Automating the Measurement of Functional Size of Conceptual Models in an MDA Environment[*]

Beatriz Marín[1], Oscar Pastor[1] and Giovanni Giachetti[1]

[1] Department of Information Systems and Computation,
Technical University of Valencia,
Camino de Vera s/n,
46022 Valencia, Spain
{bmarin, opastor, ggiachetti }@dsic.upv.es

Abstract. The manual measurement of functional size is generally very time-consuming and has many precision errors. For this reason, it is necessary to automate the measurement process to obtain a solution that can be applied in a MDA industrial development. The OO-Method COSMIC Function Points (OOmCFP) is a measurement procedure that has been designed to measure the functional size of object-oriented applications generated from their conceptual models by means of model transformations. This work presents the definition of the mechanisms that are necessary to automate the OOmCFP procedure. This work also presents the OOmCFP tool that implements the OOmCFP procedure. Since this tool measures the functional size of industrial applications generated in MDA environments from their conceptual models, it is not necessary to perform the measurement task on the final code. The OOmCFP tool incorporates the benefits that the COSMIC measurement method provides. These benefits are demonstrated through a comparative analysis.

Keywords: Conceptual modeling, Object orientation, Functional size measurement, COSMIC, MDA, Tool.

1 Introduction

The Model-Driven Architecture (MDA) approach [17] separates application and business logic from the platform technology, allowing code generation by means of model transformations. In MDA contexts, conceptual models are used as input to the process of code generation. Thus, the conceptual models must have enough semantic formalization in order to specify all the functionality of the final application and also to avoid different interpretations for the same model.

The OO-Method approach [18] [20] is an object-oriented method that provides the required semantic formalization to define complete and unambiguous conceptual models, allowing the automatic generation of software products [19] using an MDA-

[*] This work has been developed with the support of MEC under the project SESAMO TIN2007-62894 and co financed by FEDER.

based technology. This method has been implemented in a suite of industrial tools by CARE Technologies [5].

The adoption of MDA-based technology has presented new challenges, such as measuring the size of the products that are generated from their conceptual models. This is important because the size of the conceptual model allows that the cost of the application that is automatically generated will be estimated correctly. The Function Point Analysis (FPA) proposal ([9] [10]), together with its adaptations of this measurement method [1] [2] [15] [23] [24], is used to do this. However, these FPA-based approaches have limitations for the measurement of conceptual models used in MDA environments [4] [8] [14]. For instance, FPA-based approaches only allow the measurement of the functionalities from the viewpoint of the human user, ignoring all the functionality that the human user does not see, which should be built for the correct operation of the application.

To overcome the limitations of the initial design of the FPA measurement method, the COSMIC measurement method was defined [3] [13]. COSMIC allows the measurement from different points of view: from the human user viewpoint (like FPA); from the developer viewpoint (including all the functionalities that should be built); and from the viewpoint of any user of the conceptual model. Currently, there are some approaches that apply COSMIC for the purpose of estimating the functional size of future software applications from conceptual models, such as Poels' proposal [22] and Diab's proposal [7]. Both of these FSM procedures were defined establishing a mapping between the COSMIC concepts and their primitives; however their proposals are not compliant with MDA principles.

For industrial MDA development, it is essential to do the measurements quickly and in a precise way because the functional size determines the cost of the generated applications. Therefore, a tool that allows the automatic measurement of conceptual models used in MDA environments is needed to avoid the excessive time and the precision errors involved in a manual measurement process.

This paper introduces the OOmCFP proposal from a practical perspective. This is a procedure to measure the functional size of OO-Method conceptual models based on COSMIC, focusing on the OOmCFP tool, which automates the measurement of OO-Method conceptual models through the implementation of the OOmCFP proposal. The OOmCFP tool includes all the benefits that are related to the COSMIC approach. It allows a better measurement of the conceptual models involved in the OO-Method MDA industrial approach and makes the practical application of the OOmCFP approach possible.

The rest of the paper is organized as follows: section 2 and section 3 present the main concepts of the COSMIC method and the OO-Method approach, respectively. Section 4 presents the OOmCFP measurement procedure and an example of the measurement of an OO-Method conceptual model using the OOmCFP proposal. Section 5 presents the tool that automates the OOmCFP proposal and a comparative analysis of the results obtained in the measurement of conceptual models of real applications. Finally, section 6 presents a discussion on the results achieved as well as suggestions for further work.

2 The COSMIC Functional Size Measurement Method

The COSMIC functional size measurement method can be used to measure any type of software. The application of this measurement method includes three phases: the measurement strategy, the mapping of concepts, and the measurement of the identified concepts.

In the *measurement strategy phase,* the *purpose* and the *scope* of the measurement exercise must be defined. Next, the *functional users,* which are types of users that send (or receive) data to (from) the functional process of the application to be measured must be identified. Finally, the *level of granularity* of the description of the piece of software to be measured is also identified.

