UPGRADE
The European Journal for the Informatics Professional
http://www.upgrade-cepis.org
Vol. V, No. 2, April 2004

UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by NOVÁTICA <http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática <http://www.ati.es/>).
UPGRADE is also published in Spanish (full issue printed, some articles online) by NOVÁTICA, and in Italian (abstracts and some articles online) by the Italian CEPIS society ALSI <http://www.alsi.it> and the Italian IT portal Tecnoteca <http://www.tecnoteca.it/>.
UPGRADE was created in October 2000 by CEPIS and was first published by NOVÁTICA and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>).

E-mail addresses for editorial correspondence: <rfcalvo@ati.es>, <nicolet@acm.org> or <rcarniel@dgt.uniud.it>

E-mail address for advertising correspondence: <novatica@ati.es>

Upgrade **Newslist** available at <http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

ISSN 1684-5285

Next issue (June 2004):
"Digital Signature"
(The full schedule of **UP**GRADE is available at our website)

UML and Model Engineering

Guest Editors: **Jesús García-Molina, Ana Moreira, and Gustavo Rossi**

Joint issue with NOVÁTICA*

Mosaic

* This monograph will be also published in Spanish (full issue printed; summary, abstracts and some articles online) by NOVÁTICA, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática) at <http://www.ati.es/novatica/>, and in Italian (online edition only, containing summary abstracts and some articles) by the Italian CEPIS society ALSI and the Italian IT portal Tecnoteca at <http://www.tecnoteca.it>.

Metrics for UML Models

Marcela Genero, Mario Piattini-Velthuis, José-Antonio Cruz-Lemus, and Luis Reynoso

Due to the important role played by models obtained in the early stages of the development of Object-Oriented (OO) systems, special attention has been paid to the quality of such models in recent years. While it is commonly believed that UML (Unified Modeling Language) is an aid to building better quality software systems, we need to have quantitative and objective measurement instruments to gather empirical evidence of this. The idea that the use of metrics for UML models can help designers make better decisions is also gaining importance in the field of software measurement, which has led to an increasing demand for metrics for UML models in recent years. The main aim of this paper is, therefore, to present a broad overview of the most relevant proposals related to metrics for UML models at a conceptual level, to give researchers and practitioners a broader insight into the work already done and still to be carried out in the field of UML model quality. This paper will hopefully also give researchers a more comprehensive view of the direction OO measurement is taking.

Keywords: Characteristics, Conceptual Models, Empirical Validation, Model Quality, Object-Oriented Metrics, UML Models, Theoretical Validation.

1 Introduction

In recent years, software developers have paid special attention to guaranteeing the quality characteristics of Object-Oriented (OO) systems from the first stages of their lifecycle, with a special focus on the quality of conceptual models. Although the conceptual modelling phase represents only a small part of the overall OO system development effort, its impact on the system finally implemented is probably greater than that of any other phase.

Recently, paradigms such as Model-Driven Development [1] and Model-Driven Architecture [27] consider conceptual

Marcela Genero is an assistant professor in the Department of Computer Science at the *Universidad de Castilla-La Mancha, Ciudad Real*, Spain. She received her MSc degree in Computer Science from the Department of Computer Science of the *Universidad del Sur*, Argentina, in 1989 and her PhD from the *Universidad de Castilla-La Mancha*. Her research interests are: advanced database design, software metrics, conceptual data model quality, and database quality. She has published several papers in prestigious conferences and journals such as CAiSE, E/R, OOIS, METRICS, ISESE, SEKE, Journal of Systems and Software, International Journal of Software Engineering and Knowledge Engineering, Information and Software Technology, Software Quality Journal, etc. She is co-editor of the books "Information and database quality", 2002, Kluwer Academic Publishers, USA, and "Metrics for Software Conceptual Models", 2004, Imperial College Press, UK.
<Marcela.Genero@.uclm.es>

