A FUNCTIONAL SIZE MEASUREMENT METHOD FOR EVENT-BASED OBJECT-ORIENTED ENTERPRISE MODELS

Geert Poels
VLEKHO Business School, Koningsstraat 336, 1030 Brussel, Belgium
Email: gpoels@vlekho.wenk.be

Keywords: metrics, software sizing, function points, object-orientation, layered architectures, events, enterprise models

Abstract: The effective management of IS-related processes requires measuring the functional size of information systems. Functional size measurement is usually performed using the Function Points Analysis method. Earlier attempts to apply Function Point counting rules to object-oriented systems met with serious problems because the implicit model of functional user requirements in Function Points Analysis is hard to reconcile with the object-oriented paradigm. The emergence of a new generation of functional size measurement methods has changed this picture. The main implementation of this generation, COSMIC Full Function Points, explicitly defines a generic model of functional user requirements onto which artifacts belonging to any IS specification or engineering methodology can be mapped. In this paper we present specific COSMIC-FFP mapping rules for methodologies that take an event-based approach to information system engineering. In particular we show that the event-oriented nature of the COSMIC-FFP measurement rules provides for a natural mapping of concepts. To illustrate the mapping rules we use MERODE, a formal event-based object-oriented methodology for systems development in information processing intensive domains. The mapping rules presented are confined to the enterprise layer in a MERODE IS architecture.

1. INTRODUCTION

To effectively manage the process of IS specification, development, and implementation, it is necessary to know the functional size of the required information system.

According to the ISO standard for functional size measurement, the functional size of an information system is captured by the user's perspective on the functional IS requirements (ISO, 1998). This perspective relates to the subset of user requirements that focus on what the system must do to fulfill the user's information needs, without considering how this will be accomplished. The functional user requirements therefore exclude all technical, quality, and performance requirements. Moreover, functional size measurement must be independent of the technical IS development and implementation decisions that are taken during the IS lifecycle, providing an 'objective' basis of comparison and assuring the timeliness of the size measurement.

Functional size measurement has traditionally been performed using Function Points Analysis (FPA) (Albrecht et al., 1983), a method for data-rich and information processing intensive systems that identifies transaction functions and logical data files based on the requirements specifications. These base functional components are subsequently quantified using a fixed schema of weights that reflects the amount of data stored in a file or manipulated by a function.

The FPA view of functional size reflects contemporary IS engineering methodology and practices based on structured principles and functional decomposition. This function-oriented view is not compatible with modern approaches towards the development of enterprise information systems that use object-oriented (OO) specifications and layered IS architectures.

In OO information systems, data and the procedures operating on these data are encapsulated in a same unit of software called the object. There is no natural mapping between the concept of object and the FPA notion of function. Dismembering object classes into data (i.e. files) and procedures (i.e. transaction functions) is an option, but poses a granularity problem. The scope of the average FPA function in non-OO software is a magnitude of order larger than the scope of a typical class method.

Another problem with FPA is that its implicit model of the system to be sized does not accommodate the concept of layered architecture, which is used to distinguish different kinds of IS components based on the type of functionality they offer to the user. The method does therefore not consider the exchange of data between system layers, positioned at different levels of functional abstraction. As a consequence, it is hard to apply the FPA measurement rules to a layer that does not directly exchange data with the system's users.

These and other problems have motivated a group of experts, called the COmmon Software Measurement International Consortium (COSMIC), to work on the principles of a next generation of functional size measurement methods. The main result of this work is COSMIC Full Function Points (Abran et al., 2001).

The COSMIC-FFP functional size measurement method proposes measurement rules for an abstract model of functional user requirements. This model is based on a base functional component typology that no longer requires data to be separated from the procedures operating on these data. Moreover, the model incorporates the notion of layers in requirements specifications.

