Verifying the Construction of a Software Model from a Requirements Model

Nelly Condori-Fernández, Oscar Pastor
Department of Information Systems and Computation,
Valencia University of Technology,
Camino de Vera s/n, 46020, Valencia-Spain
{nelly,opastor}@dsic.upv.es

Abstract

A software model is the outcome of abstracting a set of relevant elements that contribute to the functional size according to measurement model.

The purpose of this paper is to verify the construction of the software model when applying the RmFFP measurement procedure with computer science students. The RmFFP procedure was designed according to the COSMIC-FFP standard method for estimating the functional size of object-oriented systems from requirements specifications obtained in the context of the OO-Method approach.

1. Introduction

Currently, various studies on software development conclude that the most critical tasks arise during the specification and analysis of requirements. Consequently, errors occurring in the initial phase of the production process can have a considerable impact on the reliability of estimation models whose key parameter is software size. Software size can be derived by means of the quantification of functional user requirements [1]. There are several Functional Size Measurement (FSM) methods that have used as a starting-point the Function Point Analysis (FPA) method [2], such as MARK II FPA [3] NESMA FPA [4] and COSMIC-FFP [5].

However, these measurement methods are complex and not easy to use due to the over-generalised nature of their measurement manuals. For this reason there is currently increasing interest in designing measurement procedures.

A measurement procedure is defined as a set of operations, specifically described, used in the performance of particular measurements in accordance with a given method [6].

RmFFP is a FSM procedure designed on the basis of the COSMIC-FFP standard method, which has received ISO/IEC 19761 approval [5].

This FSM procedure was defined to estimate the functional size of the applications generated with OO-Method from a requirements specification [7]. To do this, a set of mapping rules has been defined to facilitate the construction of the software model [8]. This software model is the abstraction of the relevant primitives of the requirements model that contribute to the functional size according to the COSMIC-FFP metamodel.

The purpose of this paper is to verify whether the construction of the software model is reliable. This verification is carried out by means of the application of RmFFP in a "Rent a Car" case study by various computer science students.

The paper is structured as follows: Section 2 discusses related work. Section 3 describes the steps set of the RmFFP process. Section 4 presents an analysis of the reliability of the data movement identification. Finally, conclusions are presented and future work is considered.

2. Related work

In the literature we can identify basically two generations of FSM methods.

The first generation methods consider only the end user viewpoint when carrying out a specific measurement. A disadvantage of this is that it does not always cover all the system's functionality. These standard methods are IFPUG-FPA [9] MARK II FPA [3] and NESMA FPA [4].

The second generation is represented by one measurement method, COSMIC-FFP [5], which was designed for various software domains such as information systems and real-time systems. This method extended the concept of user and made a distinction between the end user viewpoint and the development viewpoint.

One disadvantage of these FSM methods is their generic character that makes applying them in particular contexts difficult. For this reason different FSM procedures have been designed to be applied in specific measurements according to a given method.

Table 1 shows the different second generation proposals that have been found in the literature, which establish a mapping between the primitives of the various software artefacts and the relevant concepts of the COSMIC-FFP metamodel.

As shown in Table 1, the proposals of Bevo et al [10], Jenner [11], and Habela [12] consider different UML diagrams for the construction of the software model. They do not consider a development method, which is a disadvantage for the construction of the diagrams. Another disadvantage of the proposals of Bevo et al and Jenner is the lack of clarity when

identifying certain basic components that contribute to functional size.

The proposals of Poels [13], Diab et al [14] and Nagano et al [15] consider software artefacts produced in the analysis phase of their respective methods (MERODE, RRRT, Shalaer-Mellor) in order to construct the software model. Diab and Nagano instantiate the COSMIC-FFP meta-model for real-time systems.

Table 1. COSMIC-FFP measurement procedures

Proposals	Context	Software artefacts	Phase of life cycle
Bévo et al.	UML	Use case diagram and classes diagram	Requirements and Analysis
Jenner	UML	Sequences diagram and use case diagram	Requirements
Poels	MERODE	Business model and services model	Analysis
Diab et al.	RRRT	States-chart diagram	Analysis
Azzouz and Abran	RUP	Use case diagram, sequence diagram, and class diagram.	Requirements, Analysis and Design
Habela et al.	UML	Use case model	Design
Nagano et al.	Shlaer- Mellor	Class diagrams, state-chart diagram and collaboration diagrams	Analysis
Condori et al.	OO-Method	Requirements model: sequence diagrams and use case diagrams.	Requirements

Azzouz and Abran [16] consider three size units that correspond to development phases. They use different diagrams for the construction of the software model, considering the design phase to be a better phase in which to estimate functional size, allowing a closer aproximation to real size.