Mario Piattini-Velthuis has an MSc and PhD in Computer Science from the *Universidad Politécnica de Madrid* and is a Certified Information System Auditor Manager by ISACA (Information System Audit and Control Association). He is a full professor in the Department of Computer Science at the *Universidad de Castilla-La Mancha, Ciudad Real*, Spain. He has authored several books and papers on databases, software engineering and information systems, and leads the ALARCOS research group of the Department of Computer Science at the same university. His research interests are: advanced database design, database quality, software metrics, and software maintenance and security in information systems. He has co-edited several books: "Advanced Databases: Technology and Design", 2000. Artech House. UK; "Auditing Information Systems" Idea Group Publishing, 2000, USA; "Information and database quality", 2002, Kluwer Academic Publishers, USA, etc. and "Metrics for Software Conceptual Models", 2004, Imperial College Press, UK. He is co-editor of the Database section of Novática.
<mario.piattini@uclm.es>

José-Antonio Cruz-Lemus is an assistant professor in the Department of Computer Science at the *Universidad de Castilla-La Mancha, Ciudad Real*, Spain. He received his MSc degree in Computer Science and is currently a Ph.D. student at the same university. His research interests are: metamodelling using UML, metrics for UML models, etc. <joseantonio.cruz@uclm.es>

Luis Reynoso is an assistant professor at the *Universidad Nacional de Comahue, Neuquen*, Argentina. He received his MSc degree in Computer Science from the *Universidad Nacional del Sur*, Argentina, in 1993. He also obtained a Magister in Computer Science from the same university in 2003. He was a fellow of the International Institute of Software Technology, one of the Research and Training Centres of the United Nations University, researching on a project about object-oriented design patterns. He has published papers in several international conferences. His research interests are focused on object oriented metrics and the combination of formal and informal methods applied to software engineering.
<lreynoso@uncoma.edu.ar>

models as the backbone of OO system development and stress the importance of building 'good' conceptual models.

While the appearance of UML as a modelling language has been a quantum leap for building higher quality models, the truth of the matter is that a modelling language can only give us syntax and semantics to work with, but cannot tell us whether or not a 'good' model has been produced. Naturally, even when a language is mastered there is no guarantee that the models produced with it will be good. It is rather like writing a story in a natural language: the language is merely a tool that the author has to master. It is still up to the author to write a good story.

Quality in conceptual modelling has traditionally been a poorly understood area. In practice, the evaluation of the quality of conceptual models is performed in an ad hoc manner, if at all. There are no generally accepted guidelines for evaluating the quality of conceptual models, and little agreement, even among experts, as to what makes a model 'good'. As a result, the quality of conceptual models produced in practice is almost entirely dependent on the competence of the modeller.

Expert modellers know intuitively what makes a data model good, but this knowledge can generally be acquired only through experience. However, for conceptual modelling to progress from a 'craft' to an engineering discipline, the desirable qualities of conceptual models need to be made explicit.

Some interesting frameworks have been proposed [21] [19] [26] [31]. While these frameworks have contributed to a better understanding of the concept of quality in conceptual modelling, they lack the quantitative and objective measures required to reduce the level of subjectivity and bias in the quality evaluation process.

It is widely recognised that, in practice, quality criteria are not in themselves enough to ensure quality, because people will generally interpret the same concept in different ways. According to TQM (Total Quality Management) literature, measurable criteria for assessing quality is necessary to avoid "style arguments". This is where software measurement plays an important role. A metric is a way of measuring a quality factor in a consistent and objective manner. It is therefore essential to have metrics which allow us to evaluate the characteristics that ensure the quality of conceptual models (in our case, metrics for UML models) and thereby help us to develop good quality OO systems by eliminating incorrect design attributes and reducing unnecessary complexity from the first stages of the development cycle, leading to a major reduction in extra work during and after implementation.

We should also bear in mind that in many cases UML models need to be complemented with OCL (Object Constraint Language) specifications which allow certain constraints to be defined that cannot be expressed by diagrammatic notation alone.

The main aim of this paper is to present an overview of current literature on metrics applicable to the measurement of quality attributes for UML structural diagrams (class diagrams), UML behavioural diagrams (use case diagrams, statechart diagrams) and OCL expressions.

