Intl. Conf. RIVF'04
February 2-5, Hanoi, Vietnam

Toward an ontological formalisation for a software functional size measurement method's application process: The FPA case

Evariste Valéry Bévo Wandji, Ghislain Lévesque, Jean-Guy Meunier

Abstract—**One of the key parameters in estimation activities (effort, costs) is the size. This important parameter can be expressed in terms of function points or lines of codes. The major advantage of software functional size measurement is the fact that it can be performed very early in the life cycle of the software (analysis or design). Despite significant improvement observed these last years (efforts in standardization for example), software functional size measurement remains a non-obvious activity, especially when applying measurement methods. A proper expertise for each method is needed. The implementation of existing software functional size measurement systems in the industry has to deal with two major technical difficulties: the difficulty of applying measurement methods (which makes the task of the measurer tiresome and requires sometimes the help of one or several experts, unfortunately not always available) [6], the lack of support tools to help measurers in their task. Relatively to this last point, Sue Black & David Wigg [5] noticed that "The software industry needs software measurement tools which can be used to compute measures across several platforms and languages to provide flexibility and usability". Today, many research laboratories are working on the development of measurement tools for software functional size measurement [10][9][22]. The design of such tools is necessarily based on: (1) the identification of all the concepts handled in a method's measurement procedure, as well as the relationships between these concepts (domain ontology); (2) the identification of all the tasks associated with a method's measurement procedure, as well as the links between these tasks (tasks ontology). This paper adresses an ontological perception of a method's measurement procedure (a software functional size measurement method's application process). The emphasis is put on domain and tasks ontologies associated with the procedure. The first two papers on this topic [2][3] looked at COSMIC-FFP and MkII-FPA's measurement procedures respectively as case study. This one concentrates on the FPA's measurement procedure.**

Index Terms—**Measurement method, measurement procedure, ontology, software functional size.**

I. Introduction

Estimation activities (effort, costs) at the beginning of any project are very important tasks for projects managers. One of the key parameters in these activities is the size. In general, the size can be expressed in terms of function points or lines of codes. The major advantage of software functional size measurement is the fact that it can be performed very early in the life cycle of the software (analysis or design). However, the software being an intellectual product, an "abstract object" [19], measuring its size in general and its functional size in particular, is not an obvious task [20]. The lack of support tools to help measurers in their task makes the task even more difficult. Today, many research laboratories are working on the development of measurement tools for software functional size measurement [10][9][22]. To design such tools the following aspects should be taken into account: (1) the identification of all the concepts handled in a method's measurement procedure, as well as the relationships between these concepts (domain ontology); (2) the identification of all the tasks associated with a method's measurement procedure, as well as the links between these tasks (tasks ontology). The existence of such ontologies in simple, expressive, clear and precise formalisms, would save much time to designers of measurement tools; it would also be helpful for a better comprehension of related method's measurement procedure, and serve as a consensual platform or rather solid basis for representation, exchange and interpretation of information, "things" ("concepts") related to a measurement procedure and measurement in general. They can be seen as the foundations for a body of knowledge associated with each method's measurement procedure.

What is the exact content of such ontologies? That's one of the key questions this paper will try to answer by proposing some examples. An overview on the place of ontologies in a method's measurement procedure is provided in section 2. The question of the formalism to use for these ontologies is examined in section 3. Finally an example of a domain ontology associated with the FPA's measurement procedure is presented, as well as an example of a tasks ontology associated with the same procedure.

Evariste Valéry Bévo Wandji, Ghislain Lévesque, Jean-Guy Meunier
L.R.G.L – L.A.N.C.I, UQAM (emails: bevo_wandji.evariste@courrier.uqam.ca,Levesque.Ghislain@uqam.ca, Meunier.Jean-Guy@uqam.ca).