Finally, our proposal, RmFFP [8], uses different diagrams of the OO-Method Requirements Model in order to estimate functional size from early stage of the life cycle. Nevertheless, the difficulty encountered in the proposals of Bevo, Jenner and Azzouz in clearly identifying certain significant concepts of COSMIC-FFP is resolved in our proposal by means of the stereotypes incorporated in the messages of the sequence diagrams. In addition we also control the duplicity problems which were identified in some primitives of the requirements model [17].

In the next section, we introduce the application of RmFFP by means of a steps set proposed by Jacquet and Abran [18].

3. Application of the RmFFP procedure

Jacquet and Abran define a measurement process model [18], which includes measurement method design, its application, analysis of the results obtained and its utilization in estimation models (See Figure 1).

In the first step, a measurement method is designed, the concept to be measured is defined and the rules to measure it are conceived. In the second step, the measurement method is applied to measure the size of software applications. In the third step, the results provided by the measurement method are presented and verified (i.e. the results can be compared to other well-known results in order to try to evaluate their correctness). Finally, the results are used in different types of models (e.g., productivity-analysis models, effort-estimation models, budgeting models).

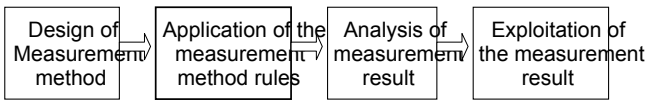

Figure 1. Measurement Process Model [18]

Therefore, ensuring high quality measurement results relies not only on design quality but also the quality of application. The application of an FSM procedure is an intellectual process that consists of abstracting the relevant primitives of the abstract artefact to be measured according to the measurement model, and quantifying the elements abstracted in order to obtain the functional size.

According to Jacquet and Abran, three steps are required in order to apply a measurement procedure: software documentation gathering, construction of the software model, and the application of numerical assignment rules.

Figure 2 shows these steps adapted in order to apply the RmFFP procedure in the OO-Method context [19].

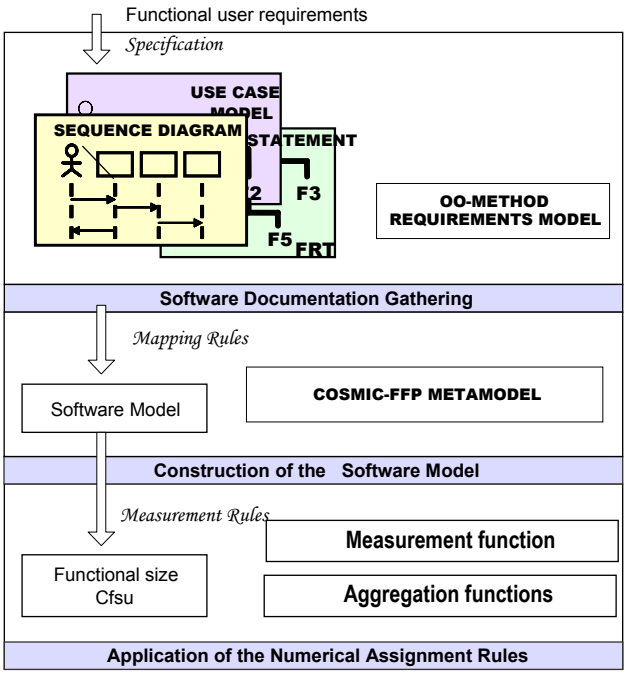

Figure 2. Application of the RmFFP procedure

3.1. Gathering of the software documentation

From the perspective of the functional size measurement methods, the aspect of most interest is functionality, which means 'what the software should do'. This functionality may be documented by software artefacts produced prior to implementation; or it may be that this documentation is not available. If the latter is the case, functional requirements can be derived from artefacts installed on the computer system even after they have been implemented (See Figure 3).