The independence of specific IS development methodologies realized by COSMIC-FFP is due to the abstractness of its underlying model. The cost of this generality is the absence of detailed measurement rules for specific requirements specification artifacts. There is currently a need for mapping the constructs used by current IS development methodologies to COSMIC-FFP concepts.

In this paper we provide a set of rules to apply COSMIC-FFP to a specific class of OO methodologies for developing enterprise information systems, i.e. the event-based approach. Event-based methodologies have in common that they assume object behavior to be triggered by the occurrence of real-world or system events, for which explicit modeling constructs are offered. We show that the presence of 'triggering events' in COSMIC-FFP provides for a natural mapping of concepts. To substantiate and exemplify this mapping we use MERODE (Snoeck et al., 1999), a formal event-based, and (partly) UML notation based, methodology that is used in the business administration domain. The scope of the mapping presented in this paper is limited to the enterprise model, i.e. the layer in a MERODE system architecture that captures the essential business requirements that the system must fulfill.

This paper is organized as follows. In section 2 we review the main principles of event-based OO enterprise modeling and illustrate these principles using MERODE. In section 3 we briefly describe the COSMIC-FFP method. Sections 4 and 5 present specific COSMIC-FFP rules for MERODE event-based OO enterprise models, respectively focusing on model mapping rules and function point counting rules. Finally, section 6 discusses related work and section 7 presents conclusions.

2. EVENT-BASED OBJECT-ORIENTED ENTERPRISE MODELING

MERODE prescribes a layered IS architecture that partitions an object model following the principle of model-driven development. According to this principle, different kinds of system specifications are distinguished based on their expected change rates. The enterprise model contains the set of specifications that stem from essential business requirements, reflecting the relevant domain knowledge for running a business. The functionality model consists of the specifications that originate from the work organization (i.e. the input services) and the end user's or organization's information needs (i.e. the output services). Finally, the user interface model specifies the facilities to trigger, interrupt, and resume the execution of the input and output services. The most stable specifications are those of the enterprise model, as they meet requirements that are also valid in the absence of an information system. The most volatile specifications are in the user interface model, which captures workflow, presentation, and dialogue aspects.

Each type of specifications gives rise to a number of object classes, which are organized in layers. The innermost layer contains the enterprise object classes; the middle layer the function object classes; the outermost layer the user interface classes. Objects are only allowed to use the services of objects in the same or a more inner layer. That way, classes are prevented to depend upon less stable classes, ensuring a strict control on the propagation of changes and hence a flexible and stable architecture.

In this paper we are concerned with the functional size measurement of enterprise models. MERODE enterprise models consist of three interrelated sub-models, presenting different views of the same business reality. The first sub-model is a class diagram (for which UML notation can be used) containing a set of object type specifications that are classified according to two mechanisms:

existence dependency[1] and generalization-specialization. The second sub-model is an object-event table specifying which (type of) enterprise objects are involved in which (type of) real-world events. According to this formalism, objects interact when they are involved in the same event. Within the enterprise model, object communication is therefore modeled using the event broadcasting mechanism instead of direct message passing between enterprise objects. The third sub-model is a collection of object life cycle models (formalized by Finite State Machines), expressing the sequence constraints that hold for the participation of enterprise objects in real-world events.

In order to apply functional size measurement we assume that there exists a unified and detailed view of these three sub-models, consisting of a set of object class specifications. Each of these specifications contains an object class name, the object class attributes, and the object class methods. Methods have a specification, consisting of the method signature (including parameters) and the method preconditions, and an implementation consisting of the method body. Each time an enterprise object participates in a real-world event, a method is triggered that may modify the value of the object's attributes. Within class specifications, inherit clauses are used to specify inheritance relationships between object classes as indicated by the generalization-specialization relationships in the class diagram. Existence dependency relationships are specified by means of the abstraction mechanism (i.e. attributes are declared of an object class data type). Sequence constraints and other business constraints are specified within the method preconditions.

