

Quantitative Assessment of UML Dynamic Models

Aline Lúcia Baroni

Universidade Nova de Lisboa (QUASAR/CITI/DI/FCT)
Quinta da Torre, 2829-516, Monte da Caparica – Portugal
+(351) 21 294 85 36 / 10 772

alinebaroni@di.fct.unl.pt

ABSTRACT

In this work we plan to provide a suite of measures to address mainly two problem areas within contemporary object-oriented software measurement: (i) the lack of measures for the early stages of system development, like conceptual modeling and (ii) the lack of measures for dynamic models of an object-oriented system. Our suite of measures is going to be taken from UML 2.0 diagrams. OCL (Object-Constraint Language) and the UML 2.0 meta-model will be used to formalize and contextualize the definitions.

Categories and Subject Descriptors

D.2.2 [Metrics]: Design Tools and Techniques – *object-oriented design methods, state diagrams*

D.2.8 [Metrics]: Product Metrics

D.3.2 [Programming Languages]: Language Classifications – *constraint and logic languages, design languages, object-oriented languages.*

General Terms

Measurement, design, experimentation.

Keywords

UML models (use cases, statecharts, activity, sequence, collaboration, communication), OCL, estimation, object-oriented design metrics, formalization, dynamic models, behavioral models, UML meta-model.

1. INTRODUCTION

Software measurement has become essential to good Software Engineering. In order to improve the quality of a released software product, it is necessary to evaluate quantitatively - through software measurement - its external and internal characteristics.

A large amount of metrics for object-oriented software models and artifacts has been produced and a considerable number of studies has shown that they can help to enhance the software development process, allowing to detect high complexity parts

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ESEC – FSE’05, September 5–9, 2005, Lisbon, Portugal.
Copyright 2005 ACM 1-59593-014-9/05/0009...\$5.00.

[20], to forecast and plan testing effort [18, 29], and to estimate the maintenance effort [5, 22].

In spite of these research efforts, there are still areas that remain largely untouched by object-oriented software measurement research [9].

There are only a few measures for the dynamic aspects of object-oriented software. Most published works on Software Engineering concentrate on the coding activity, focusing on the data and function dimensions of software. They include well-known metrics, like the MOOSE [15] and MOOD [3] sets, which assume the existence of details such as the associations between classes, the inheritance tree or lattice, the message passing structure, the attributes referenced within the class methods and the use of information hiding, polymorphism, etc. Notwithstanding, they do not consider the modelling of dynamic aspects as for instance, the ones presented in statechart diagrams, activity diagrams, sequence diagrams and some new UML 2.0 diagrams [25, 26].

Although industry begs for measurement instruments that can be applied in the early phases of the development process (mainly for early quality control and project budgeting decisions), several published object-oriented software measures can only be used after (high level) system design. As quality indicators and predictors of structural problems, metrics should be available as early as possible in the software life cycle, and not dependant on source code availability. Some exceptions are presented later on.

This work tries to mitigate these problems, by the creation, formalization and validation of a set of metrics that can be collected in the earlier phases of the development life cycle throughout the analysis of dynamic UML diagrams features.

This paper is organized as follows. Section 2 briefly introduces the related work. Section 3 presents the solution we are going to employ. Section 4 discusses the expected results and how this work is continuously evaluated. Section 5 shows some early conclusions.

2. RELATED WORK

In [8] we showed that the majority of the existing metric sets were developed based on the existence of the source code of the application under study. In spite of presenting these sets, we indicated the need of having early estimates. During the past decades, most of the research on the field of metrics was centered on the code and people extracting metrics results could get these values only in the late phases of the development life cycle. Nowadays, we want to obtain earlier estimates, in the requirements, analysis and design phases, for improving software quality and correcting possible anomalies since the beginning of development. This shortcoming was discussed in some relevant conferences in the area [4, 6].

We already showed that earlier measurement is possible, in [2, 8], [10], where we formalized different sets in the literature, initially designed for being code-dependent (with some exceptions), and evaluated them upon class diagrams, using the GOODLY desing language [1] and the UML 1.4 core meta-model.