In the *mapping phase,* the *functional processes* (the elementary components of a set of functional user requirements) must be identified. Next, the *data groups* must be identified. A data group is a set of data attributes that are distinct, non empty, non ordered, non redundant, and that participate in a functional process. The identification of the *data attributes* of a data group is optional.

In the *measurement phase*, the *data movements* (Entry, Exit, Read and Write) for every functional process must be identified. When all the data movements of the functional process are identified, the measurement function must be applied: this is a mathematical function that assigns 1 CFP to each data movement of the functional process. Then, after all the functional processes are measured, the measurement results are aggregated to obtain the functional size of the piece of software that has been measured.

Figure 1 shows the COSMIC metamodel, which illustrates the information that should be represented by the software artefact to be measured.

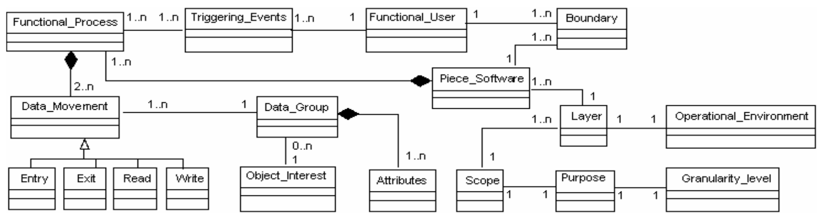


Fig. 1. Metamodel of COSMIC.

3 The OO-Method Approach

OO-Method is a method that allows the automatic generation of software from conceptual models. It has a formal definition supported by OASIS [21], which is an object-oriented, formal specification language for Information Systems. This method is supported by the compiler of OO-Method conceptual models that is implemented in the OlivaNova Suite [5].

The OO-Method model compiler generates applications according to a three-tier software architecture: a tier for the client component, which contains the graphical user interface-related software components; a tier for the server component, which contains the business rules and the connections to the database; and a tier for the database component, which contains the persistence aspects of the applications.

The software production process in OO-Method is represented by three models:

- The Requirements Model, which specifies the system requirements using a set of techniques such as the Mission Statement, the Functions Refinement Tree, the Use Case Model, and the Sequence Diagrams Model.
- The Conceptual Model, which captures the static and dynamic properties of the functional requirements of the system by means of an Object Model, a Dynamic Model, and a Functional Model. The conceptual model also allows the specification of the user interfaces in an abstract way through the Presentation Model. With all of these models, the conceptual model has all the details needed for the automatic generation of the software application. The complete definition of the elements of the conceptual model of OO-Method is described in detail in [19].
- The Execution Model, which allows the transition from the problem space (represented by the conceptual model) to the solution space (the corresponding software product). This model fixes the mappings between conceptual primitives and their corresponding software representations in a target software development environment.

For the purpose of this work, we only need to focus on the Conceptual Model, which is the artifact from which we want to measure the functional size through the corresponding measurement process.

4 OOmCFP: A Measurement Procedure for the OO-Method Conceptual Model

OOmCFP (OO-Method COSMIC Function Points) is a measurement procedure that was developed for measuring the functional size of the OO-Method applications that are based on the MDA approach [16]. In the OOmCFP procedure, the *entity* to be measured is an OO-Method conceptual model, and the *attribute* to be measured is the functional size, which is defined by the ISO/IEC 14143-1 standard as the size of software derived by quantifying the functional user requirements [12].

The OOmCFP was defined in accordance with the COSMIC measurement manual version 3.0 [3]. We selected this functional size method for the design of OOmCFP for the simplicity with which it quantifies functional size without being limited by maximum values, as occurs in other standards (IFPG FPA, NESMA FPA or MARK II FPA). Given that the OOmCFP procedure was designed in accordance with COSMIC, a mapping between the concepts used in COSMIC and the concepts used in the OO-Method conceptual model has been defined (Table 1). It is important to note that the mapping has been done in only one direction since only some of the elements of the conceptual model are relevant to the measurement of the functional size when COSMIC is used.

Table 1. Results obtained from the mapping between COSMIC and OO-Method.