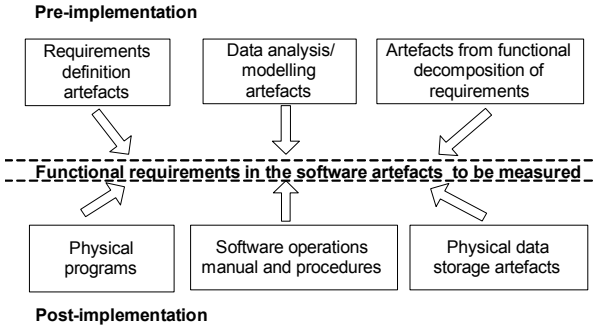

Figure 3. Functional User Requirements Model [21]

RmFFP uses the system functionality specified in a semi-formal way using the OO-Method Requirements Model. As shown in Figure 4, the Requirements Engineering phase culminates with the obtaining of the sequence diagrams. This diagram model is the principal artefact on which the measurement will be carried out. For this reason specification quality will affect the quality of results of the measurement.

OO-Method assures the traceability and consistency of the functional specification by means of the semiautomatic generation of the sequence diagrams model from the use case model [20].

- Traceability: it is possible to accurately determine the impact caused in the sequence diagrams model when changes are carried out in the use case model and vice versa. Thanks to this traceability it is possible to estimate the functional size with greater accuracy at an earlier phase. There will be a greater degree of proximity between the size obtained from the requirements specification and the size of the final application.
- Consistency: the deduction of each sequence diagram from the use case model is always carried out using the same criteria. As these criteria are inherent to the development method and are independent of subjective reasoning, the functional size obtained will not be affected by the different levels of detail that may be specified. This consistency will also contribute to the accuracy and reliability of the functional size.

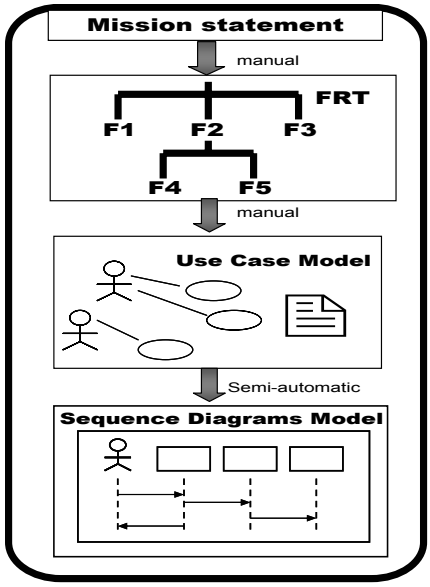

Figure 4. Requirements engineering phase

Figure 4 shows the OO-Method Requirements Engineering phase, which starts with the definition of the *Mission Statement* that describes the purpose of the system and its main functionalities. The Functions Refinement Tree (FRT) is then obtained by means of a hierarchical decomposition of the business functions of the system.

The leaves of this tree represent the entry point for building the Use Case Model, which models the system's functional requirements from the user's perspective. The leaf nodes of the FRT are considered to be primary use cases. It is also possible to have secondary use cases, which are important for organizing and managing complexity through

relationships among use cases that are stereotyped as EXTEND and INCLUDE. The construction of the use case model is carried out manually.

Finally, the sequence diagrams are built semi-automatically from each use case. The notation of these sequence diagrams is provided for UML with some stereotypes incorporated to classify the different types of interaction, such as: signal, service, query and connect [7] (see Figure 5).

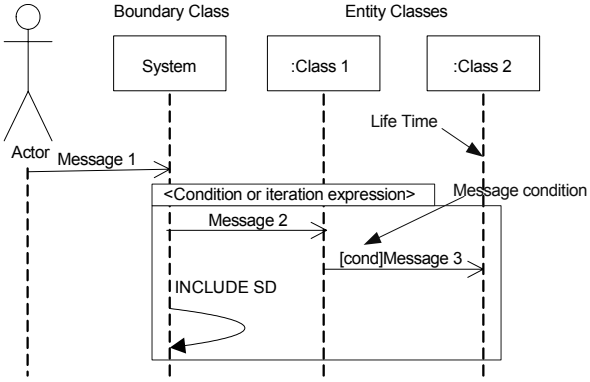

Figure 5. Structure and notation of sequence diagram

3.2. Construction of the software model

The software model is built once the documentation of the system has been gathered. This model describes how the software to be measured is represented according to the measurement method.

With RmFFP, as shown in Figure 2, the construction of the software model includes the instantiation of the COSMIC-FFP metamodel in a particular context, such as OO-Method.