A number of experts in software measurement put special emphasis on certain issues that must be taken into account when defining metrics for software, such as:

- Metrics must be defined with clear aims in mind.
- Metrics must be theoretically validated, by addressing the question "is the metric actually measuring the attribute it is supposed to be measuring?".
- Metrics must be empirically validated, by addressing the question *"is the metric of practical use?"* (in the sense that it is related to other external quality attributes as intended.)
- The calculation of the metrics must be easy and should ideally be automated by means of a tool.

We will consider these suggestions when presenting each of our proposals.

The rest of the paper is organised as follows: Sections 2 to 4 outline our different proposals for UML diagram metrics (use cases diagrams, class diagrams and statechart diagrams, respectively). Metrics for OCL expressions are presented in Section 5. The final section presents some concluding remarks and briefly describes future and emerging trends in the field of metrics for UML models.

2 Metrics for UML Use Case Diagrams

UML actually only focuses on the diagrammatic notation of use cases. As several authors have remarked, use case diagrams must be understood as nothing more than a table of contents of use cases, not as an alternative to their textual specification. In use case diagrams, only the name of the use case, the participating actors and some use case relationships are shown. The essence of use cases, i.e. their sequence of actor–system interactions, can in no way be derived from use case diagrams. It is therefore necessary to complement use case diagrams with use case textual specification.

There are a number of proposals for specific metrics for use case diagrams, such as [23] and [30]. There are also some others which are specifically used for use cases, among others [14] [18] [4]. A thorough study of metrics for use cases can be found in [15].

Our aim here is to describe existing metrics for use case diagrams, so we will go on to outline the metrics put forward by Marchesi [23] and Saeki [30].

Marchesi [23] proposed a set of metrics to measure use case diagram complexity. He stated that the number of use cases (NCU), the number of actors (Na) and the number of include and extend relationships are good indicators of system complexity.

In [30] a set of metrics for use cases diagrams are defined to obtain an indicator of modifiability. The basic idea behind the defined metrics is that if a use case needs to be changed, other use cases will probably also need to be changed: i.e. those that have a relationship with the originally changed use case. In short, include and extend relationships and control and data dependency relationships are considered. Intuition suggests that the more relationships there are in the model, the more difficult it will be to make any change.

The type of use case is another factor that has an influence on the modifiability of use cases. In simple terms, if a use case has

Metric name	Definition
WMC	The Weighted Methods per Class metric is defined as follows: $WMC = \sum_{i=1}^{n} C_i$ Where c_1, ..., c_n is the complexity of the methods of a class with methods M_1, ...,M_n. If all method complexities are considered as unity, WMC = n, the number of methods.
DIT	The Depth of Inheritance of a class is the DIT metric for a class. In cases involving multiple inheritance, the DIT will be the maximum length from the node to the root of the tree.
NOC	The Number of Children is the number of immediate subclasses subordinated to a class in the class hierarchy.

Table 1: CK Metrics [12].

Metric name	Metric definition
Nassoc	The total number of associations.
Nagg	The total number of aggregation relationships within a class diagram (each whole-part pair in an aggregation relationship).
Ndep	The total number of dependency relationships.
Ngen	The total number of generalisation relationships within a class diagram (each parent-child pair in a generalisation relationship).
NaggH	The total number of aggregation hierarchies (whole-part structures) within a class diagram.
NgenH	The total number of generalisation hierarchies within a class diagram.
MaxDIT	The maximum DIT (Depth of Inheritance Tree) value obtained for each class of the class diagram. The DIT value for a class within a generalisation hierarchy is the longest path from the class to the root of the hierarchy.
MaxHAgg	The maximum HAgg value obtained for each class of the class diagram. The HAgg value for a class within an aggregation hierarchy is the longest path from the class to the leaves.

Table 2: Metrics for Measuring the Structural Complexity of UML Class Diagrams.

several goals (or types according to Saeki), it is more susceptible to change than if it only has one goal. In order to evaluate modifiability, the defined metrics are NOD (Number of Dependencies) and NUCT (Number of Use Case Types). What the author achieved was to define an indicator ($0 \leq$ *MODIFIABILITY* ≤ 1) that would show the degree of modifiability of a use case model.