OOmCFP
Purpose: To Measure the functional size of the OO-Method conceptual models to estimate the cost of the applications specifically generated by the OlivaNova Suite.
Scope: The OO-Method conceptual model, which has all the functionality details from which the final software application will be built.
Granularity Level: Low level, since all the details in the OO-Method conceptual model are needed to generate the applications.
Layers: The Client component, the Server component, and the Database component of an OO-Method application since each component is generated for a specific software environment.
Pieces of Software: The Client component, the Server component, and the Database component of an OO-Method application since every layer has at least one piece of software.
Functional Users: - Human users are functional users of the client component of an OO-Method application since data is sent (or receive) to (from) this component. - The Client component of an OO-Method application is a functional user of the Server component of the application. This user is called *client functional user*. - The Server component of an OO-Method application is a functional user for both the Client component and for the Database component of the OO-Method application. This user is called *server functional user*.
Boundaries: The OO-Method applications have three boundaries that separates the users from the layers: one boundary between the human user and the Client component; one boundary between the client functional user and the Server component; and one boundary between the server functional user and the Database component – see Figure 2.
Triggering Events: - The human functional user carries out triggering events that occur in the real world. - The client functional user carries out triggering events that occur in the interaction units of the presentation model of the OO-Method conceptual model. - The server functional user carries out the triggering events that occur in the server component of the software.
Functional Processes: Direct successors of the menu of the presentation model of OO-Method conceptual model. Every child represents a single functional process, either a selection of a given class population (a Population Interaction Unit (PIU)) or an execution of a service (a Service Interaction Unit (SIU)). These interaction units can be combined into more complex interaction units (as a Master Detail Interaction Unit (MDIU)).
Data Movements: The data movements that can occur in the OO-Method applications are shown in Figure 2. Note that the *write* and *read* data movements only can occur between the server functional user and the database component of an OO-Method application.
Data Groups: The classes of the object model of the OO-Method conceptual model, which are used in the functional process.
Data Attributes: The set of attributes of each class that is identified as a Data Group.

Once the mapping between COSMIC and OO-Method has been defined, the measurement rules of the OOmCFP must also be defined. These rules are the rules that assign a numerical value to the data movements that take place between the functional users and the software components of an OO-Method application. The data movements that can occur in the OO-Method applications are shown in Figure 2.

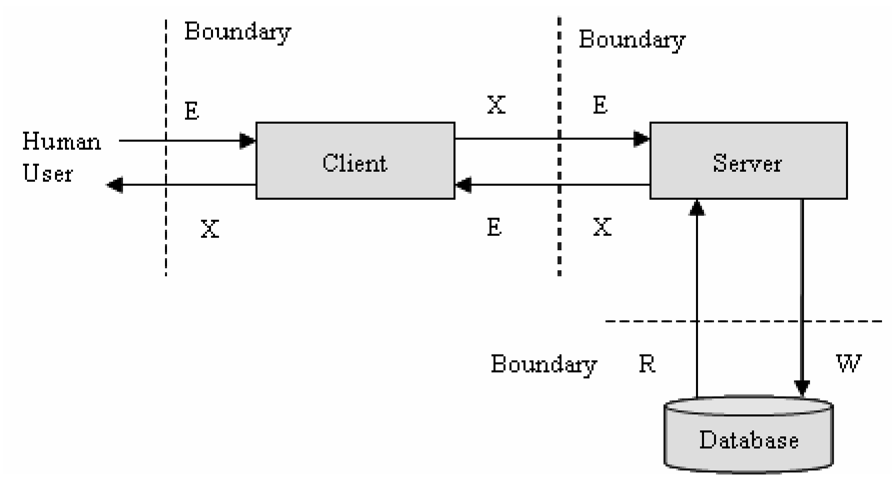


Fig. 2. Data movements that can occur in the functional processes of an OO-Method application.

Given that the applications generated from the OO-Method conceptual model has a three-tier architecture, three types of entry (E) data movements can occur in a functional process: from the human user to the client component of the application; from the client component of the application to the server component of the application; and from the server component of the application to the client component of the application (see Figure 2). A set of measurement rules has been defined for each type of entry data movements.

Three types of exit (X) data movements can occur in a functional process: from the client component of the application to the human user; from the client component of the application to the server component of the application; and from the server component of the application to the client component of the application. Figure 2 shows the exit data movements. A set of measurement rules has been defined for each type of exit data movements.

Only one type of read (R) data movements can occur in OO-Method applications: only the server component of the software can read the persistence storage (see Figure 2). A set of measurement rules has been defined for this type of data movements.

Only one type of write (W) data movements can occur in OO-Method applications: only the sever component of the software can write to the persistence storage (Figure 2). A set of measurement rules for the write data movements has been defined.

According to the COSMIC functional size measurement method, each data movement will be assigned one size unit, which is referred to as 1 CFP. To measure the functional size of a functional process, the functional size of all the data movements of the functional process should be added – see formula (1).

$$\text{SizeFunctionalProcess} = \sum_{i=1}^{n} \mathit{DataMovement}_i \tag{1}$$

Once all the functional processes are measured with formula (1), then all the measurements should be added to obtain the functional size of the layer that contains these functional processes – see formula (2).

$$\text{SizeLayer} = \sum_{i=1}^{n} \mathit{SizeFunctionalProcess}_i \tag{2}$$

To measure of the generated software applications from the developer's viewpoint, it is necessary to add the functional size of every layer. This calculation is represented in formula (3).

$$\text{SizeOOMethodApplication} = \sum_{i=1}^{n} \mathit{SizeLayer}_i \tag{3}$$

Finally, with the three formulas, it is possible to measure the functional size of the OO-Method software applications that are generated from their conceptual model in an MDA environment. The measurement rules include all the functionalities needed by the application for its correct operation; in other words, it includes all the functionalities from the developer's viewpoint.