3. EVENT-BASED FUNCTIONAL SIZE MEASUREMENT

The COSMIC-FFP method can be applied to a system as a whole, or to any layer within a system that is obtained by an architectural decomposition based on a functional view. Once the scope of measurement is established and the system or layer boundaries are defined, the COSMIC-FFP method proceeds through two stages: a mapping phase and a measurement phase.

[1] The principle of existence dependency is the most distinctive feature of MERODE. For more information on this topic we refer to (Snoeck et al., 1999).

3.1 The Mapping Phase

During the mapping phase functional processes and data groups are identified within the system or system layer that must be sized. These are the only two types of constructs that are used within the system model assumed by COSMIC-FFP.

A functional process is seen as a unique set of data exchanges across the boundaries of the system (or system layer), that implement a cohesive and logically indivisible set of functional user requirements. For any system or system layer, two boundaries are considered: a front-end boundary and a back-end boundary. The front-end boundary separates the system or system layer from its user, which in case of a layer can be another system layer. The back-end boundary separates the system or system layer from data storage, which in case of a layer can be realized through invoking the services of another system layer. Data exchanges across system or layer boundaries can take four forms: entry, exit, read, and write. Fig. 1 illustrates these four types of data exchange for the software component of an information system. In the front-end direction data is exchanged with the user across the I/O boundary. In the back-end direction data is exchanged with the storage devices.

As a practical guideline to delimit a cohesive and logically indivisible set of functional user requirements, the COSMIC-FFP measurement manual states that a functional process is triggered by an event occurring outside the system or layer boundary and is complete when it has executed all that is required to be done in response to the triggering event.

Apart from functional processes, all data groups referenced by these functional processes need to be identified. A data group is defined as a distinct, non-empty, non-ordered, and non-redundant set of data attributes, where each included data attribute describes a complementary aspect of the same object of interest.

3.2 The Measurement Phase

Within each functional process, all COSMIC-FFP sub-processes are identified. In the current version of COSMIC-FFP, these sub-processes are exactly the same as the data exchanges occurring during the execution of a functional process. There are four COSMIC-FFP sub-process types:

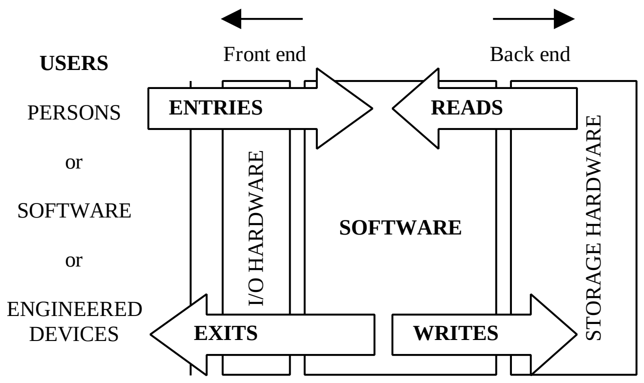


Figure 1: Generic exchange of data through software from a functional perspective

1. An ENTRY (E) is a movement of one or more data attributes found in one data group from the user's side of the boundary to the inside of the boundary.
2. An EXIT (X) is a movement of one or more data attributes found in one data group from inside the boundary to the user side of the boundary.
3. A READ (R) is a movement of one or more data attributes found in one data group from storage to the functional process to which the read sub-process belongs.
4. A WRITE (W) is a movement of one or more data attributes found in one data group from the functional process to which the write sub-process belongs to storage.

The COSMIC-FFP sub-process types are the base functional component types used by the COSMIC-FFP method. They determine the particular point of view of functional size taken by this method.

The function point counting procedure of COSMIC-FFP is trivial once sub-processes are identified. Each instance of a sub-process type found in the functional processes has a functional size equal to 1 Cfsu (Cosmic functional size unit), which is the measurement standard used by COSMIC-FFP. The sum of these values is the functional size of the system or system layer to be sized. Given the choice of measurement standard, this sum is equal to the count of sub-processes (or data exchanges).