To our knowledge there is no evidence of any theoretical or empirical validation of these two proposals. Neither is there any automated support for the metrics calculation.

3 Metrics for UML Class Diagrams

The main purpose of this section is to present a summary of the most important proposals for metrics[1] currently applicable to UML class diagrams at a conceptual level, with an analysis of their strengths and weaknesses. Most of the metric proposals we consider (listed below, identified by their author(s)) were not originally defined to measure UML class diagrams, though they can be adapted for this purpose:

- Chidamber and Kemerer [12]. These metrics, also called CK metrics, were defined at class level and their purpose is to measure design complexity in relation to their impact on external quality attributes such as maintainability, reusability, etc. This proposal is one of the most widely known and used. Only three of the six CK metrics are available for a UML class diagram at conceptual level (see Table 1.)
- Li and Henry [20]. These metrics measure different internal attributes such as coupling, complexity and size, and were successfully used to predict maintenance effort. They were defined at class level.
- Brito e Abreu and Carapuça [8]. They were defined to measure the use of OO design mechanisms such as inheritance, information hiding, coupling and polymorphism and the consequent relationship with software quality and development productivity. They can be applied at class diagram level.
- Lorenz and Kidd [22]. They were defined at class level to measure the static characteristics of software design, such as the use of inheritance, the number of responsibilities in a class, etc.
- Briand et al. [5]. These metrics were defined at class level and are counts of interactions between classes. Their purpose is to measure coupling between classes.
- Marchesi [23]. The aim of these metrics is to measure system complexity, the balance of responsibility between packages and classes, and cohesion and coupling between system entities.
- Harrison et al. [17]. They proposed the Number of Associations per Class metric as an inter-class coupling metric.
- Genero et al. [15] [16]. They defined a set of metrics for structural complexity of UML class diagrams due to the use of UML relationships, such as: associations, generalizations, dependencies and aggregations (see Table 2). The authors suppose these metrics to be good indicators of UML class diagram maintainability characteristics.
- Bansiya et al. [2] [3]. These metrics were defined at class level for assessing design properties such as encapsulation, coupling, cohesion, composition and inheritance.

Table 3 shows whether there are published studies relating to the theoretical and empirical validation of the abovementioned

1. Given the huge number of metrics that can be applied to UML class diagrams at a conceptual level it is impossible to list all of them here, so we have included only a sample of them. Further details regarding definition and validity can be found in the original papers.

SOURCE	VALIDATION				TOOL
	EMPIRICAL		THEORETICAL		
	Experiments	Case Studies	Property-Based Approaches	Measurement Theory Based- Approaches	
Chidamber and Kemerer [12]	YES	YES	YES	YES	YES
Li and Henry [20]		YES			YES
Brito e Abreu and Carapuça [8]		YES		YES	YES
Lorenz and Kidd [22]		YES			YES
Briand et al. [5]		YES	YES		YES
Marchesi [23]		YES			YES
Harrison et al. [17]		YES			
Bansiya et al. [2] [3]		YES			YES
Genero et al. [15] [16]	YES		YES	YES	YES

Table 3: Summary of Proposals for UML Class Diagram Metrics.

proposals for metrics. The last column indicates if any automatic calculation tool is included.

4 Metrics for UML Statechart Diagrams

Metrics for UML statechart diagrams are scant. Derr [13] defined the number of states and the number of transitions as metrics to measure the complexity of OMT (Object Modelling Technique) statechart diagrams (although they can also be applied to UML.)

Carbone and Santucci [11] have proposed two metrics: totSta(c) (total number of states for a class), and totAction(c) (total number of actions for a class). These and other metrics are used for class diagrams, use case diagrams, etc. in order to determine the total complexity of an OO system.

However neither Carbone and Santucci's nor Derr's proposals have gone beyond mere definition.

The work of Miranda et al. [25] is perhaps the most complete. Based on the hypothesis that the size and structural complexity of UML statechart diagrams may influence their understandability (and therefore their maintainability), they defined a set of metrics for the structural complexity and size of UML statechart diagrams.