II. ONTOLOGIES *VS* A METHOD’S MEASUREMENT PROCEDURE

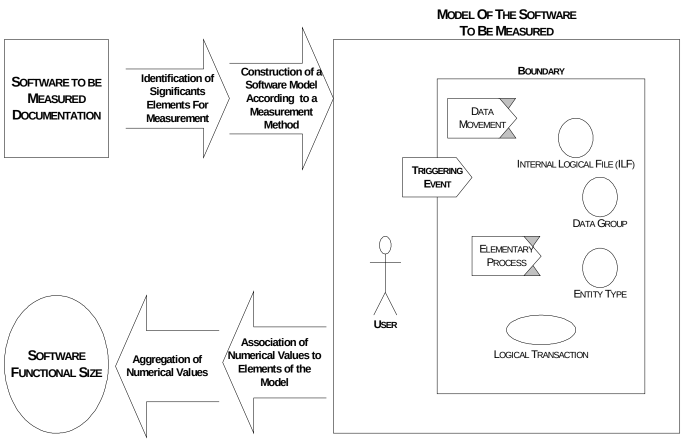
Model Of The Software To Be Measured
Software to be Measured Documentation
Identification of Significants Elements For Measurement
Construction of a Software Model According to a Measurement Method
Boundary
Data Movement
Internal Logical File (ILF)
Triggering Event
Data Group
Elementary Process
Entity Type
User
Logical Transaction
Software Functional Size
Aggregation of Numerical Values
Association of Numerical Values to Elements of the Model

Figure 1 : An abstraction of a method’s measurement procedure (*Adapted from [11]*)

It clearly appears from figure 1 that a method’s measurement procedure has two (2) main phases: an "identification and modeling" phase (identification of the relevant elements relatively to measurement and construction of a software model, in accordance with the selected measurement method), and a "measurement" phase (assignment/association of numerical values to some elements of the previously built model and aggregation of these numerical values to derive the functional size of the software). Each of these phases consists of a series of specific measurement tasks [9], described in measurement handbooks in natural language (English, French ...). Each measurement task has data, information or control data as input and produces data, information or control data as output. It would be advisable to gather these tasks in a real tasks ontology (according to Mizoguci’s definition of tasks ontology [16][17]), presented in a simple and expressive formalism, with a clear and precise description of tasks. This would constitute a good step towards the complete or partial automation of these tasks, as well as contributing to a better comprehension of a method’s measurement procedure.

A method’s measurement procedure is based on two key elements: A software functional size measurement method (concepts, measurement tasks & measurement rules) and the specifications of the software to be measured (specifications or design documents...). The first task of a "measurer" when applying a software functional size measurement method consists in producing a description of the being measured software from specifications documents, using concepts of a language related to the measurement method: Only elements of the software which are relevant relatively to measurement are selected. He thus obtains a software model relatively to the considered measurement method. Such a model is just an instantiation of what is commonly called the "meta-model" (“static model” [4], “generic software model” or ”context model” [1]) of the considered measurement method, from specification documents of the software being measured. The “static model” of a measurement method should correspond to the domain ontology (« explicit formal specifications of the terms in the domain and relations among them » [14]) related to the method’s measurement procedure. Very often (if not almost always) in measurement handbooks, the static models appear in the form of a lexicon (list of concepts with their respective definitions). The relationships between concepts are not always clarified. A “measurer” thus does not have for each method’s measurement procedure (the current ones), a real domain ontology (according to Mizoguci’s definition of domain ontology [16][17]), presented in a simple and expressive formalism. In fact, only part of such an ontology is available.

All things considered, the existence of a domain ontology and tasks ontology related to a method’s measurement procedure, presented in an adequate formalism, with tasks and concepts clearly and precisely described, would save much time to designers of measurement tools, and contribute to a better comprehension of the measurement procedure as well. It would also serve as consensual rallying points to structure, represent, exchange and interpret information, "things" ("concepts") related to measurement procedure and measurement in general. These ontologies make explicit the necessary semantic structures related to each method’s measurement procedure. In the next section we examine the question of the formalism to use for such ontologies.

III. FORMALISMS TO PRESENT ONTOLOGIES RELATED TO A METHOD’S MEASUREMENT PROCEDURE

In measurement handbooks for current software functional size measurement methods, natural language is the language used to present some elements of ontologies related to any method’s measurement procedure. Consequently it often results in a poor uniformity in interpretations, especially among non-experts. A more formal and less ambiguous language would be helpful. Presently, when designing measurement tools, measurement handbooks are not sufficient; the help of some experts of the considered method is needed, in particular for some significant elements which are not explicit in handbooks (some measurement rules, some relationships between measurement concepts...). Clear ontologies may help to reduce such a strong dependence.

