A COSMIC Measurement Procedure for Sizing Web Applications Developed using the OO-H Method

S. Abrahão
Department of Computer Science and Computation
Camino de Vera, s/n, 46022
Valencia, Spain
sabrahao@dsic.upv.es.it

L. De Marco, F. Ferrucci, C. Gravino, F. Sarro
Dipartimento di Matematica e Informatica
University of Salerno
Via Ponte Don Melillo, I-84084 Fisciano (SA), Italy
{fferrucci, gravino, fsarro}@unisa.it

ABSTRACT
In the last years some model-driven development approaches have been proposed to automatically generate Web applications starting from conceptual models. Among them, the Object-Oriented Hypermedia method (OO-H) has been successfully used in the development of industrial Web applications. This method is based on two conceptual models, the UML class diagram to represent content and behavioral requirements, and the Navigation Access diagram to represent navigational requirements. The need of functional sizing this kind of Web applications motivated us to define a COSMIC measurement procedure to be applied on the OO-H conceptual models. In this paper, we present the mapping rules that have been conceived together with an example of their application.

Categories and Subject Descriptors
D.2 [Software Engineering]: General [2.0]; Metrics [D.2.8]

General Terms
Measurement, Design, Experimentation.

Keywords
Model-driven development, Web applications, OO-H, Functional Size Measurement, COSMIC.

1. INTRODUCTION
Model-driven development processes aim to develop software at a higher level of abstraction exploiting models and model transformations. In the last years, some model-driven development approaches have been proposed to automatically generate Web applications starting from conceptual models (e.g., OO-HDM [23], WebML [7], OO-H [12], W2000 [5]). These approaches support the construction of different views (i.e., models) of a Web application comprising at least a structural model, a navigation model and a presentation model. Some of these approaches also provide support for model transformations and code generation. A Web development process that follows a model-driven development approach basically transforms models that are independent from implementation details (Platform-Independent Models - PIM) into other models that contain specific aspects from a concrete platform (Platform-Specific Models - PSM). PSMs can then be compiled to automatically generate the Web application source code (Code Model - CM). The great part of these approaches have been developed in academic contexts, however they have also been used in several industrial settings.

Since these approaches are being applied in practice it is important to provide guidelines for practitioners on how the size of this type of Web applications can properly be measured. To address this issue, some size measurement procedures have recently been proposed [1][3][6][11]. However, they are based on the first generation of functional size measurement methods. In the last years some researchers have applied the COSMIC method to Web applications and some case studies have been carried out to evaluate its use for Web development effort estimation [9][10][20][22], reporting encouraging results. Nevertheless, no of these studies was concerned with Web applications developed using a model-driven approach.

In this work, we present the design of a COSMIC measurement procedure for sizing Web applications developed using the Object-Oriented Hypermedia method (OO-H) [12]. This method is based on two conceptual models, the UML class diagram to represent content and behavioral requirements, and the Navigation Access diagram to represent navigational requirements.

The design and application of the COSMIC measurement procedure is presented by using a process model for software measurement (see Figure 1) [15]. According to this process, a measurement method is designed, that is, the concept to be measured is defined and the measurement rules are devised. Then, the measurement method is applied. The results of the method are then presented and verified. Such verification includes determining whether the value that is produced is the result of a correct application and interpretation of the measurement rules. Finally, the results are used to build different types of models (e.g., productivity analysis models, effort estimation models). In this paper, we focus on the first two activities of the measurement process. The validation of the measurement results will be subject of future work.
This paper is organized as follows. Section 2 presents an overview of the OO-H method. Section 3 presents the design of the COSMIC measurement procedure developed for sizing OO-H Web applications. Section 4 describes the application of the measurement procedure to a real case study. Section 5 discusses related work. Finally, section 6 presents the conclusions and future work.

2. THE OO-H METHOD

The OO-H (Object-Oriented Hypermedia) method [12] provides designers with the semantics and notation for developing Web applications. The method includes: a design process, a pattern catalog, a Navigation Access Diagram (NAD), and an Abstract Presentation Diagram (APD). The VisualWADE\textsuperscript{1} tool automates the entire OO-H development process.