In terms of the validation of the OOmCFP procedure, since the validation of COSMIC (from the perspective of the measurement theory) has been carried out successfully using the DISTANCE framework [6], the theoretical validation of the OOmCFP procedure can be inferred. Moreover, an expert has validated the conformity of the OOmCFP procedure with the COSMIC version 3.0.

4.1 A Measurement Example

Figure 3 shows an example of an OO-Method conceptual model that allows the automatic generation of a fully working application. This application allows the creation and deletion of invoices with their details, and also allows the creation of the customers associated to the invoice. The administrator of the application, which is represented by the class Admin, can execute the services of the application.

The populations: PIU_Admin, PIU_Customer, PIU_Invoice[†], and the master detail MDIU_Invoice are identified as functional processes by applying the mapping rules presented in Table 1.

[†] PIU is the OO-Method acronym for "Population Interaction Unit". A PIU represents an entry-point for the application, through the presentation of a set of instances of a class. An instance can be selected, and the corresponding set of actions and/or navegations specified in the Presentation Model are offered to the user. More details can be found in [19].

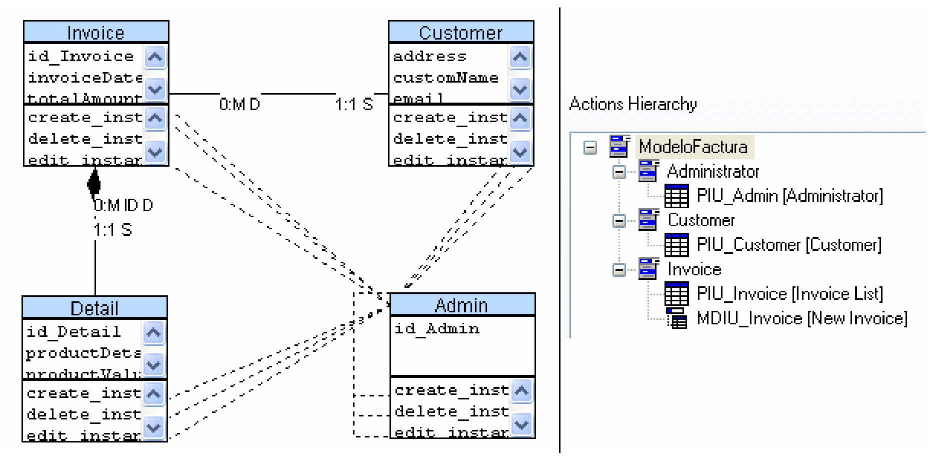


Fig. 3. Example of an OO-Method conceptual model. Left: Object model. Right: Presentation Model.

After identifying the functional processes, the OOmCFP measurement rules are applied to identify the data movements that occur in each functional process. The measurement rules applied to the example are presented in the following table.

Table 2. Measurement rules of OOmCFP applied to the example.

Component	Measurement Rule
Client	**Rule 3.** 1 entry data movement for each different class that corresponds to an argument of a SIU that participates in a functional process.
Client	**Rule 10.** 1 entry data movement for each different class that contributes with attributes to the display set of a PIU or IIU that participates in a functional process.
Client	**Rule 15.** 1 exit data movement for all the attributes that are shown in a display set of a PIU or IIU that participates in a functional process.
Client	**Rule 22.** 1 exit data movement for the set of data-valued arguments of a SIU that participates in a functional process.
Client	**Rule 23.** 1 exit data movement for each different class that corresponds to an argument of a SIU that participates in a functional process.
Server	**Rule 7.** 1 entry data movement for the set of data-valued arguments of a SIU that participates in a functional process.
Server	**Rule 8.** 1 entry data movement for each different class that corresponds to an argument of a SIU that participates in a functional process.
Server	**Rule 25.** 1 exit data movement for each different class that contributes with attributes to the display set of a PIU or IIU that participates in a functional process.
Server	**Rule 30.** 1 read data movement for each different class that contributes with attributes to the display set of a PIU or IIU that participates in a functional process.
Server	**Rule 31.** 1 read data movement for each different class that is used in the derivation formula of the derived attributes of the display set of a PIU or IIU that participates in a functional process.