As size metrics they defined:

- NEntryA. The total number of entry actions, i.e. the actions performed each time a state is entered
- NExitA. The total number of exit actions, i.e. the actions performed each time a state is left.
- NA. The total number of activities (do/activity).
- NSS. The total number of states also considering the simple states within the composites states
- NCS. The total number of composite states
- NE. The total number of events.
- NG. The total number of guard conditions.

As structural complexity metrics they defined:

- NT (Number of Transitions). Counts the total number of transitions, considering common transitions (in which source and target states are different), and final transitions, self-transitions (in which source and target states are the same) and internal transitions (transitions within a state responding to an event but without leaving the state).
- CC (McCabe's Cyclomatic Complexity) [24][2]. Defined as |NSS-NT|+2

The proposed metrics were theoretically validated following the framework proposed by Briand et al., concluding that NA, NSS, NCS, NE, NEntryA, NExitA and NG are size metrics, and NT and CC are complexity metrics. The use of DISTANCE framework [28] also ensures that metrics can be used as ratio scale measurement instruments.

Their hypothesis was to some extent empirically corroborated by a controlled experiment and its replication. As a result of all this experimental work, we can conclude that metrics NA, NSS, NG and NT seem to be highly correlated with the understandability of UML statechart diagrams. Nevertheless, these findings must be considered as preliminary.

5 Metrics for OCL Expressions

The only proposal of metrics for OCL expressions is the one put forward by Reynoso et al. [29], who hypothesized that OCL expression structural properties have an impact on the cognitive complexity of modellers, due to the fact that when developers try to understand an OCL expression (considered in our study as a single mental abstraction: a chunk) they apply cognitive techniques, such as 'chunking' and 'tracing'. These techniques are concurrently and synergistically applied in problem solving and are part of the Cognitive Complexity Model (CMM model) defined by Cant et al. [9] [10], which gives a general cognitive theory of software complexity that

2. Although McCabe's Cyclomatic Complexity was originally defined to calculate single module complexity and entire system complexity, we have adapted it to measure the structural complexity of UML statechart diagrams.

Cognitive technique	Common characteristics of each group of OCL concepts	Samples of OCL concepts related to each group
Tracing	OCL concepts which allow an expression to use properties belonging to other classes or interfaces, different from the type used in the contextual instance	Navigation and collection operation, Messaging, parameter whose types are Classifiers defined in the class diagram, etc.
Chunking Group 1	OCL facilities related to the language itself.	Variables defined by *let* expressions, *If* conditional expressions, predefined iterator variables, literals, etc.
Chunking Group 2	OCL concepts related to the contextual instance and some of its properties.	Reference to attributes or operations of the contextual instance, values postfixed with @pre, variables defined through <<definition>> constraints, etc..

Table 4: OCL Concepts Grouped According to Their Cognitive Techniques.

elaborates on the impact of structure on understandability. 'Chunking' involves the recognition of a set of declaration and extraction information, which is remembered as a chunk, whereas 'tracing' involves scanning, either forward or backwards, in order to identify relevant 'chunks'. 'Tracing' has been observed as a fundamental activity in software comprehension. For this reason, they have defined metrics related to these cognitive techniques, grouping them into three major groups of OCL concepts (see Table 4.)

Although OCL expressions can be added to any UML diagram, [29] have begun by defining metrics for OCL expressions that can be used in a UML class diagram.

Examples of metrics related to 'tracing' are (among others):

- Number of Navigated Relationships (NNR). This metric counts the total number of relationships navigated in an expression. If a relationship is navigated twice, for example using different properties of a class or interface, this relationship is counted only once. Whenever an association class is navigated we will consider the association to which the association class is attached.
- Number of Attributes referred through Navigations (NAN). This metric counts the total number of attributes referred through navigations in an expression.
- Number of Navigated Classes (NNC). This metric counts the total number of classes, association classes or interfaces to which an expression navigates. If a class contains a reflexive relation and an expression navigates it, the class will be considered only once in the metric. Also, as a class may be reachable from a starting class/interface by different forms of navigations (i.e. following different relationships), we must consider this situation as a special case: if a class is used in two (or more) different navigations the class is counted only once.