2.1 The OO-H Development Process

The OO-H development process starts from the domain information structure that is captured in an UML-compliant class diagram. From there, we can create different NAD instances that represent the navigation and interaction requirements. A single NAD is a partial view from the UML class diagram. It structures the navigational view of the Web application for a specific kind of user. It also defines constraints about navigation and contents, which should be showed to the user. For this purpose, OO-H uses the Object Constraint Language (OCL).

By using the UML class diagram and the NADs as input an APD can be automatically obtained. The APD contains the definition of abstract Web pages (pages that are not attached to any specific Web language) based on a set of XML templates that capture the relevant presentation properties of the Web application under construction. The APD can be refined by applying a pattern catalog. This catalog contains a set of constructs that effectively solve consistency problems. Once the APD is refined, we can automatically generate a front-end Web application (static or dynamic) for the desired target platform, such as HTML, WML, active server pages (ASPs), and JavaServer pages (JSPs). Figure 2 presents an overview of the OO-H development process. For more information about the method readers are referred to [12].

2.2 The OO-H Metamodel

Figure 3 shows an excerpt of the OO-H Metamodel [12] including the main concepts used to represent navigational properties. The navigation model of a Web application is defined by a set of navigational maps. Each navigational map structures the navigational view of the system for a specific kind of user and has a unique entry point that indicates the starting point of the navigation process.

A navigational map is made up of a set of navigational elements that can be specialized as navigational nodes and/or navigational links.

\textsuperscript{1} http://www.visualwade.com/

Figure 1. Measurement process model steps [15]

Figure 2. The OO-H development process

Figure 3. An excerpt of the OO-H metamodel

Navigational links define the navigation paths that the user can follow through the user interface. Each navigational link has an origin (org) node and its corresponding destination (dst) node. There are two types of links: internal links (I-Link) to define the navigation path inside a navigational map, and service links (S-Link) to show the services available to the user associated to that navigational map. In addition, as previously mentioned, OO-H allows the use of OCL to define constraints over the class diagram and the NAD. In the NAD, such constraints are expressed by means of filters defined upon links.

3. DESIGN OF THE COSMIC MEASUREMENT PROCEDURE

The COSMIC measurement method involves applying a set of models, principles, rules and processes to the Functional User Requirements (FURs) of a given piece of software. The result is a numerical ‘value of a quantity’ (as defined by ISO) representing
the functional size of the piece of software according to the COSMIC method [8].

The application of the method requires the identification of two models: the context model and the software model [8] (which are reported in Table 2 and Table 3 of Appendix A). The former establishes the bounding of the application from its operating environment, and illustrates the generic functional flow of data attributes from a functional perspective. The flow of data attributes is characterized by two directions, back-end and front-end, and by four distinct types of movements: entries and exits, which allow the exchange of data with user, and reads and writes, which allow the exchange of data with the persistent storage hardware. The software model assumes that two general principles hold for the software to be mapped and measured: 1) software takes input and produces useful output to the users, and 2) software manipulates pieces of information designated as data groups which consist of data attributes. This software model allows us to consider the functional user requirements decomposed in a set of functional processes, where each process is a unique set of sub-processes performing either a data movement or a data manipulation.

The basic idea underlying the COSMIC approach is that, for many kinds of software, the main programming efforts are devoted to handle data movements from/to the software system and the users. Thus, their number can provide a meaningful sight of the system size. Four kinds of data movements are considered: Entry, Exit, Write, Read. The size of the software, expressed in terms of Cosmic Functional Point (CFP) [8], is given by the sum of all data movements (i.e., each data movement is one CFP).

In order to provide a COSMIC measurement procedure for Web applications developed with OO-H (in short OO-HCFP) we took into account the process model for software measurement shown in Figure 1 [15]. In the first step, four activities are suggested for a complete design of a measurement method: definition of the objectives, characterization of the concept to be measured, definition or selection of the metamodel, and definition of the numerical assignment rules.

### 3.1 Definition of the Objectives

The goal of this work is to design a functional size measurement procedure for the purpose of automatically sizing OO-H conceptual models with respect to their functional size from the point of view of the researcher. The context is that this procedure should conform to the COSMIC method version 3.0.1 [8].

### 3.2 Characterization of the Concept to be Measured

The entity to be measured consists of an OO-H conceptual model. It is composed of a UML class diagram and a NAD that capture the structural and the navigational views of a Web application, respectively. These views contain all the elements that contribute to the functional size of a Web application. The attribute to be measured is functional size, which is defined in ISO/IEC 14143-1 [14] as the size of the software derived by quantifying the functional user requirements.