4. FUNCTIONAL SIZE MEASUREMENT MAPPING RULES

The COSMIC-FFP mapping phase can be subdivided in two sub-phases. The first sub-phase results in an instance of the COSMIC-FFP context model, where the scope of the functional size measurement exercise is determined by identifying system layers and boundaries. The second sub-phase constructs an instance of the abstract COSMIC-FFP model that determines the functional structure (i.e. functional processes and data groups) of the system or system layer within the scope of functional size measurement.

4.1 COSMIC-FFP Context Model for MERODE Enterprise Models

According to the COSMIC-FFP measurement manual, the identification of system layers is facilitated by the use of a layered IS architecture on condition that it provides a functional view of the various system components. The MERODE architecture provides such a view, as it distinguishes between business functionality, information system functionality, and user interface functionality. The manual further stipulates that if a specific architectural paradigm is used, then an equivalence should be established between specific architectural objects and the concept of layers in COSMIC-FFP. The principle of model-driven development allows distinguishing three such layers in the COSMIC-FFP context model for MERODE: the enterprise layer, the functionality layer, and the user interface layer.

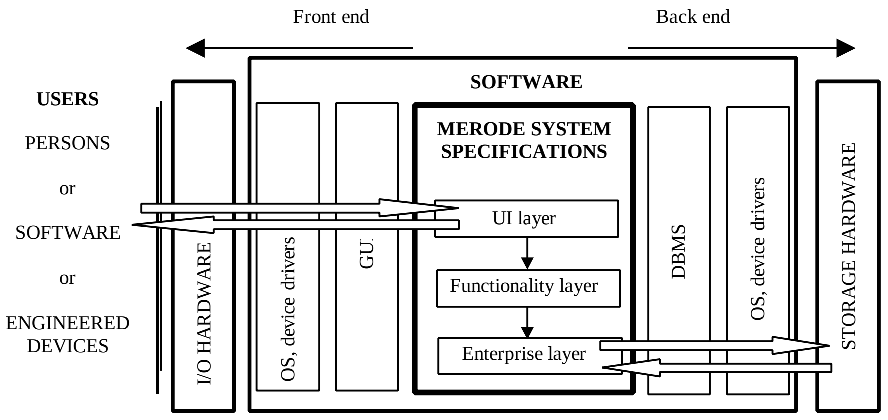


Figure 2: COSMIC-FFP context model for MERODE

The boundaries between these three layers are clearly defined in MERODE (Fig. 2). The user interface classes use the services of the function object classes, which in turn use the services of the enterprise object classes. Therefore, the user interface layer is a client to the functionality layer, which is in turn a client to the enterprise layer. The user interface layer has a front-end boundary with the user, via some intermediate layers outside the scope of MERODE (e.g. I/O devices, device drivers, OS, GUI). The enterprise layer has a back-end boundary with storage devices, again via a number of subordinate layers like DBMS, OS, and device drivers.

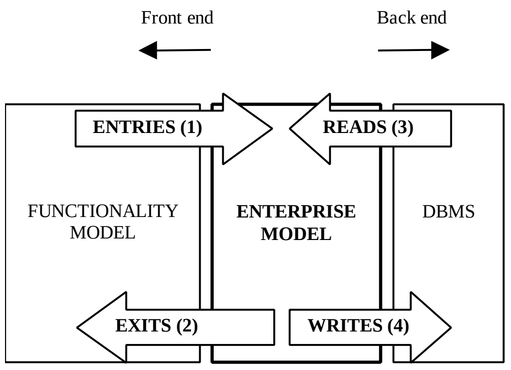
As this paper is only concerned with functional size measurement for MERODE enterprise models, we limit the scope of functional size measurement to the enterprise layer of the software application. It should be understood that the enterprise object classes specify only a subset of the functional user requirements. Though it seems that these business requirements do not directly deliver information system functionality (i.e. information products) to the system users, which is the aspect of size considered by COSMIC-FFP, they do represent part of the user practices and procedures that the system must perform to fulfill the user's needs. We therefore consider the business functionality specified in the enterprise object classes as part of the functionality delivered by the system to the users (via the functionality and user interface layers). The functional size of the enterprise layer contributes to the functional size of the entire system. As opposed to other functional size measurement methods, COSMIC-FFP allows measuring the functional size of separate layers.