This metamodel has been elaborated in order to clearly represent the diverse generic concepts of COSMIC-FFP presented in the measurement manual [21], and also to identify the relationships existing between these concepts.

Figure 6 shows the COSMIC-FFP metamodel, which has been represented by means of the UML class diagram, chosen because of its simplicity, expressiveness and popularity.

As can be observed in the metamodel, the object of interest to be measured can be identified from many measurement viewpoints. The viewpoint determines the level of detail that can be seen in an object of interest (e.g. the measurement viewpoint of the developer). This viewpoint is also determined by the purposes of the functional size measurement. The measurement purpose defines why the measurement is being undertaken and what the result will be used for. The purpose helps the measurer to determine the scope to be measured; hence, measurement scope is the functionality to be included in a particular functional size measurement.

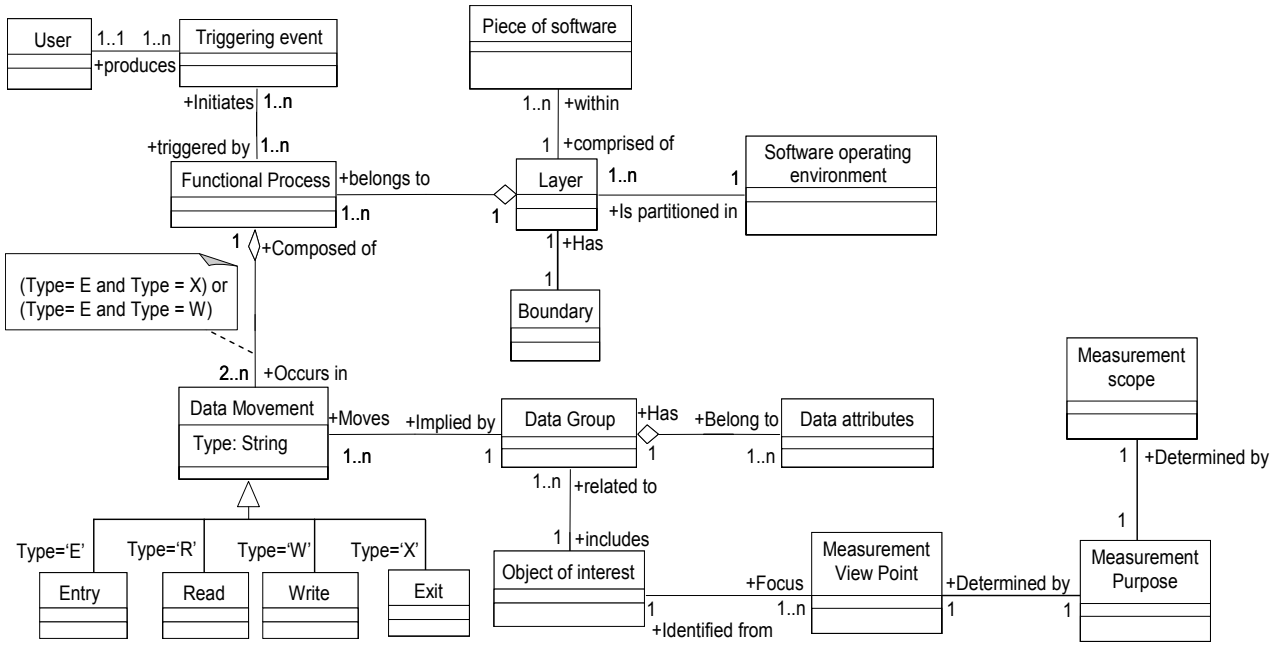

Figure 6. Cosmic-FFP metamodel

The object of interest may be any physical thing or any conceptual object described by a set of attributes that belong to a data group. Each data group must be directly related to a single object of interest.

For this reason, an aggregation relationship has been used between the concepts of data group and data attribute. In addition, the minimum cardinality is one because an object of interest cannot be empty. A data attribute is the smallest piece of information that

belongs to an identified data group. A data movement occurs in a functional process and moves a data group. As shown in Figure 6, the data movements can be of four types: entry, read, write, and exit. For instance, an entry moves a data group from a user across the boundary into the functional process where it is required. Each functional process is composed of a minimum of two data movements: one Entry, and one Exit or one Write. This is represented by means of the minimum cardinality of two in the "Occurs_in" role between the Functional Process and Data Movement classes.

