of the report presenting the detailed measurement information are provided in Fig. 16. We observe that J-UML COSMIC is able to identify the same data movements manually determined by the experts of the COSMIC community for this functional process: 1E, 3W, and 1X. This happens for all the functional processes of Web Advise Module, obtaining as final number of CFPs the value

4.3.4 View report card use cases

4.3.4.1 View report card

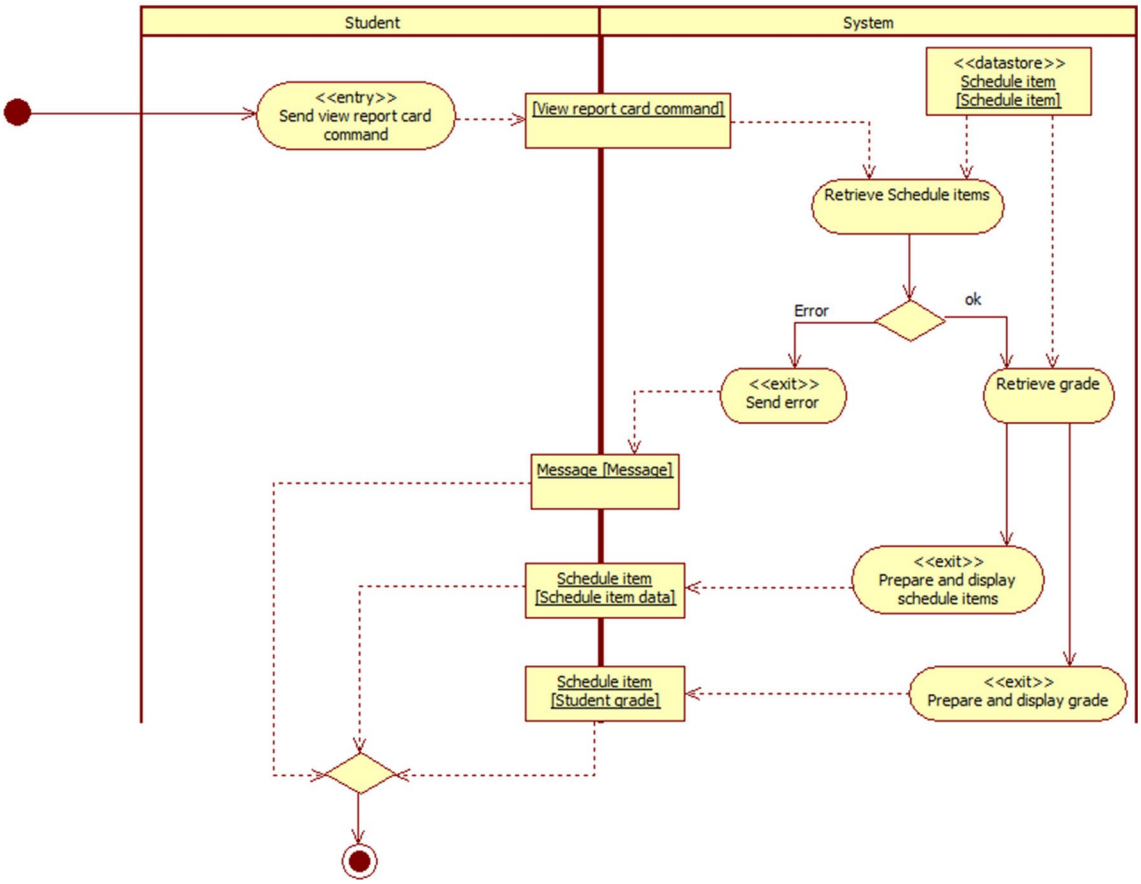


Fig. 17 The activity diagram view report card of course registration system

determined by the experts (see page 10 of the detailed report shown in Fig. 16).[2] So, J-UML COSMIC calculated 7E, 16X, 7R, and 10W as done by COSMIC experts.