### 3.3 Selection of the Metamodel

Metamodeling is a key concept of the Model Driven Application (MDA) paradigm and is used in Software Engineering to describe the basic abstractions that define the models and their relationships. The metamodel of a Functional Size Measurement (FSM) method provides a precise basis to design the measurement rules that identify and measure these concepts. As OO-HCFP is intended to conform to the COSMIC FSM method [8], Figure 4 illustrates the concepts that must be identified in order to size a software project using the COSMIC method. The relationships among the concepts are also shown. In other words, it represents at higher level the concepts specified by the context model (Table 2) and software models (Table 3) that we have to instantiate to provide the OO-HCFP.

### 3.4 Definition of the Mapping Rules

Table 4 presents the mapping rules we conceived to identify the elements of an OO-H conceptual model that contribute to the functional size of a Web application. They allow us to map each concept of COSMIC (Figure 4) into a concept of OO-H. They have been defined taking into account the COSMIC context model (Table 2) and the software model (Table 3) [8]. In the following we provide a description of these rules and a rationale for them.

First of all, Rule 1 is related to FURs: for an OO-H Web application FURs are represented by the FIM since it contains the UML class diagram and the NADs.

Rules 2.1-2.3 are related to software layer identification. This has been carried out taking into account the architecture of OO-H Web applications that is based on the Model View Control (MVC) model. In particular, the User Interface layer is mapped into the Navigational classes involved into the Navigational Target of the NADs, because they represent the navigational pages which the users interact with (Rule 2.1). The Business Logic layer (see Rule 2.2) is formed by all the I-links starting from the navigational classes involved into the Navigational Target of the NADs, because they represent the invocation of the operations of the functionalities of the system. Moreover, it includes all the S-Links of the NADs because they represent the invocation of some operations of the system (thus lying into the Business Logic layer of the application). The Data layer (see Rule 2.3) is formed by all the classes of the UML class diagram (representing the persistent information of the system).

Rules 3.1 and 3.2 are related to another concept of COSMIC, namely peer component, meant as a partition within one layer into a mutually co-operating set such that each partition fulfills a specific portion of a FUR. Taking into account the layer stratification of an OO-H Web application identified above, a peer component is a Navigational Class of the NADs, because the NADs cover the User Interface and Business Logic layers, or a
class of the UML class diagram because it covers completely the mapping to identify the Data layer.

Rule 4 specifies that we can decide to measure a whole OO-H Web application by taking into account the whole OO-H conceptual model, or a part of it by including in the scope a subset of NADs and a (connected) subset of classes in the UML class diagram.

As for the identification of functional users, let's recall that COSMIC defines a “(type of) user as a sender and/or an intended recipient of data in the FURs of a piece of software” [8]. Thus, we considered as users all the classes with stereotypes «Actor» or «Legacy» in the UML class diagram (Rule 5.1). Moreover, we also took into account the peer components (outside the boundary of the scope) that can have a client/server relationship with the OO-H Web application (Rule 5.2). Finally, we also included every human user that interacts with the system (Rule 5.3).

To identify the COSMIC application boundary (i.e., the border between the software being measured and the external applications or user domain) into an OO-H Web application we can trace an imaginary line in the UML class diagram that puts the «Actor» and «Legacy» classes outside the boundary whereas the other classes (or those included in the scope) are considered inside it. Of course, we also included in the boundary the NADs’ functionalities that involve the classes of the Class Diagram (Rule 6).

Rules 7, 8 and 9 were conceived to map the COSMIC concepts of functional process, functional sub-process, and triggered event. According to the COSMIC Generic software model (see Table 3), a functional process is each functionality that a functional user can execute in the system, with at least two data movements [8]. The concepts of functional processes and sub-processes were mapped into Navigational Targets (Rule 7) and methods of Navigational Classes (Rule 8), respectively. Indeed, a Navigational Target is an element of a NAD that can be considered as a grouping mechanism: it includes the elements that contribute to realize a set of related functional requirements. Moreover, in COSMIC a functional user can trigger an event that causes the start of a functional process, considered as an executable set of data movements [8]. Thus, a triggered event can be mapped in an entry point followed by an R-Link that points to a starting navigation point (i.e., a Collection or a Navigational Class) inside each Navigational Target (Rules 9.1 and 9.2).