The set of functional processes performed at the same level of abstraction constitutes the concept of layer. A layer is the result of the functional partitioning of the software operating environment and can be divided into one or more pieces of software. The software operating environment is the set of software that is operating concurrently on a specified computer system. In a multi-layer software environment, each layer is a user of another layer because a layer uses the functional services provided by other subordinate layers. A user is any person or thing that communicates or interacts with the software at any time. Finally a triggering event is an event that initiates one or more functional processes; these events are triggered directly or indirectly by any user.

To facilitate the instantiation of this metamodel, a set of mapping rules were defined [8]. The application of these rules allows us to obtain the software model to be measured. Nevertheless, some questions arise, such as: what concept of the metamodel should be first instantiated, and what concepts could be instantiated in a parallel way. To respond to these questions, an activities diagram has been constructed in order to represent the operations sequence of the RmFFP procedure.

The RmFFP process starts with the definition of the measurement context, which includes three activities: the identification of purpose, viewpoint and scope of the measurement. The mapping phase is then carried out in order to construct the respective software model, which is guided by means of a set of activities specified in Figure 7. Each activity of this phase is realized by a set of mapping rules. This phase culminates with the identification of data movement types, which are constituted as the basic components of COSMIC-FFP.

Taking into account the COSMIC-FFP metamodel (Figure 6) and the activities diagram (Figure 7), we verify that the concepts of layer and triggering event were not instantiated explicitly in some primitive of the Software Requirements Model. The identification of a triggering event is not an indispensable activity, since this activity contributes to the identification of the functional processes, which are already clearly identified by the Users (Rule 1) and the boundary (Rule 2). The identification of layers is also not necessary given that the functional requirements have the same abstraction level; thus there is no functional division of the operating environment of the software. In addition, the identification of data attributes is carried out only if a measurement sub-unit is required.