6.2 Second study: course registration system

In the second study, the COSMIC method is applied to the Course Registration Software System (C-Reg), which is a Web application documented in the Rational Unified Process document (RUP version 2003.06.00.65). The application provided by the members of the COSMIC group offers a complete analysis and presents the results of applying the COSMIC method to the business application domain [45]. Although C-Reg is a compact application (102 CFP), it covers different aspects of the COSMIC method, from the determination of various parameters of the Measurement Strategy Phase to the identification of the data movements involved in communications between software systems. Moreover, it deals with real-world software requirements issues.

Figure 17 shows one of the UML diagrams considered in the study: the activity diagram *View Report Card*. The documentation provided by the COSMIC community states that: *"The system retrieves the grade information for each of the courses the Student completed during the previous semester. The system prepares, formats, and displays the grade information. When Student is finished viewing the grade information the Student selects 'close'"*.

Figure 18 shows the last 3 pages of the report presenting the detailed measurement information. We observe that J-UML COSMIC is able to identify the same data movements determined by the experts of the COSMIC community, i.e., 1E, 2R, and 3X, for View Report Card (see pages 25 and 26 in Fig. 18). Similarly, for the remaining diagrams J-UML COSMIC obtained the same data movements identified by experts, thus showing 32E, 44X, 13R, and 13W (see the end of the report in Fig. 18).

For this second case study, we also show how the use of the UML artifacts at different levels of granularity can impact the

[2] All the details about the data movements calculated for each functional process by J-UML COSMIC can be found on the Web appendix.

Fig. 18 An excerpt of the detailed measurement information for course registration system

Layer to measure :Single Layer

No.	Functional process	Data group	Subprocess	E	X	R	W	CFP
		Student grade	Display grades ActionState node labelled with the stereotype <<Exit>> which moves a data group from the functional user to the system, crossing the application boundary. Therefore, applying the R45 rule, we can identify an Exit.	0	1	0	0	1
		Schedule item data	Enter or change grades Action State node labelled with the stereotype <<Entry>>, which muoves a data group from the functional user to the system, crossing the application boundary. Applying the R43 rule, we can identify an Entry.	1	0	0	0	1
		N/A	Record grades The activity Record grades UMLActionState doesn't move data groups from/to the system.	0	0	0	0	0
		Message	Send error ActionState node labelled with the stereotype <<Exit>> which moves a data group from the functional user to the system, crossing the application boundary. Therefore, applying the R45 rule, we can identify an Exit.	0	1	0	0	1
17	Activity Diagram : View report card	View report card command	Send view report card command Action State node labelled with the stereotype <<Entry>>, which muoves a data group from the functional user to the system, crossing the application boundary. Applying the R43 rule, we can identify an Entry.	1	0	0	0	1
		N/A	Retrieve Schedule items The activity Retrieve Schedule items UMLActionState doesn't move data groups from/to the system.	0	0	0	0	0

Layer to measure :Single Layer

No.	Functional process	Data group	Subprocess	E	X	R	W	CFP
		Schedule item	Schedule item Outgoing transition from an object node labelled with the stereotype <<datastore>>, which moves a data group from the persistent storage to the system. Applying the R46 rule, we can identify a Read.	0	0	1	0	1
		Schedule item	Schedule item Outgoing transition from an object node labelled with the stereotype <<datastore>>, which moves a data group from the persistent storage to the system. Applying the R46 rule, we can identify a Read.	0	0	1	0	1
		Message	Send error ActionState node labelled with the stereotype <<Exit>> which moves a data group from the functional user to the system, crossing the application boundary. Therefore, applying the R45 rule, we can identify an Exit.	0	1	0	0	1
		N/A	Retrieve grade The activity Retrieve grade UMLActionState doesn't move data groups from/to the system.	0	0	0	0	0
		Schedule item data	Prepare and display schedule items ActionState node labelled with the stereotype <<Exit>> which moves a data group from the functional user to the system, crossing the application boundary. Therefore, applying the R45 rule, we can identify an Exit.	0	1	0	0	1
		Student grade	Prepare and display grade ActionState node labelled with the stereotype <<Exit>> which moves a data group from the functional user to the system, crossing the application boundary. Therefore, applying the R45 rule, we can	0	1	0	0	1