Since then, a plenty of metric collections for use case diagrams and their textual specifications was created. Karner introduced the use case points [21] to estimate the size and effort of software project. This proposal was validated by Anda et al. [7]. Marchesi proposed four metrics to evaluate complexity [23]. Smith and Feldt described frameworks in which use cases are used to form an effort estimate [17, 28]. Sellers et al. go further, standardizing the use cases textual descriptions to estimate the resources needed to build a product in the requirements phase [19]. The result is a set with twelve metrics. Bernárdez, Durán and Genero performed an empirical evaluation and review of ten metrics-based verification heuristics for use cases [13]. In a more recent work, Saeki measures the modifiability of use case diagrams [27].

For statecharts, a small number of suites were implemented. Derr's research measured the complexity of OMT structural diagrams [16]. Carbone and Santucci provided an estimation of the system size in terms of SLOCs [14]. Miranda et al. payed attention to the complexity and maintainability of projects based upon statechart diagrams [24].

Some drawbacks of the existing works are the complete lack of formal definitions (as it happens in [13, 14, 16, 17, 19, 21, 24]), the lack of computational support ([16, 17, 19, 21, 24, 27, 28]), the subjectivity factors occasionally required in the computation (observed in [14, 21, 23]) and the lack of validation of the set ([16, 17, 19, 23, 27, 28]).

As far as we know, metrics for other types of dynamic diagrams have been completely neglected by the software engineering community.

3. QUANTITATIVE ASSESSMENT OF UML DIAGRAMS

In our work we try to contribute to the measurement community by creating, formalizing and validating metric suites for the UML behavioral models. This is essential to:

- (i) Have a better understanding of what existing metrics are really capturing, whether they are really different, and whether they are useful indicators of quality attributes such as maintainability, productivity, reusability, etc. The need for new metrics arises from the results of analysing and classifying such studies.
- (ii) Help metric designers to make better decisions, which is ultimately the goal of any measurement proposal.
- (iii) Suppress the absence of measurement support for UML-compliant methodologies towards domain analysis. During domain and systems analysis a strong emphasis is put on dynamic aspects. Concepts like uses cases, scenarios, transactions, activities and events are used to discover the relevant domain objects and to model the interactions between them (e.g. in a collaboration diagram).
- (iv) Define the metrics according to what the methodologies offer to their users. For example, when designing a class diagram,

MOOSE and MOOD metrics can be used to improve the quality of structural artifacts. The same is assumed for other conceptual models accessible in the literature. Desirable qualities of behavioural modelling need to be made explicit.

- (v) Analyse to impact of the static evolution of a system in relation to its dynamic evolution. In an object-oriented implementation, the dynamic aspects are somewhat subordinate to the static aspects.
- (vi) Direct some attention to untouched areas. For example, many of the rules that are captured during behavioural modeling are translated into class invariants or into pre-conditions and post-conditions that are attached to specific methods within the class definitions. These types of assertions did not receive the attention of research into object-oriented design and code measurement.

Complementary, the following problems remain:

- (i) *Lack of formality* – the metrics are either informally defined (using natural language) or partially formalized (using a formal language or mathematics) but have their underlying concepts informally defined.
- (ii) *Lack of context* – the metrics do not capture the scope of the requirements of a particular quality model.
- (iii) *Lack of validation* – metrics validation is not performed or is performed with a simple experimentation lacking replication, significant data or statistical analysis.
- (iv) *Lack of open tools* – the available tools implement the metrics internally but without showing how the computations are performed (in this sense we call them “black boxes”: the code for implementing the metrics is hidden from the users).
- (v) *Lack of coverage* – several sets have a large overlap in what concerns quality characteristics (complexity, size, effort, coupling, cohesion), disregarding many of them.

The consequences of these problems are:

- (i) Ambiguous definitions can lead to different interpretations, producing diverged results on the same software artifacts. Conversely, the use of complex formalisms in metrics definitions may not be easily held by practitioners.
- (ii) The lack of context behind the metric definitions can generate sets in which metric results do not employ the same scale (one is a ratio, another is an absolute value, other is a simple count of an attribute, etc.), are not applied to similar entities (e.g.: one is directed to product, other to the process) and which are not fitted into a quality model. Besides, using a well-known context can facilitate the practitioner's understanding of the metrics.
- (iii) Insufficient validations hamper the proof of usefulness of the sets and their widespread acceptance. Application of the metrics on toy examples does not allow generalizing conclusions.
- (iv) The unavailability of open-source tools poses the limitation of not knowing how the results are extracted and if the implementations of the metrics correspond exactly of what they are supposed to be.