Several formalisms exist in the literature for knowledge representation within the framework of ontologies. Some of them are well known, namely: relational formalisms [object–oriented types (or frames) or semantic networks types (or conceptual graphs)] and logic based formalisms [predicates based logic, fuzzy logic, etc.]. Formal languages are appropriate for automation while a good expressivity and simplicity characterize relational languages. In general, one needs to find a good compromise between expressivity, simplicity and efficiency. Object-oriented formalism (UML class diagram, UML component diagram [6]) was selected for domain ontologies and tasks hierarchy. The simplicity, expressivity and popularity of this formalism guided the choice (the fact that it is independent of any implementation tool was also taken into account). In future work this formalism will be combined with Standford certitude theory to take into account uncertainty associated with some aspects of measurement. Tasks and concepts description is inspired from CommonKADS methodology [21].

CommonKADS is a methodology developed by a number of industry-university consortia over the past decade and nowadays in use worldwide by companies and educational institutions. According to their authors [Schreiber et al. 99], it offers a structured approach, covering the complete route from corporate knowledge management to knowledge analysis and engineering, all the way to "knowledge-intensive" systems design and implementation, in an integrated fashion. Their way of describing concepts and tasks was exploited in this paper.

The next section is dedicated to some examples of ontology associated with the FPA's measurement procedure. FPA [15] is one of the three (3) functional size measurement methods approved as an International Standard (ISO) at this date. FPA is older than the two odther methods. It is one of the few functional size measurement method to be widely used in the industry, especially in north America.

IV. EXAMPLES OF ONTOLOGY RELATED TO A METHOD'S MEASUREMENT PROCEDURE

A. *Domain ontology related to FPA's measurement procedure*

FPA measurement manual v4.1 [15] was necessary to produce this ontology. An analysis of the FPA measurement procedure was performed. All relevant concepts related to the procedure were identified and their respective definition extracted from the measurement manual. Relationships between identified concepts were then examined. The concepts and relationships between them were then represented using the chosen formalism (Object-oriented formalism [UML class diagram] [6]). In order to document this representation, the CommonKADS methodology was put to contribution in describing represented concepts and relationships between them. The help of FPA experts is still needed to refine and strengthen this work.