Fig. 3 details the COSMIC-FFP context model for MERODE by focusing on the data exchanges across the boundaries of the enterprise layer.

In the front-end direction, function objects communicate with enterprise objects by means of messages and status inspections. When an event occurs in the real world, a function object sends an event message to all enterprise objects involved. For each of these enterprise objects an event method is triggered by the event message. If this method has parameters, then the event message carries the required data. With this type of object communication, there is no exchange of data from the enterprise layer to the functionality layer, as event methods have no return type. Status inspections[2] allow function objects to inspect the value of enterprise object attributes. Hence, they carry data from the enterprise layer to the functionality layer.

In the back-end direction, it is assumed that the services of a DBMS are used to store and retrieve the enterprise object state vectors, which contain the object attribute values.

[2] Status inspections can be implemented by sending messages to trigger 'accessor' methods, having return types. Such methods are however not specified in MERODE enterprise classes, as they are considered an implementation issue.



Legend:

(1) Entry of data through event message parameters
(2) Exit of data through status inspections
(3) Retrieval of object state vectors
(4) Storage of object state vectors

Figure 3: detailed COSMIC-FFP context model for MERODE enterprise models

4.2 Abstract COSMIC-FFP Model for MERODE Enterprise Models

The abstract COSMIC-FFP model contains all elements of the functional user requirements within the scope of the context model (in our case limited to the enterprise layer), that are considered as relevant for a COSMIC-FFP functional size measurement exercise.

The definition of a functional process in the measurement manual leads to an unambiguous interpretation of this concept in the context of MERODE enterprise models. This definition states that "A functional process is a unique set of data movements […] implementing a cohesive and logically indivisible set of [functional user requirements]. It is triggered directly, or indirectly via an 'actor', by an Event (-type) and is complete when it has executed all that is required to be done in response to the triggering Event (-type)" (Abran et al., 2001, p. 26). Furthermore, this triggering event must occur outside the boundary of the measured system or system layer.

In case of MERODE enterprise models, business functionality is triggered by events occurring in the real world. Input function objects qualify as actors that transmit real-world events to the enterprise model via event messages. These messages trigger a set of event methods, leading to the execution of the corresponding method bodies. Adhering to the above definition, a MERODE enterprise model functional process can therefore be defined as the collection of data exchanges that are contained within the set of event methods triggered by a real-world event (type). This definition is operationalized as Mapping Rule 1.

Mapping Rule 1: For each event type in the object-event table, identify a functional process. Include in this functional process all event methods specified in the enterprise object classes, occurrences of which are involved in occurrences of the event type.

According to their definition (cf. section 3.1) data groups in COSMIC-FFP correspond to the sets of attributes in the enterprise object classes.

Mapping Rule 2: For each object type in the object-event table, identify a data group. This data group is the set of attributes found in the enterprise object class that specifies the data, functionality, and behavior of the object type.

5. FUNCTIONAL SIZE MEASUREMENT COUNTING RULES

A COSMIC-FFP sub-process is defined as "a data [exchange] occurring during the execution of a functional process" (Abran et al., 2001, p. 34). Data exchange sub-processes move data contained in exactly one data group. The data is moved across the boundary of the system or system layer.

The following kinds of data exchange occur during the execution of a MERODE enterprise model functional process:

1. The functional process is triggered by event messages, which may carry data from the functionality layer to the enterprise layer.
2. During the execution of the functional process, the values of enterprise object attributes are updated and stored in the enterprise object state vectors.

3. During the execution of the functional process, values of enterprise object attributes may be retrieved from the enterprise object state vectors.