- (v) Some design entities have never been measured. Some of them may be irrelevant for some purposes but others are simply not taken in consideration. Typically, only the structural aspects are measured and usually this represents a small part of the code or design.

Our research aims to tackle these problems and their consequences according to the subsequent steps, for each kind of UML behavioral diagram:

1. classify the existing work to elucidate what is extensively studied and what is still missing;
2. for the missing components, analyze what is sensible to be measured and why;
3. using the elements obtained in the previous step, compose a suite of measures/metrics;
4. formalize and contextualize the metrics using OCL upon the UML 2.0 meta-model;
5. test and refine the metric sets using an OCL environment and the formal definitions, and perform theoretical validations (with the aid of Weyuker's properties [30]);
6. carry out significant experiments to empirically validate the metrics, collecting data from several projects and examining the impact of the data under the conducted projects using a white box approach previously demonstrated in [8, 11];
7. scrutinize the results to generalize the conclusions;
8. make the results public, providing the formal metric definitions and the white box environment for their extraction to the metrics community.

Figure 1 illustrates the high spots of our investigation, regarding the metrics sets:

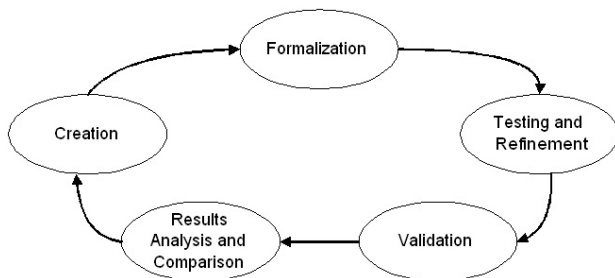


Figure 1. Tasks executed for each UML behavioral diagram.

The ovals in Figure 1 correspond to the steps 3 to 7, starting from the *creation*. The work advances only when the previous step is finished. For instance, the validation is only carried out when the tests and refinements are completed. As soon as each cycle is done, we will publish the results and make them available in our website (<http://ctp.di.fct.unl.pt/QUASAR/>). For the sake of completeness, we can formalize, test and validate also the sets found out in step 1.

4. RESULTS

We expected to accomplish all steps described in the preceding section for all the UML 2.0 behavioral diagrams.

Currently we have done step 1 and we are going to continue advancing with the sequence diagrams.

We plan to evaluate each cycle of our work, publishing the results as soon as they are available and getting the feedback from the community.

One important remark is that we already developed and validated the technique of formalizing metrics definition with OCL, not only under the context of the UML meta-model [8], but also with an ontology for the SQL:2003 standard [12]. The formalizations of these works served as inputs to an OCL evaluator tool, in a white box approach, in which the metrics for UML class diagrams and also object-relational database schemas could be extracted.

5. CONCLUSIONS

In this paper we have outlined the strategy for our research by identifying the current niches found in the behavioral modelling measurement area. We showed both the limitations of some current proposals and how we can collaborate to mitigate them.

Our work is still in its initial phase but we have already demonstrated the validity of the techniques we are willing to use, like the use of OCL as a language to formalize metric definitions and the availability of a tool that can interpret these definitions and automating the metrics collection.

We have also searched for related work and we are currently working on a framework to classify and compare these works, allowing us to identify the areas where research still needs to evolve. In [9] we published the current state of the art.

We are starting the tasks of Figure 1, for the UML sequence diagrams because, as far as we know, there are no works concerning them. We are dependant on the emergence of tool support for UML 2.0 to explore the new features and diagrams of this notation.

6. SUPERVISION

This PhD work is advised by Professor Fernando Brito e Abreu (fba@di.fct.unl.pt) and Professor Pedro Guerreiro (pg@di.fct.unl.pt), and takes place at the QUASAR Research Group (New University of Lisbon).

7. ACKNOWLEDGMENT

Our thanks to FCT/MCES (Fundação para a Ciência e a Tecnologia: <http://www.fct.mces.pt/>) for partly supporting this work.