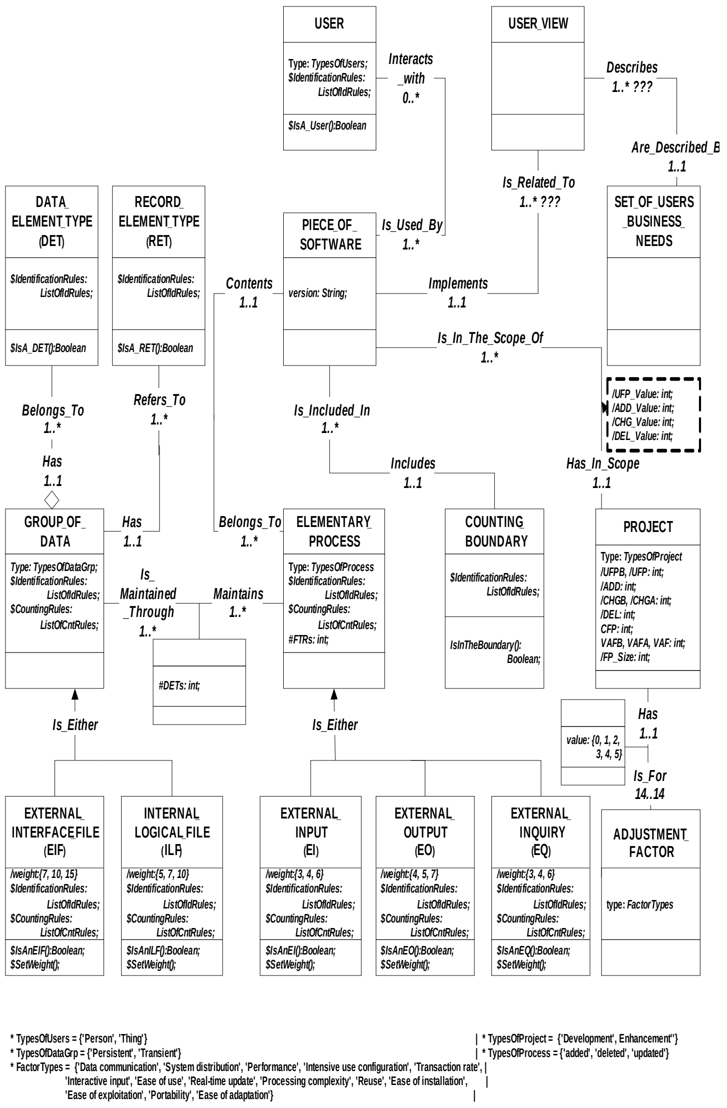
USER
Type: TypesOfUsers; $IdentificationRules: ListOfIdRules;
$IsA_User():Boolean
Interacts _with 0..*
USER_VIEW
Describes 1..* ???
Are_Described_B 1..1
Is_Related_To 1..* ???
SET_OF_USERS _BUSINESS_ NEEDS
DATA_ ELEMENT_TYPE (DET)
$IdentificationRules: ListOfIdRules;
$IsA_DET():Boolean
RECORD_ ELEMENT_TYPE (RET)
$IdentificationRules: ListOfIdRules;
$IsA_RET():Boolean
PIECE_OF_ SOFTWARE
version: String;
Is_Used_By 1..*
Contents 1..1
Implements 1..1
Is_In_The_Scope_Of 1..*
/UFP_Value: int; /ADD_Value: int; /CHG_Value: int; /DEL_Value: int;
Belongs_To 1..*
Has 1..1
Refers_To 1..*
Is_Included_In 1..*
Includes 1..1
Has_In_Scope 1..1
GROUP_OF_ DATA
Type: TypesOfDataGrp; $IdentificationRules: ListOfIdRules; $CountingRules: ListOfCntRules;
Has 1..1
Is_ Maintained _Through 1..*
Maintains 1..*
#DETs: int;
Belongs_To 1..*
ELEMENTARY_ PROCESS
Type: TypesOfProcess $IdentificationRules: ListOfIdRules; $CountingRules: ListOfCntRules; #FTRs: int;
COUNTING_ BOUNDARY
$IdentificationRules: ListOfIdRules;
IsInTheBoundary(): Boolean;
PROJECT
Type: TypesOfProject /UFPB, /UFP: int; /ADD: int; /CHGB, /CHGA: int; /DEL: int; CFP: int; VAFB, VAFA, VAF: int; /FP_Size: int;
Has 1..1
value: {0, 1, 2, 3, 4, 5}
Is_For 14..14
Is_Either
Is_Either
EXTERNAL_ INTERFACE_FILE (EIF)
/weight:{7, 10, 15} $IdentificationRules: ListOfIdRules; $CountingRules: ListOfCntRules;
$IsAnEIF():Boolean; $SetWeight();
INTERNAL_ LOGICAL_FILE (ILF)
/weight:{5, 7, 10} $IdentificationRules: ListOfIdRules; $CountingRules: ListOfCntRules;
$IsAnILF():Boolean; $SetWeight();
EXTERNAL_ INPUT (EI)
/weight:{3, 4, 6} $IdentificationRules: ListOfIdRules; $CountingRules: ListOfCntRules;
$IsAnEI():Boolean; $SetWeight();
EXTERNAL_ OUTPUT (EO)
/weight:{4, 5, 7} $IdentificationRules: ListOfIdRules; $CountingRules: ListOfCntRules;
$IsAnEO():Boolean; $SetWeight();
EXTERNAL_ INQUIRY (EQ)
/weight:{3, 4, 6} $IdentificationRules: ListOfIdRules; $CountingRules: ListOfCntRules;
$IsAnEQ():Boolean; $SetWeight();
ADJUSTMENT_ FACTOR
type: FactorTypes
* TypesOfUsers = {'Person', 'Thing'} | * TypesOfProject = {'Development', Enhancement''}
* TypesOfDataGrp = {'Persistent', 'Transient'} | * TypesOfProcess = {'added', 'deleted', 'updated'}
* FactorTypes = {'Data communication', 'System distribution', 'Performance', 'Intensive use configuration', 'Transaction rate', 'Interactive input', 'Ease of use', 'Real-time update', 'Processing complexity', 'Reuse', 'Ease of installation', 'Ease of exploitation', 'Portability', 'Ease of adaptation'}

Figure 2 : Domain ontology associated with FPA's measurement procedure *Adapted from [4]*

Below are descriptions of concepts appearing in Figure 2.

NB: All the definitions have been taken from the FPA Measurement Manual, v4.1 [15].