Layer to measure :Single Layer

No.	Functional process	Data group	Subprocess	E	X	R	W	CFP
			identify an Exit.					
				32	44	13	13	102

Table 6 Results of the measurement considering different granularity levels

Doc. 1		Doc. 2		Doc. 3	
Functional process	*CFP*	Functional process	*CFP*	Functional process	*CFP*
Add a professor	5	Logon	3	Logon	3
Modify a professor	6	Add a professor	5	Add a professor	5
Delete a professor	6	Modify a professor	6	Enquire on a professor	4
Select Courses to Teach	9	Delete a professor	6	Modify a professor	3
Add a student	4	Select Courses to Teach	9	Delete a professor	3
Modify a student	6	Add a student	4	Select Courses to Teach	9
Delete a Student	6	Modify a Student	6	Add a student	4
Create a schedule	13	Delete a Student	6	Enquire on a student	4
Modify a schedule	15	Create a schedule	13	Modify a student	3
Delete a schedule	7	Modify a schedule	15	Delete a Student	3
Close registration	9	Delete a schedule	7	Create a schedule	13
Submit grades	12	Close registration	9	Enquire on a schedule	4
		Submit grades	12	Modify a schedule	13
		View Report Card	6	Delete a schedule	4
				Close registration	9
				Submit grades	12
				View Report Card	6
Total	98	Total	107	Total	102

result of the measurement. In particular, we show that the tool is able to leverage the input of more detailed UML artifacts to identify the COSMIC elements and increase the precision of its measurement.

In particular, in Table 6 we report the results we achieved when applying J-UML COSMIC to the following documentation:

- (Doc. 1) the requirements analysis is not complete. Indeed, we consider only use cases and sequence diagrams and some functionality is not included (i.e., packages of "Access management" and "View report card"), so some domain classes are not present. Furthermore, neither analysis classes nor activity diagrams are present.
- (Doc. 2) all the functionality has been described and analyzed, the domain classes are complete, the analysis classes have been introduced, however the requirements analysis needs further refinement. Indeed, the functional processes "Modify" and "Delete" for Professor, Student, and Schedule elements have to be refined and activity diagrams have to be specified. For this reason, the documentation does not contain the "Enquire" functional process (see the Course Registration System documentation and manual measurement from the COSMIC community included in the Web appendix for details, where the experts analyze and discuss this case).
- (Doc. 3) the complete documentation used to obtain the final measurement reported in Fig. 18 and discussed above.

All the documentation for both case studies can be found in the Web appendix.[3]

7 Conclusions and future work

We have proposed a flexible COSMIC measurement procedure that is applicable to different UML artifacts which can have different levels of granularity (such as use case models, package diagrams, component diagrams, class diagrams, activity diagrams, sequence diagrams). This allows measures to obtain more accurate measurements when new models have been defined or the existing ones have been enriched with more detail. We have also presented J-UML COSMIC, a tool that supports the proposed measurement procedure and simplifies project monitoring and control, because it (almost) automatically allows us to obtain increasingly accurate measurements. This allows project managers to always have an updated project functional size and, according to it, to determine what corrective actions/improvements apply to the project management process. As shown in the paper,

[3] See https://docenti.unisa.it/004724/risorse?categoria=348&risorsa=2376.

the proposal presents some novel aspects with respect to the existing state-of-the-art.