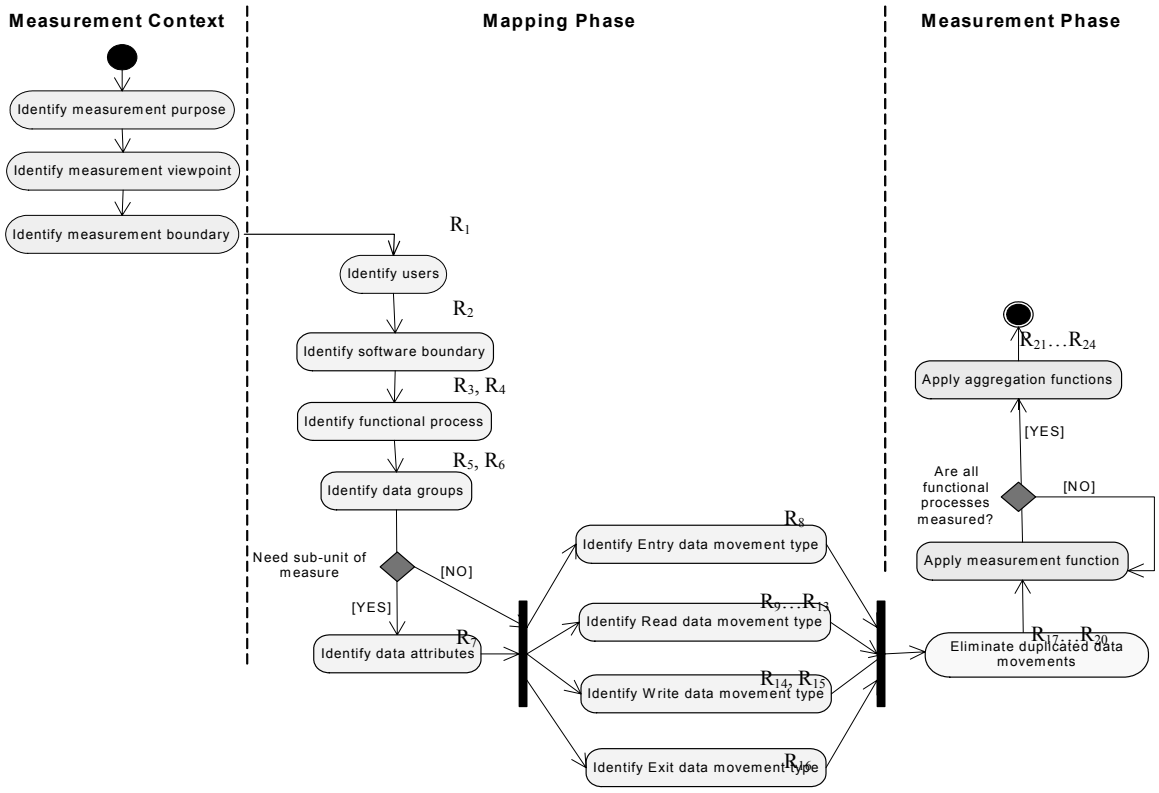

Figure 7. Activity diagram of RmFFP

3.3. Application of the rules of numerical assignment

As shown in Figure 7, duplicated data movements are eliminated before the application of the measurement function. In order to carry out this elimination activity a set of rules were defined to avoid pseudo data movements.

The purpose of the measurement phase is to quantify the software model built in the mapping phase. To do this, we apply the measurement function and the measurement rules that were defined for the respective aggregation functions. Finally, the functional size is obtained in Cfsu (Cosmic functional size unit) units.

As shown in Figure 2, the software model is constituted as a by-pass for the quantification of this model (step 3), which was built from documented software (step 1). The quality of the construction of the software model is therefore important in the obtaining of reliable results. To ensure the quality of the software model, we have to ensure the correct application of the respective rules defined in the carrying out of each activity of this phase. As shown in Figure 7, the data movements identified are constituted as entries for the next phase. For this reason, we analyze the reliability of the identification of data movements, for which we carried out an initial empirical study which is described in the next section.

4. Analyzing reliability in the identification of data movements

In order to analyze reliability in the identification of data movements, the RmFFP procedure was used by twenty-three computer science students at the Valencia University of Technology who had similar backgrounds in the use of the OO-Method Requirements Model. These students were selected by convenience, i.e., they were students enrolled in the "Software Development Environments" course.

To carry out this descriptive analysis, it was first necessary to plan a training session in order to develop skills in measurement using RmFFP with the 22 students. Having a sufficient level of knowledge of the OO-Method Requirements Model was a prerequisite for using RmFFP. This training session fitted well into the scope of the "Software Development Environments" course.

Reliability was verified in terms of reproducibility, which is defined as the proximity between the results of measurements of the same measurand carried out by different subjects [22].

To quantify reproducibility, firstly we collected data obtained by each student for the requirements specification of the "Rent a car" case study. The measurement of this case study was carried out at the data movement type level.

Secondly, to analyze the degree of variability in the measurements, we apply the equation proposed by Kemerer [23]:

$$REP_i = \left| \frac{\sum_{k=1,k\neq i}^{n} \frac{Size_k}{n-1} - Size_i}{\sum_{k=1,k\neq i}^{n} \frac{Size_k}{n-1}} \right|$$

This equation was calculated by taking the difference in absolute value between the size value produced by a subject i and the average value produced by the other n- 1 subjects in the sample, divided by this average value. The scores (REPi) closest to zero indicate the least variability in the measurement, and thus the greatest reproducibility.

Table 2 shows the variability obtained in each data movement type.

Table 2. Variability of data movement type

Variability	Entry	Read	Write	Exit
Minimum	0.018	0.008	0.004	0.000
Maximum	0.410	0.149	0.081	0.000
Standard dev.	0.095	0.048	0.020	0.000
Mean	0.117	0.046	0.020	0.000

As shown in Table 2, the average variability obtained for the ENTRY data movement type is slightly higher when compared to the variability of other data movement types. To investigate this minor difference, we checked the application of Rule 8 to identify possible causes of this variability. We found that some students had difficulty in identifying the data groups involved in an Entry data movement type. In relation to this problem, we identified two guidelines to assist in the identification of entry data movements in the messages with the stereotype <<signal>>:

- All messages have, at least, n parameters (p1, p2,..pn). The parameters involved in messages that have the entity class type as receiver class, are attributes of this receiver class. However, the parameters involved in the messages with a receiver class of boundary type (System) can be attributes of various classes of entity type. Therefore, the data movement identification in this message type is not so evident. According to the COSMIC-FFP manual [19], a data movement moves one or more data attributes that belong to a single data group. We identified two interaction fragment types that illustrate the data group involved in the signal messages.
  - The reception of the message <<signal>> induces the system class to send "n messages" to "n lines of life" with at least one parameter p_j (See Figure 8). Therefore the number of data groups involved in the message signal is determined by the number of messages induced by the class system.