COSMIC counting rules focus on sub-processes represented by data movements [8], where each data movement moves a single data group (see Table 3). Furthermore, a data group consists of a unique set of data attributes describing a single object of interest. Thus, we provided three rules (10.1-10.3) to map these two concepts in the OO-H conceptual models. In particular, to identify data groups, we took into account the concepts of the UML class diagram (Rule 10.1) since it represents persistent data managed by the system. First, we identify one data group for each aggregation hierarchy in the UML class diagram since aggregation represents “is part of” relationships. Similarly, we identify one data group for each composition hierarchy since composition is considered a stronger form of aggregation where the whole and the parts have coincident lifetimes. Finally, we identify one data group for class of the UML class diagram that does not participate in an aggregation or composition hierarchy.

As for inheritance hierarchies, we identify one data group for the father class and another data group for the child class if it includes new attributes. An OO-H Web application is also composed of other elements that can be considered objects of interest such as the Web pages of the application that lie on an application server, waiting to be required from a user. In this case all the software modules available on an application server can be considered as data groups, being compliant with COSMIC. In the OO-H models, we can identify the abstract Web pages with the Navigational Classes and Collection inside the Navigational Target of the NADs (Rules 10.2 and 10.3).

Finally, Rules 11.1 and 11.2 specify how to identify data attributes from the classes of the UML class diagram and the NADs.

3.5 Definition of the Measurement Rules

According to COSMIC measurement method the functional size of the software is given by the sum of all data movements. Thus, to assign a quantitative value to an OO-H Web application we defined the set of measurement rules shown in Table 5. These rules allow us to identify Entry, Exit, Read, and Write data movements from OO-H conceptual models. In the following, we provide a description of such rules and a rationale for them.

Rules 12.1-12.7 were conceived to count the Entry data movements. In particular, one Entry is assigned to each R-Link pointing to a Collection (Rule 12.1) or to a Navigational Class (Rule 12.2). Indeed, this kind of link carries the information needed by the system to move a Web page from a Web server to the client user interface and it is triggered by a human user. The data group moved is the Collection element or the Navigational Class element of the Navigational Target involved. Observe that the considered R-Link must have the attribute User Interaction set to manual.

Rule 12.3 states that an Entry data movement can be also identified with an I-Link between two Navigational Classes of the same Navigational Target or between a Navigational Class and a Collection. This kind of link carries information about the Navigational Class that the user wants to navigate. Again, the considered I-Link must have the attribute User Interaction set to manual.

Rules 12.4, 12.5, and 12.6 allow us to count Entry data movements due to the S-Links that produce the parameters for the invocation of methods with the hidden, constant, or immediate mode. Indeed, they are used to represent the interaction with a human user who introduces information, for example, into a text area (immediate), or clicking on a HTML link (hidden) or on a HTML button (constant).

The last rule for counting Entry data movements (i.e., Rule 12.7) considers the OCL filters on the I-Links labeled with a question mark. In this case it is assumed that the interface is waiting for an external input, i.e., the user must introduce the information needed by the operation.

Regarding the counting of Exit data movements, we provided four rules (13.1-13.4). Rule 13.1 identifies one Exit for each I-Link between two Navigational Classes of the same Navigational Target (or between a Navigational Class and a Collection). Indeed, these links carry out from Web server the Web page needed to provide the user with the required functionality (the
Navigational Class destination). Similarly, Rule 13.2 counts one Exit for each R-Link to a Navigational Class or a Collection from the entry point. This link carries out from a Web server the Web page needed by the user to initiate the required functionality.

Following the suggestion of COSMIC User Manual, we also introduced a rule to count an Exit data movement for each Error Message that occurs during the use of the system (Rule 13.3).

Rule 13.4 considers the OCL constraints on the I-Link in order to count Exit data movements for the data retrieved by these links. In particular, when this kind of link is executed together with a Read data movement about the data group interested (see below) an Exit data movement has to be counted to represent the information presented to the user via the Navigational Class (i.e., the target of the considered link).

Rules 14.1-14.4 were related to Read data movements. Rules 14.1 and 14.2 allow us to count one Read for each link that points to a Navigational Class or a Collection inside a Navigational Target (i.e., a Requirements Link - R-Link) and for each I-Link between the same Navigational Targets, respectively. Indeed, these links move data from the Web server to the user in order to provide the Web page required and in both case the presentation of a correct Web page implies a read from the Web server, where these data groups lie.