The proposal has, like other similar approaches in the literature, some limitations. As for the proposed measurement procedure, a limitation derives from the need to stereotype the data movements in the activity diagrams and in the sequence diagrams. Even if this annotation practice is allowed in UML and adds new stereotypes to the activity diagrams only, it introduces an additional task for the analyst, which will take more time in the software model design. The further cost for these practices is minimal, because the necessary information is often present in UML models, and therefore, it only needs to be extracted and annotated in the diagrams. As another limitation, companies could adopt a subset of the UML models which does not include those needed by the proposed procedure. Regarding the tool J-UML COSMIC, we can identify the following limitations: (1) the measurement scope must be manually identified; (2) we developed a single converter and we can convert only StarUML projects. However, this limitation is only apparent since StarUML can import Rational Rose and XMI projects; (3) no large-scale validation has been carried out yet.

As future work, other useful features could be implemented. The measurement results, the local calibration data, and the classifications of the managed projects may be registered in a central repository to be used as a data source to perform the following analyses: (1) understanding the relationships between the different measurement methods and (2) determining the measurement accuracy at different stages of the software life cycle. An additional feature that we consider important is "versioning" both at project level (for instance, subversioning SVN) and at UML artifacts level as they are "refined". Moreover, the modular tool architecture allows us to add new measurement methods (for instance, a new estimation method, a variant of an existing one, etc.), new UML projects "Converters," and new "business logic components" to manage the Strategy Phase and/or the Mapping phase, which can be customized in terms of user interfaces and behaviors, and stored using the "Configuration manager" to be available at next tool reboot. Finally, we could enhance our procedure to investigate possible relations between abstraction levels of the measured artifacts and functional size.

References

1. Sommerville, I.: Software Engineering, 9th edn. Addison-Wesley, Harlow (2010)
2. Cheung, Y., Willis, R., Milne, B.: Software benchmarks using function point analysis. Benchmarking Int. J. **6**(3), 269–276 (1999)
3. Fetcke, T.: The Warehouse Software Portfolio. A Case Study in Functional Size Measurement, Technical Report No. 199920, Departement d'informatique, Université du Quebec, Montreal, Canada (1999)
4. Albrecht, A.: Measuring application development productivity. In: Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, pp. 8392 (1979)
5. Gencel, C., Demirors, O.: Functional size measurement revisited. ACM Trans. Softw. Eng. Methodol. **17**(3), 15:1–15:36 (2008)
6. ISO: ISO/IEC 14143-1:2007: Information Technology Software measurement Functional Size Measurement (2007)