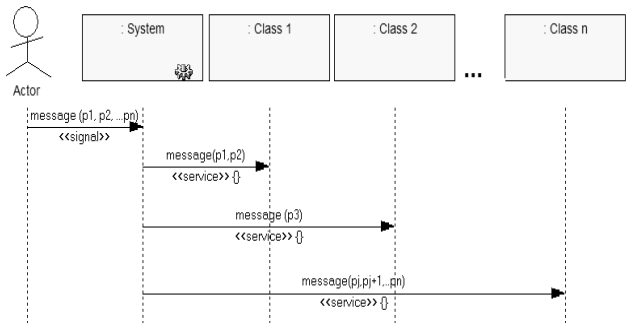

Figure 8. Type 1 interaction fragment

- The occurrence of connect messages in a scenario is conditional on the prior occurrence of the service message. The connect message is activated when the service message needs to establish or to eliminate links among the class objects (See Figure 9). Therefore, the number of data groups involved in the signal message is determined by the receiver class of the service message plus the receiver class of the connect message.

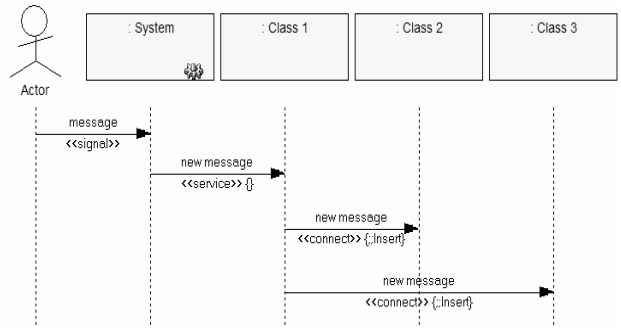

Figure 9. Type 2 interaction fragment

- Scenario start messages will not be considered as the entry data movement type. This is an exception to Rule 8, because this rule permits acceptance of every message labelled with the stereotype signal and input value as an entry data movement type.

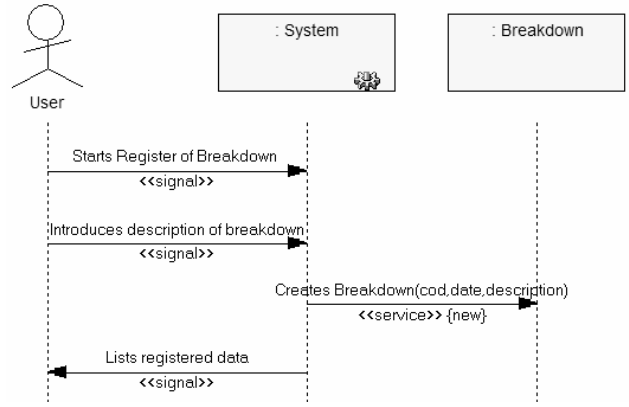

Figure 10. Registering Breakdowns

As shown in Figure 10, the scenario begins when the user starts the registry of a breakdown that occurred in the hospital, which is not considered as a data movement. Then, the user introduces the breakdown description, which is identified as an Entry data movement type (Rule 8). After data introduction, the system creates a new object of the Breakdown class, which is identified as a Write data movement type (Rule 14). As a result, the system shows the registered data, which is identified as an Exit data movement type (Rule 16).

We assigned one numeric value of 1 Cfsu to each data movement identified. Finally, by applying Rule 21, we obtained 3 Cfsu as the functional size of this scenario.

5. Conclusions and Future Work

The construction of the software model is an essential requisite for the obtaining of functional size. The quality of this model will depend on various factors, such as the quality of the functional specification and the completeness and reliability of the mapping rules.

The elaboration of the COSMIC-FFP metamodel and the elaboration of the activities diagram allowed verification of the completeness of the mapping rules, which were defined in [5]. The trigger event and layer were the only concepts that could not be represented explicitly by any rule.

An analysis on reliability in the identification of data movements was carried out. The results show that the entry data movement type is less reproducible that other data movement types. The ambiguity in the identification of data groups involved in the signal messages could be a possible cause of this variability. To reduce this ambiguity, we identified specific guidelines for the identification of the entry data movement type.