Component	Measurement Rule
Server	**Rule 35.** 1 read data movement for each different class that is used in the default value formula of an object-valued argument of a service that is related to a SIU that participates in a functional process.
Server	**Rule 36.** 1 read data movement for each different class that is used in the valuation formula of the event that is related to a SIU that participates in a functional process.
Server	**Rule 38.** 1 read data movement for each different class that is used in the formula of the transaction, the operation or the global service that is related to a SIU that participates in a functional process.
Server	**Rule 50.** 1 write data movement for the class that contains a destroy event or transaction that is related to a SIU that participates in a functional process.
Server	**Rule 51.** 1 write data movement for the class that contains a creation event or transaction that is related to a SIU that participates in a functional process.
Server	**Rule 52.** 1 write data movement for the class that contains an event that has valuations and that is related to a SIU that participates in a functional process.

With the measurements rules presented above, we have identified the data movements that occur in the functional processes. The following table shows the data movements that are identified for each functional process in the client component and in the server component of the OO-Method application.

Table 3. Data movements of the functional processes that occur in the client component and in the server component of the OO-Method application.

Functional Process	Client Component				Server component			
	Entry	Exit	Read	Write	Entry	Exit	Read	Write
PIU_Admin	8	8	0	0	6	2	2	4
PIU_Customer	8	8	0	0	6	2	2	5
PIU_Invoice	2	1	0	0	0	2	3	0
MDIU_Invoice	8	11	0	0	8	2	4	6

Once all of the data movements are identified, the formulas defined in OOmCFP are applied to obtain the functional size for each functional process. Thus, Table 4 presents the functional size of the functional process in the client component of the software and in the server component of the OO-Method application.

Table 4. Data movements of the functional processes that occur in the software components of the application.

Functional Process	Client Component	Server Component
PIU_Admin	16	14
PIU_Customer	16	15
PIU_Invoice	3	5
MDIU_Invoice	19	20

Next, by applying formula (2), we can obtain the functional size of each piece of software: the functional size of the client component of the application is 54 cfp; and the functional size of the server component of the application is 54 cfp.

Finally, we obtain the functional size of the OO-Method application by applying formula (3). The resultant functional size is 108 cfp.

It is important to note that the manual measurement of this small example took 70 minutes. Keeping in mind that the model has only four classes, the manual measurement of real applications that may contain 100 or more classes would require at least 116 hours. Therefore, it is very important to automate the measurement procedure to be able to measure efficiently conceptual models of real applications. By automating the measurement procedure many possible human errors could be avoided. The next section presents the development of the tool that automates the OOmCFP procedure.

5 The Automation of the OOmCFP Procedure

Since the automation of the measurement of conceptual models with the OOmCFP proposal is essential, a tool must be developed to implement the measurement rules defined in OOmCFP and to aggregate the results according to the formulas presented in Section 4. The OOmCFP tool has been developed using Visual Studio .Net 2003 with the language C#.

This tool must have a flexible architecture that allows adaptation to the evolution of conceptual models. It must also be agile in the measurement process.

To provide this flexible architecture, the OOmCFP tool was developed with a set of layers that allows easy incorporation of new measurement rules or changes in the existing measurement rules.

The first layer of the OOmCFP tool consists of the pre-charge of the OO-Method conceptual model that is generated from the Olivanova Suite [5] in an XML file. In this layer, the functional elements are organized in a hierarchical way, according to the functional processes identified for the client component and the server component.

In the second layer of the OOmCFP tool, each element that participates in each functional process is identified, and the measurement of the data movements that occur in each functional process are performed through the rules defined in OOmCFP. To reduce the coupling of the measurement of the elements, each rule is grouped by element and is implemented in an independent way. The result of the analysis of each element is stored in the same element.

The third layer of the OOmCFP tool consists of the aggregation of the values of each element according to the formulas defined in the OOmCFP. Thus, the tool obtains the functional size of each functional process, the functional size of each component of the application, and the functional size of the complete application.

The last layer consists of the generation of a final measurement report of the measurement in an XML file. This XML file can be transformed into other formats using XSLT. By default, the OOmCFP tool transforms the XML file in an HTML page.

Since the longest processing time and run time occur in the identification and measurement step of functional elements, we have implemented a cache mechanism to provide agility to the counting process. This reduces the high amount of time required to analyze elements that have already been analyzed. Thus, when a new functional element is identified, the cache mechanism verifies whether or not it already exists. It so, the value of the measurement is recovered.

To avoid overflow, the related elements are stored in an auxiliary array (Figure 4). Once the analysis of the first element is finished, the analysis of the elements stored in the auxiliary array continues sequentially. If the related elements are also related to other elements, these elements are added at the end of the auxiliary array, eliminating the loop of iterations and avoiding overflow.