7. Abran, A., Desharnais, J., Lesterhuis, A., Londeix, B., Meli, M., Morris, P., Oligny, S., ONeil, M., Rollo, T., Rule, G., Santillo, L., Symons, C., Toivonen, H.: The COSMIC Functional Size Measurement Method Measurement Manual, version 4.0.1 (2015)
8. Bevo, V., Levesque, G., Abran, A.: Application de la methode FFP partir d'une specification selon la notation UML: compte rendues premiers essais dapplication et questions. In: International Workshop on Software Measurement, Lac Superieur, Canada, Sep. (1999)
9. Jenner, M.S.: COSMIC-FFP and UML: estimation of the size of a system specified in UML problems of granularity. In: The 4th European Conference on Software Measurement and ICT Control, Heildelberg, pp. 173-184 (2001)
10. Azzouz, S., Abran, A.: A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO 19761: COSMIC-FFP. In: Software Measurement European Forum (SMEF 2004), Rome, Italy (2004)
11. Luckson, V., Lèvesque, G.: Une méthode efficace pour l'extraction des instances de concepts dans une spécification UML aux fins de mesure de la taille fonctionnelle de logiciels. In: The Seventeenth International Conference Software, System Engineering, Their Applications (ICSSEA 2004), Paris
12. Lind, K., Heldal, R.: A model-based and automated approach to size estimation of embedded software components. In: ACM/IEEE The 14th International Conference on Model Driven Engineering Languages and Systems. Wellington (2011)
13. Lind, K., Heldal, R.: Estimation of real-time software code size using COSMIC FSM. In: International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC 2009), Tokyo, pp. 244–248
14. Habela, P., Glowacki, E., Serafinski, T., Subieta, K.: Adapting use case model for COSMIC-FFP based measurement. In: 15th International Workshop on Software Measurement (IWSM 2005), Montréal, pp. 195-207 (2005)
15. Levesque, G., Bevo, V., Cao, D.T.: Estimating software size with UML models. In: Proc. of the 2008 C3S2E Conference, Montréal, pp. 81–87 (2008)
16. Sellami, A., Ben-Abdallah, H.: Functional size of use case diagrams: a fine-grain measurement. In: Fourth International Conference on Software Engineering Advances (ICSEA 2009), pp. 282–288 (2009)
17. Van den Berg, K.G., Dekkers, T., Oudshoorn, R.: Functional size measurement applied to UML-based user requirements. In: Proc. of the 2nd Software Measurement European Forum (SMEF 2005), Rome, Italy, pp. 69–80 (2005)
18. Lavazza, L., Del Bianco, V.: A Case Study in COSMIC Functional Size Measurement: The Rice Cooker Revisited, IWSM/Mensura, pp. 101–121 (2009)
19. Costagliola, G., Di Martino, S., Ferrucci, S., Gravino, C., Tortora, G., Vitiello, G.: A COSMIC-FFP approach to predict web application development effort. J. Web Eng. **5**(2), 93–120 (2006)
20. Del Bianco, V., Lavazza, L., Liu, G., Morasca, S., Abualkishik, A.Z.: Model-based early and rapid estimation of COSMIC functional size—an experimental evaluation. Inf. Softw. Technol. **56**(10), 1253–1267 (2014)
21. Kruchten, P.: The Rational Unified Process, An Introduction, p. 298. Addison Wesley, Reading (2000)
22. OMG: Official Web Site Of The Object Management Group, UML section, OMG. www.uml.org