In terms of future work, we plan to carry out an experimental study on the reproducibility of RmFFP taking into account these guidelines.

Acknowledgements

This research is part of the DESTINO project (ref. TIN2004-03534) supported by the Ministry of Science and Technology of Spain.

References

[1] ISO/IEC 14143-1- Information Technology - Software Measurement-Functional Size Measurement. Part 1: Definition of Concepts, 1998.

[2] Albrecht A. J., Measuring application development productivity. In IBM Application Development Symposium, 1979, pp. 83-92.

[3] UKSMA, MKII Function Point Analysis Counting Practices Manual. Version 1.3.1, United Kingdom Software Metrics Association 1998.

[4] NESMA, 1997. Definitions and Counting Guidelines for the Application of Function Point Analysis.

[5] ISO, ISO/IEC 19762 Software Engineering, COSMIC-FFP, TO functional size measurement method, International Standards Organization, 2002.

[6] ISO, 1993. International Vocabulary of Basic and General Terms in Metrology, International Organization for Standardization, Switzerland.

[7] E. Insfrán, O. Pastor and R. Wieringa, Requirements Conceptual Engineering-Based Modelling. Journal Requirements Engineering, Springer-Verlag, 2002, 7(2): 61-72.

[8] N. Condori-Fernández, S. Abrahão, O. Pastor, Towards to Functional Size Measure for Object-Oriented Systems from Requirements Specifications. IEEE Quality Software International Conference 2004, Braunschweig, Germany.

[9] IFPUG, Function Point Counting Practices Manual, Release 4.1, International Function Point Users Group, Westerville, Ohio, USES 1999.

[10] Bévo V., Lévesque G., and Open A. UML Notation for Functional Size Measurement Method. In Proc. 9th International Workshop on Software Measurement,Canada, September 8-10, 1999, Pp. 230-242.

[11] Jenner M.S. COSMIC-FFP and UML: Estimation of the Size of to System Specified in UML-Problems of Granularity. In Proc. Fourth European Conf. Soft. Measurement and ICTWith-trol, Germany,May 2001, pp. 173-184.

[12] Habela P., Glowacki E., Serafinski T., Adapting Use Marry Model for COSMIC-FFP based Measurement, in the 15th International Workshop on Software Measurement,Montreal, Canada,Shaker-Verlag, 2005.

[13] Poels G. Functional Size Measurement of Multi-Layer Conceptual Object-Oriented Models. In Proc. 9th International Object-Oriented Information Systems Conference,Geneva, Switzerland, September 2-5, 2003, Pp. 334-345.

[14] 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, 2005, 47(3) : 151-166.

[15] Nagano S., Ajisaka T., Functional metrics using COSMIC-FFP for object-oriented real-time systems. In Proc. 13th International Workshop on Software Measurement, Montreal, Canada, September 23-25, 2003.

[16] Azzouz S., they Open A. To Proposed Measurement Mention in the Rational Unified Process and its Implementation with ISO 19761: COSMIC-FFP. In Software Measurement European Forum,Rome, Italy,2004.

[17] N. Condori-Fernández, S. Abrahão, O. Pastor. The Problem of the data movement duplicity in a functional measurement procedure. Workshop Latin American of Engineering of Requirements and Environments Software, Buenos Aires, Argentina, 2006.

[18] Jacquet J. P. and Abran A., 1997. From Software Metrics to Software Measurement Methods: To Process Model, 3rd Int. Standard Symposium and Forum on Software Engineering Standards Walnut Creek.

[19] O. Pastor, J. Gomez, E. Insfran, and V. Pelechano: The OO-Method approach for information systems modeling: from conceptual object-oriented modeling to automated programming. Information Systems 26 (2001) 507-534.

[20] I. Diaz, L. Dark, O. Pastor, A. Matteo. Interaction Transformation Patterns Based on Semantic Roles, International Conference on Applications of Natural Language to Information Systems, Alicante-Spain, 2005.

[21] COSMIC-FFP Measurement Manual version 2.2, Common Software Measurement International Consortium, January 2003.

[22] ISO, "ISO/IEC 14143-3 - Information technology - Software measurement -Functional size measurement-Part 3: Verification of functional size measurement methods", 2003.

[23] Kemerer C. F. Reliability of Function Points Measurement: A Field Experiment. Communica-tions of the ACM, February 1993, 36(2): 85-97.