Rules 14.3 and 14.4 take into account OCL constraints on links, since they implies further Read data movements. In particular, for each I-Link with an OCL constraint that requires further elaboration of information from persistent storage to be presented in the Web page, we count one Read for each distinct data group involved. Similarly, for each S-Link getting an OCL annotation that involves one or more classes different from the source and destination Navigational Classes, we count one Read for each distinct data group involved.

Finally, the Write data movements are identified from the execution of S-Links of the Navigational Classes of the NADs. Indeed, each S-Link starts from an operation and indicates the execution of an operation that lies in the business logic layer.

4. APPLICATION OF THE COSMIC MEASUREMENT PROCEDURE

We present an example of application of the proposed COSMIC measurement procedure by considering a Web application (named “Task manager”) developed for a Spanish Web development company using OO-H (and VisualWADE). In this company, a Web development project involves a number of tasks that are assigned to a user (a programmer). Every task records the start date, the estimated end date, the priority, and so on. The project manager organizes the tasks in folders according to some criteria: pending tasks, critical tasks, etc. Additionally, external files can be associated to a task (e.g., requirement documents, code). The programmers can also record comments to the tasks and send messages to other programmers. Every working day, programmers can generate a report (daily report) including the information related to the tasks that they were working on. Finally, the stakeholders of the project are recorded as contacts in the application.

Table 1 shows the results of the application of the proposed COSMIC measurement procedure on the Task manager conceptual models. We got 181 CFP given by adding the number of Entry, Exit, Read, and Write data movements.

![UML class diagram](image)

In the sequel we illustrate a concrete application of the mapping and measurements rules taking into account a portion of the Task manager conceptual model consisting of the UML class diagram shown in Figure 5 and the NAD modeling the information on CONTACTS presented in Figure 6. The complete size measurement report is available at [http://www.dmi.unisa.it/people/sarro/www/research/OO-HCFP.html](http://www.dmi.unisa.it/people/sarro/www/research/OO-HCFP.html).

4.1 Applying the Mapping Rules

From Figure 6 we can identify one navigational process, CONTACTS, and two functional sub-processes, the method New of the Navigational Class CONTACTO and the method New of the Navigational Class CONTACTO1. The Triggered Event in this diagram is represented by the set composed of the Entry Point, the filled circle in the top right side, the R-Link (LR5), and the Collection “menu contacto”. The data groups identified here are the Navigational Classes and the Collection of the NAD (i.e., Menu contacto, Contacto, Contacto1) and the related part of the UML class diagram shown in Figure 6 (i.e., the CONTACT Class). The data attributes to consider here are all the attributes of the classes involved: the class CONTACT of the UML class diagram, and the Navigational Classes CONTACTO and CONTACTO1 of the NAD.
Finally, we have counted one Write by applying Rule 15 on the S-link (LS16) whose attributes are relative to class CONTACT.

**5. RELATED WORK**

In this section, we discuss the FSM existing approaches for sizing model-driven Web applications [1][3][6][11]. In particular, the work described in the present paper follows the same line of [2], where a Function Point measurement procedure (OO-HFP) was defined for OO-H Web applications. Thus, the main difference with the present one lies in the use of the first generation of Functional User Measurement methods, namely Function Point Analysis (FPA). The counting of OO-HFP was automated and integrated in the VisualWade tool. Moreover, a validation of the approach was carried out by comparing its estimation accuracy with the one achieved by using the set of measures defined by Mendes et al. for the Tukutuku database [19]. The results of the empirical study were promising since the obtained effort estimates were comparable with those obtained using the Tukutuku measures, thus revealing that the OO-HFP approach can be suitable to estimate development effort.

Another model-driven approach to automatically size Web applications, in terms of Function Points, was provided in [11]. It was based on the Web Modeling Language (WebML) and the supporting tool WebRatio [7]. An initial validation of the approach was performed by comparing the Function Points count with the results obtained manually by two skilled analysts.