The first type of data exchange relates to entry sub-processes in COSMIC-FFP. Data is moved from the functionality layer's side of the boundary to the side of the enterprise layer. As each ENTRY moves data from only one data group, the triggering of a functional process by a real-world event may result in the execution of more than one entry sub-process. Each event method that is triggered may require parameters referring to the enterprise object attributes. Hence, the potential number of entry sub-processes is equal to the number of object types involved in the event type. This reasoning is formalized in Counting Rule 1.

Counting Rule 1: For each functional process, an ENTRY functional sub-process is counted for each of the event methods it includes, on condition that the method signature has at least one parameter.

The second type of data exchange relates to write sub-processes. Each time that the execution of an event method body involves initializing or modifying the value of at least one enterprise object attribute, a WRITE occurs. Conceptually, the data is moved from the enterprise layer's side of the boundary to the DBMS side. As during the execution of a functional process more than one event method can be triggered and all these triggered methods relate to different data groups (i.e. enterprise objects), more than one write sub-process may be identified.

Counting Rule 2: For each functional process, a WRITE functional sub-process is counted for each of the event methods it includes, on condition that the method body updates the value of at least one enterprise object attribute.

The third type of data exchange relates to read sub-processes. The processing within an event method body may require a READ to supply the value of one or more enterprise object attributes. Even before a method body is executed, it might be necessary to get the value of enterprise object attributes when checking method preconditions. This access to the enterprise object attributes can conceptually be described as a movement of data from the DBMS side of the boundary to the enterprise layer's side. Again, more than one read sub-process can be involved in the execution of a MERODE enterprise model functional process.

Counting Rule 3: For each functional process, a READ functional sub-process is counted for each of the event methods it includes, on condition that the method body or preconditions retrieve the value of at least one enterprise object attribute.

There is no equivalent to the COSMIC-FFP EXIT functional sub-process in MERODE enterprise models. Although, conceptually, status inspections bring data lying inside the enterprise layer within reach of the functionality layer (cf. Fig. 3), these data movements are not within the realm of the MERODE enterprise model functional processes. Exit data exchanges are part of the MERODE functionality model functional processes, which implement information system functionality required by users. As such they are not documented in the MERODE enterprise model.

To summarize, the functional size of a MERODE enterprise model is equal to the number of ENTRY, WRITE, and READ sub-processes in its functional processes.

6. RELATED WORK

To 'solve' the OO-FPA mapping problem, many FPA variants have been proposed (Rains, 1991; Whitmire, 1993; Thomson et al., 1994; Hastings, 1995; Zhao et al., 1995; Sneed, 1996; Caldiera et al., 1998; Fetcke et al., 1998; Uemura et al., 1999; Pastor et al., 2001; Ram et al., 2001). None of these proposals has succeeded in becoming a widely accepted functional size measurement method for OO systems.

The feasibility of mapping OO concepts onto the abstract COSMIC-FFP functional user requirements model is evidenced by recent work in the field (Bévo et al., 1999; Jenner, 2001; Diab et al., 2001).

Both Bévo et al. (1999) and Jenner (2001) provide a mapping for UML-based specifications. Bévo et al. map constructs used in class diagrams and use case diagrams onto the abstract COSMIC-FFP model. The use case diagram is used to identify the system (i.e. the collection of use cases modeled), its boundary (i.e. the system boundary specified in the use case diagram), and its users (i.e. the actors in the use case diagram). An equivalence to the layer concept in COSMIC-FFP is not established. Bévo et al. further contend that UML classes correspond to COSMIC-FFP data groups and that use cases correspond to COSMIC-FFP functional processes. A scenario, i.e. a specific sequence of actions illustrating a use case instance, is considered as a sequence of COSMIC-FFP sub-processes.