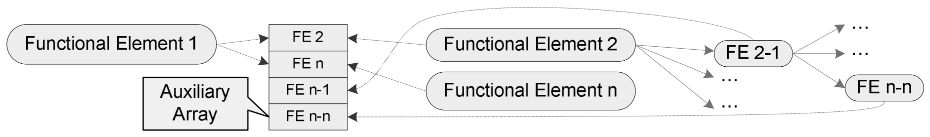


Fig. 4. Schema of the solution to avoid overflow problems.

The architecture of the OOmCFP tool provides an efficient measurement process. Therefore, the measurement of conceptual models that generate real applications is done in a few seconds, thus expediting the process of estimating the cost of the final application.

The precision of the measurement is defined as the closeness of agreement between quantity values obtained by replicated measurements of a quantity under specified conditions [11]. In general, it is not possible to ensure precision in manual measurements since people can make mistakes in the identification of the functional process, the application of the measurement rules, or even the application of the formulas. In contrast, when a tool performs the measurement, it can ensure the precision of the measures because it is an automated measurement where a precise procedure will always produce the same result in any measurement task. Consequently, the OOmCFP tool avoids the errors of the manual measurements and assures the precision of the measurements.

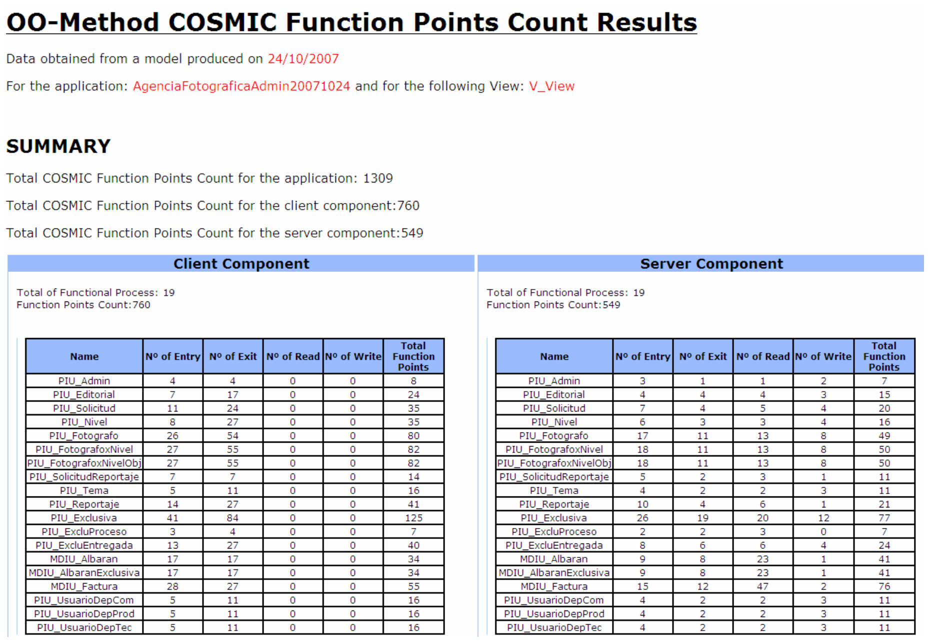
5.1 Using the OOmCFP Tool

The steps for using the OOmCFP tool are the following:

The first step is to load the XML file that contains the OO-Method conceptual model and to specify the path where the report will be saved.

The second step is to show a summary of the model that will be measured and the path of the report.

The third step is to show the number of functional processes that have been measured and the function points for every layer of the application. In this step, the report with all the results of the measurement is saved in the path indicated in the first step. Figure 6 shows the report generated by the OOmCFP tool.



OO-Method COSMIC Function Points Count Results

Data obtained from a model produced on 24/10/2007

For the application: AgenciaFotograficaAdmin20071024 and for the following View: V_View

SUMMARY

Total COSMIC Function Points Count for the application: 1309

Total COSMIC Function Points Count for the client component:760

Total COSMIC Function Points Count for the server component:549

Client Component

Total of Functional Process: 19
Function Points Count:760

Name	Nº of Entry	Nº of Exit	Nº of Read	Nº of Write	Total Function Points
PIU_Admin	4	4	0	0	8
PIU_Editorial	7	17	0	0	24
PIU_Solicitud	11	24	0	0	35
PIU_Nivel	8	27	0	0	35
PIU_Fotografo	26	54	0	0	80
PIU_FotografoxNivel	27	55	0	0	82
PIU_FotografoxNivelObj	27	55	0	0	82
PIU_SolicitudReportaje	7	7	0	0	14
PIU_Tema	5	11	0	0	16
PIU_Reportaje	14	27	0	0	41
PIU_Exclusiva	41	84	0	0	125
PIU_ExcluProceso	3	4	0	0	7
PIU_ExcluEntregada	13	27	0	0	40
MDIU_Albaran	17	17	0	0	34
MDIU_AlbaranExclusiva	17	17	0	0	34
MDIU_Factura	28	27	0	0	55
PIU_UsuarioDepCom	5	11	0	0	16
PIU_UsuarioDepProd	5	11	0	0	16
PIU_UsuarioDepTec	5	11	0	0	16