In [1] [2] a measurement procedure for Web applications, called OO-Method Function Points for the Web (OOmFPWeb), was proposed. It was designed to conform to the IFPUG (International Function Point Users Group) counting rules for FPA [13] and to be used within the context of a model-driven production method for Web applications called OOWS (Object-Oriented Web Solutions) [21]. A family of experiments was carried out to evaluate OOmFPWeb. Results showed that OOmFPWeb is efficient when compared to current industry practices [4].

Marin et al. [16][18] proposed a measurement procedure, namely OO-Method COSMIC Function Points (OOmCFP), to measure the functional size of object-oriented applications generated from their conceptual models. In particular, the proposal is based on COSMIC method and uses the OO-Method development method [38] as the reference MDA approach. The authors also presented the mechanisms that are necessary to automate the OOmCFP measurement procedure and a tool that realizes it [17]. To design and automate OOmCFP, the authors applied the same strategy defined in [2], where a measurement procedure, called OO-Method Function Points (OOmFP), was designed, automated and validated to measure the functional size of OO applications obtained with OO-Method.

**6. CONCLUSIONS AND FUTURE WORK**

We have defined and applied a new measurement procedure based on the COSMIC method for sizing Web applications developed with the OO-H method. The proposed procedure, named OO-HCFP, is based on the use of a process model for software measurement [15], which consists of four steps. In the paper, we have focused on the first two activities of the measurement process model, namely design and application. OO-HCFP was conceived by mapping COSMIC and OO-H concepts. The mapping and measurements rules allow us to obtain the functional size of the OO-H Web applications from OO-H conceptual models.
layers are organized in a hierarchy

A layer is a partition resulting from the functional division of a software architecture which together with hardware forms a whole computer system where:

- layers are organized in a hierarchy

Table 2. The COSMIC Software Context Model [8]

<table>
<thead>
<tr>
<th>PRINCIPLES – The COSMIC Software Context Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software is bounded by hardware</td>
</tr>
<tr>
<td>A layer is a partition resulting from the functional division of a software architecture which together with hardware forms a whole computer system where:</td>
</tr>
<tr>
<td>* layers are organized in a hierarchy</td>
</tr>
</tbody>
</table>

Future work will be concerned with the execution of the last steps of the employed process model for software measurement. To this end, we will validate the use of OO-HCFP for Web effort estimation by comparing this approach with others. Moreover, we will automate the application of the OO-HCFP in VisualWADE, with the aim of having a procedure that avoids the ambiguity of interpreting the COSMIC counting rules. This will also avoid the need for training in order to apply COSMIC in an accurate and repeatable way.

7. REFERENCES


A layer may contain one or more separate 'peer' pieces of software and any one piece of software may further consist of separate peer components.

The software in any two layers in the software architecture that exchange data interpret only part of that data identically.

The scope of a piece of software to be measured shall depend on the purpose of the measurement.

The functional users of a piece of software shall be identified from the functional user requirements of the piece of software to be measured as the senders and/or intended recipients of data.

A piece of software interacts with its functional users via data movements across a boundary and the piece of software may move data to and from persistent storage within the boundary.

The FUR of software may be expressed at different levels of granularity.

The level of granularity at which measurements should normally be made is that of the functional processes.

If it is not possible to measure at the level of granularity of the functional processes, then the FUR of the software should be measured by an approximation approach and scaled to the level of granularity of the functional processes.

Table 3. The COSMIC Generic Software Model [8]