23. Diab, H., Koukane, F., Frappier, M., St-Denis, R.: ucROSE: automated measurement of COSMIC-FFP for rational rose real time. Inf. Softw. Technol. **43**, 151–166 (2005)
24. COSMIC Group: The COSMIC Functional Size Measurement Method, Version 3.0.1, Measurement Manual. www.cosmic-sizing.org
25. Jenner, M.S.: Automation of counting of functional size using COSMIC-FFP in UML. In: International Workshop Software Measurement (IWSM 2002), pp. 43–51 (2002)
26. IBM: Official Web Site Of Rational Rose, IBM. www-03.ibm.com
27. Bevo, V.: Analyse et formalisation ontologique des procedures de measures associes aux method de mesure de la taille fonctionnelles des logiciels: de nouvelles perspectives pour la mesure, doctoral thesis, UQAM, Montreal (2005)
28. Condori-Fernandez, N., Abrahao, S., Pastor, O.: On the estimation of the functional size of software from requirements specifications. J. Comput. Sci. Technol. **22**, 358–370 (2007)
29. COSMIC Group: Rice Cooker. Cosmic Group Case Study. École de technologie Supérieure, Université du Québec á Montréal, UQAM, Mo
30. Ungan, E., Demirörs, O.: A Functional Software Measurement Approach to Bridge the Gap Between Problem and Solution Domains, Mensura/IWSM 2015: Software Measurement, pp. 176–191
31. Rollo, T.: Sizing E-commerce. In: Proc. of the ACOSM 2000—Australian Conference on Software Measurement, Sydney (2000)
32. Mendes, E., Counsell, S., Mosley, N.: Comparison of web size measures for predicting web design and authoring effort. IEE Proc. Softw. **149**(3), 86–92 (2002)
33. Conallen, J.: Building Web Applications with UML, Addison Wesley Object Technology Series (1999)
34. Barkallah, S., Gherbi, A., Abran, A.: COSMIC functional size measurement using UML models. In: Kim, T. et al. (eds.) Software Engineering, Business Continuity, and Education. ASEA 2011. Communications in Computer and Information Science, vol. 257. Springer, Berlin, Heidelberg
35. Marín, B., Oscar, P., Giachetti, G.: Automating the measurement of functional size of conceptual models in an MDA environment. In: 9th International Conference on Product-Focused Software Process Improvement (PROFES 2008), pp. 215–229
36. Abrahao, S., De Marco, L., Ferrucci, F., Gomez, J., Gravino, C., Sarro, F.: Definition and evaluation of a COSMIC measurement procedure for sizing web applications in a model-driven development environment. Inf. Softw. Technol. (2018). https://doi.org/10.1016/j.infsof.2018.07.012
37. Abrahao, S., De Marco, L., Ferrucci, F., Gravino, C., Sarro, F.: A COSMIC measurement procedure for sizing web applications developed using the OO–H. In: Proc. of International Workshop on Advances in Functional Size Measurement and Effort Estimation, art. 2 (2010)
38. Soubra, H., Abran, A., Stern, S., Ramdan-Cherif, A.: A refined functional size measurement procedure for real-time embedded software requirements expressed using the simulink model. In: 22nd IWSM-MENSURA (2012)
39. Soubra, H., Abran, A., Stern, S., Ramdan-Cherif, A.: Design of a functional size measurement procedure for real-time embedded software requirements expressed using the simulink model. In: 21st IWSM-MENSURA (2011)
40. Mathworks, Official Web Site Of Mathworks, Simulink Section, Mathworks. www.mathworks.com/
41. COSMIC Group: The COSMIC Functional Size Measurement Method, Version 3.0, Advanced and Related Topics (2007). www.cosmic-sizing.org
42. De Vito, G., Ferrucci, F.: Approximate COSMIC Size: The Quick/Early Method. In: Proc. of EUROMICRO-SEAA, pp. 69–76 (2014)
43. Soubra, H., Abran, A., Ramdane-Cherif, A.: Verifying the accuracy of automation tools for the measurement of software with COSMIC—ISO 19761 including an AUTOSAR-based example and a case study. In: Proc. of Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement, pp. 23–31 (2014)
44. Vogelezang, F., Onvlee, J., van der Vliet, E., van Heeringen, H., de Wilde, F., Bellen, P.: Web advice module case study: using COSMIC in a service oriented architecture. In: Proc. of Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement, pp. 112–114 (2012)
45. Khelifi, A., Abran, A., Symons, C., Desharnais, J.-M., Lesterhuis, A.: C-Registration System—Proposed Measurement Etalon. www.cosmic-sizing.org
46. Symons, C.: Advancing Functional Size Measurement Which Size Should We Measure (2007)
47. Vazquez, C., Siqueira Simoes, G.: Functional User Requirements: Level of Granularity and FPA, IFPUG, MetricViews **10**(2) (2016)
48. Timp, A.: FPA applied to UML/Use Cases, ISMA-3 (2008)
49. Timp, A.: uTip—Early Function Point Analysis and Consistent Cost Estimating, IFPUG uTip #03, v. 1.0, 2015/7/1. http://www.ifpug.org/uTips/uTip003EarlyFPAandConsistentCostEstimating.pdf
50. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, Boston (2000)

Gabriele De Vito received the Master of Science degree in Computer Science from the University of Salerno, Italy and he is currently the CTO of Innovaway, an international company that provides ICT services. His research interests include functional size measurement, empirical software engineering, and artificial intelligence.

Filomena Ferrucci is professor of software engineering and software project management at University of Salerno, Italy. Her main research interests include software metrics, effort estimation, empirical software engineering, search-based software engineering, and human–computer interaction.

Carmine Gravino is associate professor at the University of Salerno, Italy. His research interests include software metrics and techniques to estimate development effort, software-development environments, design pattern recovery from object-oriented code, empirical software engineering.