These metrics were also theoretically validated using the framework of Briand et al. [6] [7]. For example, NNR, NNC and NAN metrics are validated as interaction-based coupling metrics. Intuition has it that metrics related to tracing could have more influence on the understandability and modifiability of OCL expressions. But this needs to be demonstrated by empirical studies, and, as yet, there is no such no empirical validation. Nevertheless, the authors are planning a controlled experiment aimed at validating some of the proposed metrics as indicators of the understandability and modifiability of OCL expressions.

6 Conclusions

The main purpose of this paper is to provide a survey of the most important work available on metrics for quality attributes of UML diagrams and OCL expressions. It aims to provide practitioners with an overall view of what has been done in the field and what metrics are available to help them to take decisions in the early phases of OO development. This work should also help researchers obtain a more comprehensive view of the direction that work in UML model measurement is taking. Further details of some of the proposals presented in this paper can be found either in the bibliographical references where the metrics were originally proposed, or in [15].

As this study shows, UML class diagrams have been the subject of the most extensive research from the measurement point of view. Metrics for UML use case diagrams, UML statechart diagrams and OCL expressions have been proposed and, to a lesser extent, validated. Other UML models covering dynamic aspects of OO systems, such as sequence diagrams, activity diagrams, etc., have been largely ignored.

Although the number of metrics that have been proposed for UML diagrams at conceptual level is low compared to the large number defined for code or advanced design, we believe a shift in effort is required, from defining new metrics to investigating their properties and applications in replicated studies. We need to have a better understanding of what metrics are really capturing, whether they are really different, and whether they are useful indicators of external quality attributes such as maintainability, productivity, etc. The need for new measurements will then arise from, and be driven by, the results of such studies.

In this area designers also ask for desirable values for each measure. However, we must be aware that the hard part is to associate the qualifications 'good' and 'bad' to numeric ranges. This makes metrics all the more useful for OO system designers, to help them make better decisions in their design tasks, which is ultimately the most important goal of any worthwhile measurement proposal.

As a final remark we would conclude by saying that clearly the field of quality metrics for UML models needs to mature. We believe that further empirical validation is necessary, in particular by applying metrics to models obtained from real projects, in order to build up a solid body of knowledge about the usefulness of metrics in practical situations.

References

[1] C. Atkinson and T. Kühne. Model-Driven Development: A Metamodeling Foundation. IEEE Software, 20(5), pp. 36– 41, 2003.

[2] J. Bansiya and C. Davis C. A Hierarchical Model for Object-Oriented Design Quality Assessment. IEEE Transactions on Software Engineering, 28(1), pp. 4–17, 2002.

[3] J. Bansiya, L. Etzkorn, C. Davis and W. Li. A Class Cohesion Metric For Object-Oriented Designs, The Journal of Object-Oriented Programming, 11(8), pp. 47–52, 1999.

[4] B. Bernárdez, A. Durán, and M. Genero. An empirical review of use cases metrics for requirements verification. Proceedings of the Software Measurement European Forum (SMEF'04), 2004.

[5] L. Briand, W. Devanbu and W. Melo. An investigation into coupling measures for C++, 19th International Conference on Software Engineering (ICSE 97), Boston, USA, pp. 412–421, 1997.

[6] L. Briand, S. Morasca and V. Basili. Property-Based Software Engineering Measurement. IEEE Transactions on Software Engineering, 22(1), pp. 68–86, 1996.

[7] L. Briand, S. Morasca and V. Basili. Defining and validating measures for object-based high level design. IEEE Transactions on Software Engineering, 25(5), pp. 722–743, 1999.

[8] F. Brito e Abreu and R. Carapuça. Object-Oriented Software Engineering: Measuring and controlling the development process, Proceedings of the 4th Int. Conference on Software Quality, McLean, VA, USA, pp. 3–5, 1994.

[9] S. Cant, B. Henderson-Sellers and R. Jeffery. Application of Cognitive Complexity Metrics to Object-Oriented Programs. Journal of Object-Oriented Programming, 7(4), pp. 52–63, 1994.