<table>
<thead>
<tr>
<th>COSMIC concept</th>
<th>Rule</th>
<th>OO-H modeling primitives</th>
</tr>
</thead>
<tbody>
<tr>
<td>FURs</td>
<td>1</td>
<td>OO-H Platform-Independent Model (containing the UML class diagram and the NADs)</td>
</tr>
<tr>
<td>Layer</td>
<td>2.1</td>
<td>(User Interface Layer) Navigational classes of the NADs</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td>(Business logic Layer) I-Links between the Navigational classes of the NADs and S-Links</td>
</tr>
<tr>
<td></td>
<td>2.3</td>
<td>(Data Layer) UML class diagram</td>
</tr>
<tr>
<td>Peer component</td>
<td>3.1</td>
<td>Navigational class of the NADs</td>
</tr>
<tr>
<td></td>
<td>3.2</td>
<td>Class of the UML class diagram</td>
</tr>
<tr>
<td>Scope</td>
<td>4</td>
<td>UML class diagram and NADs or a subset of NADs and (connected) subset of classes in the UML class diagram</td>
</tr>
<tr>
<td>Functional users</td>
<td>5.1</td>
<td>Class with the stereotype &lt;&lt;Actor&gt;&gt;, &lt;&lt;User&gt;&gt;, or &lt;&lt;Legacy&gt;&gt; in the UML class diagram</td>
</tr>
<tr>
<td></td>
<td>5.2</td>
<td>Peer component outside the boundary of the scope and that is in a client/server relationship with the software to measure</td>
</tr>
<tr>
<td></td>
<td>5.3</td>
<td>Human user who interact with the system</td>
</tr>
<tr>
<td>(Inside the) Boundary</td>
<td>6</td>
<td>Subset of the classes of the UML class diagram and related NADs</td>
</tr>
<tr>
<td>Functional Process</td>
<td>7</td>
<td>Navigational Target of a NAD</td>
</tr>
<tr>
<td>Functional Sub-Process</td>
<td>8</td>
<td>Method of a Navigational class</td>
</tr>
<tr>
<td>Triggered event</td>
<td>9.1</td>
<td>Entry point with R-Link and Collection (of a NAD)</td>
</tr>
<tr>
<td></td>
<td>9.2</td>
<td>Entry point with R-Link and Navigational Class (of a NAD)</td>
</tr>
<tr>
<td>Data group</td>
<td>10.1</td>
<td>Classes of UML class diagrams; Aggregation hierarchy; Composition hierarchy; Inheritance hierarchy (a data group for the father class and data group for the child class if it includes new attributes).</td>
</tr>
<tr>
<td></td>
<td>10.2</td>
<td>Navigational Classes, that appear into the Navigational Target of the NADs considered as FURs</td>
</tr>
<tr>
<td></td>
<td>10.3</td>
<td>Collection that appears into the Navigational Target of the NADs considered as FURs</td>
</tr>
<tr>
<td>Data attribute</td>
<td>11</td>
<td>Each attribute of each class with the stereotype &lt;&lt;Class&gt;&gt;, &lt;&lt;Actors&gt;&gt;, &lt;&lt;Legacy&gt;&gt;, or &lt;&lt;User&gt;&gt; of the UML class diagram</td>
</tr>
</tbody>
</table>

Table 4. Summary of mapping rules

<table>
<thead>
<tr>
<th>Data movement</th>
<th>Rule</th>
<th>OO-H characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>1</td>
<td>1 R-link that points to a collection and is triggered by a human user to start a functional process (user interaction set to manual)</td>
</tr>
<tr>
<td></td>
<td>12.1</td>
<td>1 R-link that points to a navigational class and is triggered by a human user to start a functional process (user interaction set to manual)</td>
</tr>
<tr>
<td></td>
<td>12.2</td>
<td>1 I-link between navigational classes or between navigational classes and collections into the same navigational target (user interaction set on manual)</td>
</tr>
<tr>
<td></td>
<td>12.3</td>
<td>1 S-link with the hidden mode</td>
</tr>
<tr>
<td></td>
<td>12.4</td>
<td>1 S-link with the constant mode</td>
</tr>
<tr>
<td></td>
<td>12.5</td>
<td>1 S-link with the immediate mode</td>
</tr>
<tr>
<td></td>
<td>12.6</td>
<td>Question mark (?) for a single data group on an I-link</td>
</tr>
<tr>
<td>Exit</td>
<td>13.1</td>
<td>1 I-link between navigational classes or between navigational classes and collections into the same navigational target</td>
</tr>
<tr>
<td></td>
<td>13.2</td>
<td>1 R-link that points to a navigational class or collection into a navigational target</td>
</tr>
<tr>
<td></td>
<td>13.3</td>
<td>1 error message thrown by the system</td>
</tr>
<tr>
<td>Read</td>
<td>14.1</td>
<td>1 R-link that points to a navigational class or collection into a navigational target</td>
</tr>
<tr>
<td></td>
<td>14.2</td>
<td>1 I-link between navigational classes into the same navigational target</td>
</tr>
<tr>
<td></td>
<td>14.3</td>
<td>For each data group involved into an OCL constraint on the I-link</td>
</tr>
<tr>
<td>Write</td>
<td>15</td>
<td>For each data group involved into the execution of a S-link</td>
</tr>
</tbody>
</table>