1) *Internal Logical Files (ILF)*

Definition:

An Internal Logical File (ILF) is a user identifiable group of logically related data or control information maintained within the boundary of the application. The primary intent of an ILF is to hold data maintained through one or more elementary processes of the application being counted.

Properties:

- *Name: {String};*
- *Description: {String};*
- *Weight: {int};*
- *IdentificationRules: {IdRule}*
- *CountingRules: {CntRule}*

Concept: IdRule

Condition: {logical expression}
Action: {procedure}
Concept: CntRule
Condition: {logical expression}
Action: {procedure}

2) External Interface Files (EIF)

Definition:

An External Interface File (EIF) is a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of of another application. The primary intent of an EIF is to hold data referenced through one or more elementary processes within the boundary of the application being counted. This means that an EIF counted for an application must be an ILF in another application.

Properties:

- *Name: {String};*
- *Description: {String};*
- *Weight: {int};*
- *IdentificationRules: {IdRule}*
- *CountingRules: {CntRule}*

Concept: IdRule
Condition: {logical expression}
Action: {procedure}
Concept: CntRule
Condition: {logical expression}
Action: {procedure}

NB: All other concepts are described at the following address: http://www.labunix.uqam.ca/~bevo/FPA/ontologies.htm

Details of relationships between concepts appearing in Figure 2 are presented below.

1) Piece_Of_Software - Elementary_Process

Description:
The elementary processes identified in a piece of software.
Arguments:
Piece_Of_Software
Roles: The number of pieces of software related to a given elementary process. [A given elementary may be identified in one and only one version of a piece of software.]
Cardinality: min 1; max 1;
Elementary process
Roles: The umber of elementary processes identified in a given piece of software. [A given version of a piece of software can contain many elementary processes.]
Cardinality: min 1; max n;

NB: All other relationships between concepts are described at the following address: http://www.labunix.uqam.ca/~bevo/FPA/ontologies.htm

The next section is devoted to task ontology.

A. Tasks ontology related to FPA's measurement procedure

Personal measurement experience and the FFA Measurement Manual v4.1 [15] were necessary to produce this ontology. An analysis of the FPA measurement procedure was performed. All relevant tasks related to the procedure were identified. Relationships between the identified tasks were then examined to produce a task hierarchy. The tasks and the relationships between them were then represented using the chosen formalism (Object-Oriented formalism [UML component diagram] [6]). In order to document this representation, the CommonKADS methodology was used to describe the tasks represented and the relationships between them. The help of FPA experts is still needed to refine and strengthen this work.

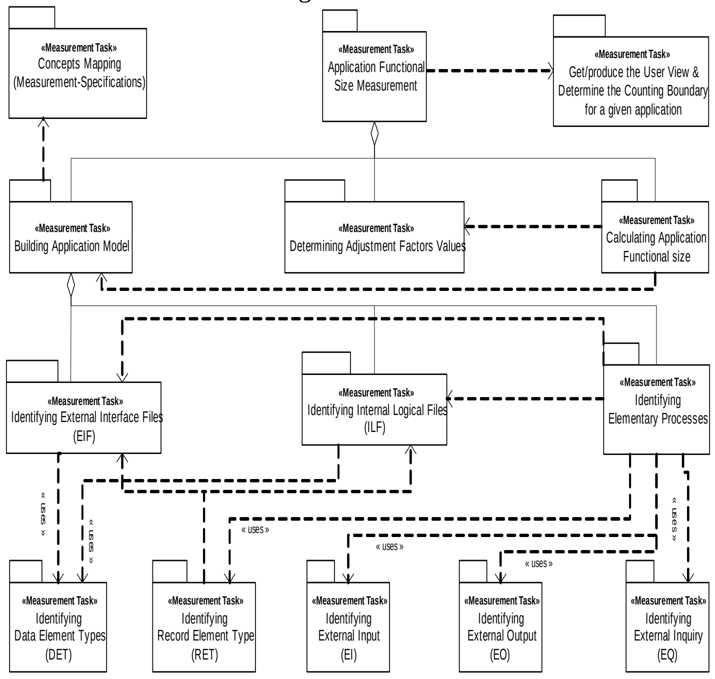
«Measurement Task» Concepts Mapping (Measurement-Specifications)
«Measurement Task» Application Functional Size Measurement
«Measurement Task» Get/produce the User View & Determine the Counting Boundary for a given application
«Measurement Task» Building Application Model
«Measurement Task» Determining Adjustment Factors Values
«Measurement Task» Calculating Application Functional size
«Measurement Task» Identifying External Interface Files (EIF)
«Measurement Task» Identifying Internal Logical Files (ILF)
«Measurement Task» Identifying Elementary Processes
« uses »
« uses »
« uses »
« uses »
« uses »
« uses »
«Measurement Task» Identifying Data Element Types (DET)
«Measurement Task» Identifying Record Element Type (RET)
«Measurement Task» Identifying External Input (EI)
«Measurement Task» Identifying External Output (EO)
«Measurement Task» Identifying External Inquiry (EQ)