Server Component

Total of Functional Process: 19
Function Points Count:549

Name	Nº of Entry	Nº of Exit	Nº of Read	Nº of Write	Total Function Points
PIU_Admin	3	1	1	2	7
PIU_Editorial	4	4	4	3	15
PIU_Solicitud	7	4	5	4	20
PIU_Nivel	6	3	3	4	16
PIU_Fotografo	17	11	13	8	49
PIU_FotografoxNivel	18	11	13	8	50
PIU_FotografoxNivelObj	18	11	13	8	50
PIU_SolicitudReportaje	5	2	3	1	11
PIU_Tema	4	2	2	3	11
PIU_Reportaje	10	4	6	1	21
PIU_Exclusiva	26	19	20	12	77
PIU_ExcluProceso	2	2	3	0	7
PIU_ExcluEntregada	8	6	6	4	24
MDIU_Albaran	9	8	23	1	41
MDIU_AlbaranExclusiva	9	8	23	1	41
MDIU_Factura	15	12	47	2	76
PIU_UsuarioDepCom	4	2	2	3	11
PIU_UsuarioDepProd	4	2	2	3	11
PIU_UsuarioDepTec	4	2	2	3	11

Fig. 6. Report generated by the OOmCFP tool.

We have verified how the OOmCFP tool works in practice using some predefined OO-Method conceptual models.

5.2 A Comparative Analysis of COSMIC and FPA

A preliminary comparative analysis has been carried out with respect to the functional size of five conceptual models used in real applications. These conceptual models belong to the Management Information System domain.

The analysis compares the functional sizes obtained for the OO-Method conceptual models using the COSMIC and FPA techniques. OOmCFP is used as the COSMIC measurement procedure, and OOmFP [1] is used as the FPA measurement procedure.

Before the analysis was carried out, we formulated the following hypothesis:

H1: The functional size of the measurement of an OO-Method conceptual model using a COSMIC-based approach is bigger than the measurement using a FPA-based approach.

Table 5 shows the results obtained. The *Model* column shows an identifier for each model; the *Classes* column shows the number of classes associated to each model; the *OOmCFP* column shows the number of function points obtained with the OOmCFP procedure; the *OOmFP* column shows the number of function points obtained with the OOmFP procedure; and the last column *Dif* shows the difference between the results obtained with the OOmCFP procedure and those obtained with the OOmFP procedure.

Table 5. Functional size of OO-Method conceptual models measured with the OOmCFP approach and the OOmFP approach.

Model	Classes	OOmCFP	OOmFP	Dif
M1	9	836	463	373
M2	17	357	308	49
M3	30	2019	1158	861
M4	83	4326	2811	1515
M5	193	14649	6267	8382

As Table 5 shows, the functional size obtained was bigger when the OOmCFP approach was used. Therefore, as expected, hypothesis H1 is true since more aspects are taken into account when COSMIC is used as the measurement strategy. This demonstrates that when COSMIC is used, a better measurement of the functionality generated from the conceptual models in MDA environments is obtained.

Table 5 also shows that the differences obtained in the comparative analysis do not follow a common pattern, because the measurement approaches analyzed use different conceptual elements to quantify the functional size of the applications.

An important final consideration is that the complexity of the conceptual model is not measured by either the COSMIC approach or by the IFPUG approach. However, we believe that the conceptual elements that COSMIC uses in the measurement of the functional size can be used to define metrics to measure the complexity of the conceptual models.

6 Conclusions and Further Work

In this paper, we have introduced OOmCFP, which is an FSM procedure for object-oriented applications generated in MDA environments. OOmCFP allows the measurement of the functional size in the conceptual models that will be transformed in the generated applications. Therefore, we consider that OOmCFP specifies the functional requirements for the development of a tool to automate the measurement of the functional size of applications generated in MDA environments.

We have presented the OOmCFP tool, which automates the OOmCFP procedure. To develop the OOmCFP tool, a set of aspects has been taken into account. These aspects are related to the performance of the functional measurement process and implementation aspects. The performance aspects are focused on the reduction of the measurement time. The implementation aspects consider a correct execution of the measurement process avoiding the overflows that can be produced by the measurement of large OO-Method conceptual models.

A measurement example demonstrates how the OOmCFP tool is more efficient than the measurements that are performed manually. This example also shows how the OOmCFP tool can obtain precise measurements for the OO-Method conceptual models.