As shown by Jenner, Bévo et al. confuse in their examples the concept of scenario with that of detailed use case. They provide for a same system specification widely ranging size 'estimates', depending on the level of detail in the use case diagram. As argued by Jenner, the fact that a system specification may look very different according to who produces it (i.e. according to the level of granularity with which use cases are described), suggests that a mapping of use cases onto COSMIC-FFP functional processes may not be appropriate. As an alternative, Jenner proposes to look at the sequence diagram for each use case and regard each use case as a sequence of COSMIC-FFP functional processes. Each functional process is triggered by something done or requested by an actor, as documented in the sequence diagram. According to Jenner, sequence diagrams are specified at a level of detail, where the granularity problem observed when specifying use case diagrams, does not occur. Jenner further uses the concept of "swimlanes" in UML sequence diagrams to establish an equivalence with the layer concept in COSMIC-FFP.

The different interpretations of functional processes in UML by Bévo et al. and Jenner stem from the fact that the concept of a 'triggering event' is not explicitly present in UML. Diab et al. (2001) provide a mapping for ROOM (Real-time Object-Oriented Modeling) specifications, which correspond more closely to the event-based approach to system specification described in this paper. Diab et al. identify functional processes based on the transitions in statechart specifications. These transitions are triggered by messages, which can be considered as external events. The interpretation of COSMIC-FFP sub-processes is largely the same as in this paper. It must be noted, however, that ROOM specifications are not organized using layered architecture principles. The application of functional size measurement to a specific layer within the IS architecture distinguishes our work from (Diab et al., 2001).

7. CONCLUSIONS

This paper presented five rules to map the constructs and artifacts used in MERODE enterprise modeling onto the concepts, definitions, principles, and rules of the COSMIC-FFP method. COSMIC-FFP is the principal implementation of a new generation of functional size measurement methods, which is meant to improve the first generation of Function Points based methods (i.e. FPA and variants).

The main distinguishing feature of COSMIC-FFP is the explicit definition of a generic, abstract model of the functional user requirements, onto which the operational artifacts of the IS development methodology that is adopted, are mapped before measurement takes place. It is this feature that makes a functional size measurement method not only independent of technology and implementation choices, as with FPA, but above all really free from the development methodology. The paradigm clash between OO methodologies and first generation Function Points based methods is therefore not observed when mapping the OO specifications in a MERODE enterprise model onto the abstract COSMIC-FFP functional user requirements model.

Two other features of COSMIC-FFP provide for a natural mapping of concepts. First, the notion of layers in COSMIC-FFP allows determining the scope of functional size measurement in the context of a layered system (or software) architecture, and thus helps conceiving specific functional size measurement procedures for the enterprise layer in the MERODE IS architecture.

Second, the focus on external events as process triggers in both COSMIC-FFP and MERODE, helps determining the appropriate level of granularity of the mapping rules. It is especially this event-orientation in COSMIC-FFP that goes well with the event-driven nature of MERODE and the event-based approach to requirements specification in general.

In our future work we will investigate the applicability of Early & Quick COSMIC-FFP (Meli et al., 2000; Santillo, 2001), a functional size measurement method currently being developed for functional user requirements at a higher level of abstraction than usually assumed in COSMIC-FFP. This would allow measuring the functional size of a MERODE enterprise model using partial, early available, information, such as an object-event table. One such high-level functional size measure, the Level of Object-Event Interaction (LOEI), which is basically a count of the object type - event type interactions in the object-event table, has recently been proposed (Poels et al., 2001). We will investigate whether the LOEI measure complies with the Early & Quick COSMIC-FFP method.