Figure 3 : Tasks ontology related to FPA's measurement procedure (Based on [15])

Below are descriptions of the tasks appearing in Figure 3.

1) Application Functional Size Measurement

Definition:

Goal: *Given a user view (formal description of the user's business functions in the user's language) related to a piece of software (an application) and a counting boundary, the task should produce a FPA model for the piece of software, determine the related adjustment factors values and calculate its functional size in FPA Functional Size Units. It is a composition of three (3) complementary tasks: "Building Application Model", "Determining Adjustment Factors Values" and "Calculating Application Functional size".*
Inputs: *User view, Counting boundary, Set of mapping results*
Outputs: *FPA model for the piece of software, functional size of the piece of software, [global precision/certainty of the measurement]*

Body:

Type: *composition*
Sub-Tasks: *"Building Application Model", "Determining Adjustment Factors Values" and "Calculating Application Functional size"*
Control loop: *execute sequentially:*

- *Building_Application_Model (+User_View, +Counting_Boundary, +Set_Of_Mapping_Results, -FPA_model_for_the_piece_of_software),*
- *Determining_Adjustment_Factors_Values (+User_View, +Project_Characteristics, -Adjustment_Factors_Values[]),*
- *Calculate_Application_Functional_Size (+FPA_model_for_the_piece_of_software , +Adjustment_Factors_Values[], -functional_size_of_the_piece_of_software, -global_precision/certainty_of_the_measurement)*

NB: All other tasks are described at the following address:
http://www.labunix.uqam.ca/~bevo/FPA/ontologies.htm

V. Research perspectives

An XML, then XMI translation of the presented ontologies is to be produced, using adequate tools [12][8]. It could be very useful for measurement results analysis & documentation tools, and some aspects of measurement results validation. XML being appropriate for documents exchange, historical databases on projects could be extended to store more detailed data related to projects functional size measurement. For example, if we consider the ISBSG[1] system used to collect historical data on projects, rather than storing textual descriptions of projects and/or numerical values representing functional size for each project, ISBSG could store in XML or XMI format, data on projects (data related to functional size measurement). This work can also be helpful for data exchange between different functional size measurement tools (in the case of many measurement tools for the same functional size measurement method).

An ontology-oriented approach will be introduced for the complete or partial automation of a method's measurement procedure. In fact, the automation work is based on the implementation of ontologies. One of the common uses of ontologies is separating the domain knowledge from the operational knowledge. This is also one of the main advantages of the ontology-oriented approach to be introduced for the complete or partial automation of a method's measurement procedure. With this approach, it is possible to formally determine in advance (without building prealably a tool), whether the procedure is currently completely automatable or not, and if not, what are the automatable tasks and the non-automatable ones. It also ensures that measurement results produced by an automating tool are in an exportable format, thus exploitable by other measurement tools. To illustrate this approach, a prototype is currently under development in LRGL (Software Engineering Research Laboratory), for COSMIC-FFP's application process automation, given specifications produced with some object CASE tools (Together, Objecteering and possibly Rational Rose). The possibility of extending the prototype to any CASE tool producing UML specifications is also considered (the principle of "parser" will be exploited). A first version of the prototype should be available by fall 2003.

A new approach to address the question of inter-methods measurement convertibility is to be introduced. This approach will also be ontology based: links should be established between domain ontologies related to different methods' measurement procedures. Current research work on ontology mapping [16] will be exploited. This work is already in process. COSMIC-FFP and MK II FPA are selected methods for tests. Then the COSMIC-FFP and FPA case will be examined.