A comparative analysis shows how a COSMIC-based approach, like OOmCFP, provides a better measurement of the conceptual models that are used in an MDA environment. This is because, in contrast to other FSM standards like IFPG FPA, NESMA FPA or MARK II FPA, it allows the functional size measurement of multi-layer applications (like the applications modeled with the OO-Method approach) from different viewpoints.

Further work will include empirical studies of the reproducibility and the repeatability of OOmCFP. It also include the analysis of the rules presented in this work in order to take into account different points of view (such as the human user viewpoint) for the measurement of applications generated in MDA environments.

References

1. Abrahão, S. and Pastor, O., *Estimating the Applications Functional Size from Object-Oriented Conceptual Models* In: International Function Point User Group Annual Conference (IFPUG'01), Las Vegas, USA, 2001.
2. Abrahão, S. and Pastor, O., *Measuring the functional size of web applications*, International Journal of Web Engineering and Technology (IJWET) 1(1), 2003, pp. 5-16.
3. Abran, A.; Desharnais, J.; Lesterhuis, A.; Londeix, B.; Meli, R.; Morris, P.; Oligny, S.; O'Neil, M.; Rollo, T.; Rule,G.; Santillo, L.; Symons, C.; Toivonen, H., *The COSMIC Functional Size Measurement Method, version 3.0* In GELOG web site www.gelog.etsmtl.ca
4. Abran, A.; Pierre, N., *Function Points: A Study of Their Measurement Processes and Scale Transformations*, Journal Systems and Software 25(2), 1994, pp. 171-184.
5. CARE Technologies Web Site, www.care-t.com
6. Condori-Fernández, N., *Un procedimiento de medición de tamaño funcional a partir de especificaciones de requisitos*, Doctoral thesis, Universidad Politécnica de Valencia, Valencia, España, 2007.
7. Diab, H.; Koukane, F.; Frappier, M.; St-Denis, R., *μcROSE: Automated Measurement of COS-MIC-FFP for Rational Rose Real Time*, Information and Software Technology 47(3), 2005, pp. 151-166.
8. Giachetti, G.; Marín, B.; Condori-Fernández, N. and Molina, J.C., *Updating OO-Method Function Points* In: 6th IEEE International Conference on the Quality of Information and Communications Technology (QUATIC 2007), Lisboa, Portugal, 2007, pp. 55-64.
9. IFPUG: International Function Point Users Group Web Site, www.ifpug.org
10. IFPUG, *Function Point Counting Practices Manual Release 4.1*, International Function Point Users Group, Westerville, Ohio, USA, 1999.
11. ISO, *International vocabulary of basic and general terms in metrology (VIM)*, International Organization for Standardization, Geneva, Switzerland, 2004.
12. ISO, ISO/IEC 14143-1, *Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts*, 1998.
13. ISO, ISO/IEC 19761, *Software Engineering – CFF – A Functional Size Measurement Method*, 2003.
14. Kitchenham, B., *Counterpoint: The Problem with Function Points*, IEEE Software Status Report 14(2), 1997, pp. 29-31.
15. Lehne, A., *Experience Report: Function Points Counting of Object-Oriented Analysis and Design based on the OOram method* In: Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'97), Atlanta, Georgia, October 1997.
16. Marín, B.; Condori-Fernández, N.; Pastor, O. and Abran, A., *Measuring the Functional Size of Conceptual Models in a MDA Environment.* Accepted in the 20th

International Conference on Advanced Information Systems Engineering (CAiSE 2008), Montpellier, France, 2008.
17. OMG: Web site of MDA, http://www.omg.org/mda/
18. OO-Method Group Web Site, http://oomethod.dsic.upv.es
19. Pastor, O. and Molina, J. C., *Model-Driven Architecture in Practice*, Springer, 2007.
20. Pastor, O.; Gómez, J.; Insfrán E. and Pelechano, V., *The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming*, Information Systems, vol.26, 2001.
21. Pastor, O., Hayes, F. and Bear, S., *OASIS: An Object-Oriented Specification Language* In: 4th International Conference on Advanced Information Systems Engineering (CAiSE 1992), Manchester, UK, 1992, pp. 348-363.
22. Poels, G., *Functional Size Measurement of Multi-Layer Object-Oriented Conceptual Models* In: Proceedings of 9th International Object-Oriented Information Systems Conference, Geneva, Switzerland, 2003, pp. 334-345.
23. Tavares, H.; Carvalho, A.; Castro, J., *Medicao de Pontos por Funcao a partir da Especificao de Requisitos* In: Workshop on Requirements Engineering, Universidad Politécnica de Valencia, Spain, November 2002, pp. 278-298.
24. Uemura, T.; Kusumoto, S. and Inoue, K., *Function Point Measurement Tool for UML Design Specification* In: 5th International Software Metrics Symposium, IEEE METRICS, Florida, USA, 1999, pp. 62-71.