[10] S. Cant, R. Jeffery and B. Henderson-Seller. A Conceptual Model of Cognitive Complexity of Elements of the Programming Process. Information and Software Technology, 7, pp. 351–362, 1992.

[11] M. Carbone and G. Santucci. Fast&&Serious: a UML based metric for effort estimation. 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'2002), pp. 35–44, 2002.

[12] S. Chidamber. and C. Kemerer. A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, 20(6), pp. 476–493, 1994.

[13] K. Derr. Applying OMT. SIGS Books, Prentice Hall. New York, 1995.

[14] P. Feldt. Requirements metrics based on use cases. Master's thesis, Department of Communication Systems, Lund Institute of Technology, Lund University, Box 118, S-221 00 Lund, Sweden, 2000.

[15] M. Genero, M. Piattini and C. Calero (Eds.) Metrics For Software Conceptual Models. Imperial College Press, UK, 2004.

[16] M. Genero, M. Piattini and C. Early Measures for UML Class Diagrams. L'Objet, 6(4), Hermes Science Publications, pp. 489–515, 2000.

[17] R. Harrison, S. Counsell and R. Nithi. Coupling Metrics for Object-Oriented Design. 5th International Software Metrics Symposium Metrics, pp. 150–156, 1998.

[18] B. Henderson-Sellers, D. Zowghi, T. Klemola and S. Parasuram. Sizing use cases: How to create a standard metrical approach. 8th Object–Oriented Information Systems, Lecture Notes in Computer Science, 2425, Springer–Verlag, pp. 409–421, 2002.

[19] J. Krogstie, O. Lindland and G. Sindre. Towards a Deeper Understanding of Quality in Requirements Engineering. 7th International Conference on Advanced Information Systems Engineering (CAISE), Jyvaskyla, Finland, June, 82-95, 1995.

[20] W. Li and S. Henry. Object-Oriented metrics that predict maintainability. Journal of Systems and Software, 23(2), pp. 111–122, 1993.

[21] O. Lindland, G. Sindre and A. Solvberg. Understanding Quality in Conceptual Modelling. IEEE Software, 11(2), pp. 42–49, 1994.

[22] M. Lorenz and J. Kidd. Object-Oriented Software Metrics: A Practical Guide. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[23] M. Marchesi. OOA Metrics for the Unified Modeling Language. 2nd Euromicro Conference on Software Maintenance and Reengineering, pp. 67–73, 1998.

[24] McCabe T. A Complexity Measure. IEEE Transactions on Software Engineering. 2(4), pp. 308–320, 1976.

[25] D. Miranda, M. Genero and M. Piattini. Empirical validation of metrics for UML statechart diagrams. Fifth International Conference on Enterprise Information Systems (ICEIS 03), 1, pp. 87-95, 2003.

[26] L. Moody, G. Shanks and P. Darke. Improving the Quality of Entity Relationship Models – Experience in Research and Practice. 17th International Conference on Conceptual Modelling (ER '98), Singapore, pp. 255–276, 1998.

[27] Object Management Group. MDA–The OMG Model Driven Architecture. Available at <http://www.omg.org./mda/>, August 1st, 2002.

[28] G. Poels and G. Dedene. DISTANCE: A Framework for Software Measure Construction. Research Report DTEW9937, Dept. Applied Economics, Katholieke Universiteit Leuven, Belgium, p. 46, 1999.

[29] L. Reynoso, M. Genero and M. Piattini. Measuring OCL Expressions: An Approach Based on Cognitive Techniques. Chapter 5 in "Metrics for Software Conceptual Models" (Eds. Genero M., Piattini M. and Calero C.). Imperial College Press, UK. 2004.

[30] M. Saeki. Embedding Metrics into Information System Development Methods: An Application of Method Engineering Technique. Lecture Notes in Computer Science, 2681, pp. 374–389, 2003.

[31] R. Schuette and T. Rotthowe. The Guidelines of Modelling – An Approach to Enhance the Quality in Information Models. 17th International Conference on Conceptual Modelling (ER '98), Singapore, pp. 240–254, 1998.