REFERENCES

Abran, A., Desharnais, J.-M., Oligny, S., St-Pierre, D., Symons, C. (2001). COSMIC-FFP Measurement Manual, Version 2.1. The Common Software Measurement International Consortium

Albrecht, A.J., Gaffney, J. (1983). Software function, source lines of code and development effort prediction. IEEE Transactions on Software Engineering 9(6) 639-648

Bévo, V., Lévesque, G., Abran, A. (1999). Application de la méthode FFP à partir d'une spécification selon la notation UML: compte rendu des premiers essais d'application et questions. Proceedings of the 9th International Workshop on Software Measurement, Lac Supérieur, Canada, 230-242

Caldiera, G., Antoniol, G., Fiutem, R., Lokan, C. (1998). Definition and Experimental Evaluation of Function Points for Object-Oriented Systems. Proceedings of the 5th International Symposium on Software Metrics, Bethesda, Maryland, USA, 167-178

Diab, H., Frappier, M., St-Denis, R. (2001). A Formal Definition of COSMIC-FFP for Automated Measurement of ROOM Specifications. Proceedings of the 4th European Conference on Software Measurement and ICT Control, Heideberg, 185-196

Fetcke, T., Abran, A., Nguyen, H. (1998). Function point analysis for the OO-Jacobson method: a mapping approach. Proceedings of the 1st European Software Measurement Conference, Antwerp, 395-410

Hastings, T. (1995). Adapting Function Points to contemporary software systems: A review of proposals. Research report, Department of Software Development, Monash University, Australia

ISO (1998). ISO/IEC 14143-1998 - Software Engineering - Software Measurement - Definition of Functional Size Measurement. The International Organization for Standardization

Jenner, S.M. (2001). COSMIC-FFP 2.0 and UML: Estimation of the Size of a System Specified in UML - Problems of Granularity. Proceedings of the 4th European Conference on Software Measurement and ICT Control, Heideberg, 173-184

Meli, R., Abran, A., Ho, V.T., Oligny, S. (2000). On the Applicability of COSMIC-FFP for Measuring Software Throughout its Life Cycle. Proceedings of the 11th European Software Control and Metrics Conference, Munich

Pastor, O., Abrahao, S.M., Molina, J.C., Torres, I. (2001). A FPA-like Measure for Object-Oriented Systems from Conceptual Models. Proceedings of the 11th International Workshop on Software Measurement, Montréal

Poels, G., Dedene, G. (2001). Measuring event-based object-oriented conceptual models. L'Objet: Software, Databases, Networks 7(4) 497-514

Rains, E. (1991). Function Points in an ADA Object-Oriented Design. OOPS Messenger 2(4) 23-25

Ram, D.J., Raju, S.V.G.K. (2001). Estimating Relative Size when Alternative Designs Exist. Proceedings of the 5th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, Budapest, 35-44

Santillo, S. (2001). Early & Quick COSMIC-FFP Analysis Using Analytic Hierarchy Process. Proceedings of the 10th International Workshop on Software Measurement, Berlin, 2000. Lecture Notes in Computer Science, Vol 2006, Springer-Verlag, Berlin, 147-160

Sneed, H. (1996). Estimating the Development Costs of Object-Oriented Software. Proceedings of the 7th European Software Control and Metrics Conference, Wilmslow, UK, 135-152

Snoeck, M., Dedene, G., Verhelst, M., Depuydt, A.-M. (1999). Object-Oriented Enterprise Modelling with MERODE. Leuven University Press

Thomson, N., Johnson, R., MacLeod, R., Miller, G., Hansen, T. (1994). Project Estimation Using an Adaptation of Function Points and Use Cases for OO Projects. Proceedings of the OOPSLA'94 Workshop on Pragmatic and Theoretical Directions in Object-Oriented Software Metrics

Uemura, T., Kusumoto, S., Inoue, K. (1999). Function Point Measurement Tool for UML Design Specifications. Proceedings of the 6th International Symposium on Software Metrics, Boca Raton, Florida, USA, 62-69

Whitmire, S.A. (1993). Applying Function Points to Object Oriented Software Models. In: Software Engineering Productivity Handbook. McGraw-Hill, New York, 229-244

Zhao, H., Stockman, T. (1995). Software Sizing for OO software development - Object Function Point Analysis. Proceedings of the 2nd GSE International Conference on Information Technology and Management, Berlin