[1] International Software Benchmarking Standards Group

References

[1] Abran, A., Desharnais, J.-M., Oligny, S., St-Pierre, D., and Symons, C., *COSMIC FFP - Manuel de mesures version 2.2*, Montréal, Mars, 2003.
[2] Bévo, V., Lévesque, G., and Meunier, J.-G., "Toward an ontological formalisation for a software functional size measurement method's application process: The COSMIC-FFP case", in *IWSM'03*, Montréal, Canada, September 23-25, 2003.
[3] Bévo, V., Lévesque, G., and Meunier, J.-G., "Toward an ontological formalisation for a software functional size measurement method's application process: The MkII-FPA case", in *ICSSEA'03*, CENAM - Paris, France, December 2-4, 2003.
[4] Bévo, V., Lévesque, G., Abran, A., and Meunier, J.-G., "Vers une approche multi-agent pour la mesure de la taille fonctionnelle des logiciels", in *IWSM'01*, Montréal, Canada, 2001
[5] Black, S. and Wigg, D., "X-Ray : A Multi-Language, Industrial Strengh Tool", in *IWSM'99*, Lac Supérieur, Canada, p.39, Sept.ember 8-10, 1999
[6] Booch, G.; Rumbaugh, J.; and Jacobson, I., The Unified Modeling Language User Guide, 1999.
[7] Carpers J., T., *Estimating Software Costs*, McGraw-Hill, New York, pp 9, 269, 276, 1996
[8] Denny, M., "Ontology Building: A Survey of Editing Tools", November 06, 2002. Available at: http://www.xml.com/pub/a/2002/11/06/ontologies.html?page=2.
[9] Desharnais, J.-M., Abran, A., Mayer, A., Buglione, L., and Bévo, V., "Knowledge Modeling for the Design of a KBS in the Functional Size Measurement Domain", in *KES'02*, Italy, 2002
[10] Diab, H., Frappier, M., and St-Denis, R., "A Formal Definition of COSMIC-FFP for Automated Measurement of Room Specifications", in *FESMA*, 2001.
[11] Fetcke, T., "A Generalized Structure for Function Point Analysis", in *IWSM'99*, Lac Supérieur, Canada, September 8-10, 1999.
[12] Fridman Noy, N., and McGuinness, D., "Ontology Development 101: A Guide to Creating Your First Ontology", March, 2001. Available at: http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html.
[13] Gramantieri, F., Lamma, E., Riguzzi, F., and Mello, P., "A system for measuring function points from specifications", 1997.
[14] Gruber T. R., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", Presented at the *International Workshop on Formal Ontology*, Padova, Italy. To appear in *a collection edited by Nicola Guarino*. Available as Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University, March 1993. {robert.stevens carole seanb}@cs.man.ac.uk.
[15] IFPUG, *Function Point Analysis Counting Practices Manual, release 4.1*, Mequon, Wisconsin, International Function Point Users Group (IFPUG), 2000.

[16] Madhavan, J., Bernstein, P., Domingos, P., and Halevy, A., "Representing and Reasoning about Mappings Between Domain Models", *Proceedings of the AAAI Eighteenth National Conference on Artificial Intelligence*, 2002.

[17] Mizoguchi, R., "Ontological Engineering : Ontology design and knowledge systematisation", in *CIRTA*, Montreal, Canada, August 2002

[18] Mizoguchi R. and Bourdeau J., "Using Ontological Engineering to Overcome Common AI-ED problems", in *International Journal of Artificial Intelligence in Education*, 2000, 11, 107-121.

[19] Naur, P. and Randel, B., "Software Engineering : A Report on a Conference sponsored by the NATO Science Committee", NATO, 1969

[20] Putnam, L., H., and Myers, W., *Measures for excellence: Reliable software on time, within budget*, Yourdon Press, Prentice Hall Building, Englewood Cliffs, New Jersey 07632, p.61,64, 1992

[21] Schreiber, G., Akkermans, H., Anjewierden, A., De Hoog, R., Shadbolt, N., Van de Velde, W., and Wielinga, B., *Knowledge engineering and management: The CommonKADS methodology*, A Bradford Book, The MIT Press, Cambridge, Massachusetts, 1999

[22] Uemura, T., Kusumoto, S., and Inoue, K., "Function-point analysis using design specifications based on the Unified modelling Language", in *Journal of software maintenance and evolution : Research and practice*, June 2001

[23] UKSMA Metrics Practices Committee, *MK II Function Point Analysis Counting Practices Manual., v.1.3.1*, UK, September 2000.