8. REFERENCES

- [1] Abreu, F.B., Ochoa, L.M. and Goulão, M.A. The GOODLY Design Language for MOOD2 Metrics Collection. In *Proceedings of the 3rd International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QUAOOSE'1999)* (Lisboa, Portugal, 1999).
- [2] Abreu, F.B. *Engenharia de Software Orientado a Objectos: uma Aproximação Quantitativa*. PhD Thesis, Instituto

Superior Técnico - Universidade Técnica de Lisboa, Lisboa, Portugal, 2001.

- [3] Abreu, F.B., Metrics for Object Oriented Software Development. In *Proceedings of the 3rd International Conference on Software Quality* (Lake Tahoe, Nevada, EUA, 1993), American Society for Quality, 67-75.
- [4] Abreu, F.B., Henderson-Sellers, B., Piattini, M., Poels, G. and Saharaoui, H. ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering. *ECOOP'01 Workshop Reader*, Springer-Verlag, 2001.
- [5] Abreu, F.B. and Melo, W.U., Evaluating the Impact of Object-Oriented Design on Software Quality. In *Proceedings of the 3rd International Software Metrics Symposium* (Metrics'96) (Berlin, Germany, 1996), IEEE Computer Society.
- [6] Abreu, F.B., Poels, G., Saharaoui, H. and Zuse, H. ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering. *ECOOP'2000 Workshop Reader*, Springer-Verlag, 2000.
- [7] Anda, B., Dreiem, H., Sjoberg, D.I.K. and Jorgensen, M., Estimating Software Development Effort Based on Use Cases - Experiences from Industry. In *Proceedings of the 4th International Conference on the Unified Modeling Language* (UML 2001) (Toronto, Canada, 2001).
- [8] Baroni, A.L. *Formal Definition of Object-Oriented Design Metrics*. Master Thesis, Vrije Universiteit Brussel, Belgium, August, 2002.
- [9] Baroni, A.L., Abreu, F.B. and Guerreiro, P. *The State-of-the Art of UML Design Metrics*. Technical Report, Universidade Nova de Lisboa, Monte da Caparica, 2005.
- [10] Baroni, A.L. and Abreu, F.B., An OCL-Based Formalization of the MOOSE Metric Suite. In *Proceedings of the 7th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering* (QAOOSE'2003) (Darmstadt, Germany, 2003).
- [11] Baroni, A.L., Braz, S. and Abreu, F.B., Using OCL to Formalize Object-Oriented Design Metrics Definitions. In *Proceedings of the 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering* (QAOOSE'2002) (Malaga, Spain, 2002).
- [12] Baroni, A.L., Calero, C., Piattini, M. and Abreu, F.B. A Formal Definition for Object-Relational Database Metrics. In *Proceedings of the 7th International Conference on Enterprise Information Systems* (ICEIS'05) (Miami, USA, 2005).
- [13] Bernárdez, B., Durán, A. and Genero, M. Empirical Evaluation and Review of a Metrics-Based Approach for Use Case Verification. *Journal of Research and Practice in Information Technology* (JRPIT), 36 (4). 247-258.
- [14] Carbone, M. and Santucci, G. Fast&&Serious: a UML Based Metric for Effort Estimation. In *Proceedings of the 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering* (QAOOSE'2002) (Malaga, Spain, 2002).
- [15] Chidamber, S.R. and Kemerer, C.F., MOOSE: Metrics for Object Oriented Software Engineering. In *Proceedings of the Workshop on Processes and Metrics for Object-Oriented Software Development* (OOPSLA'93) (Washington DC, EUA, 1993).
- [16] Derr, K. *Applying OMT*. Prentice Hall International, New York, 1995.
- [17] Feldt, P. *Requirements Metrics Based on Use Cases*. Master Thesis, Department of Communication Systems, Lund Institute of Technology, Lund University, Lund, Sweden, 1999.
- [18] Harrison, W. Using Software Metrics to Allocate Testing Resources. *Journal of Management Information Systems*.
- [19] Henderson-Sellers, B., Zowghi, D., Klemola, T. and Parasuram, S. Sizing Use Cases: How to Create a Standard Metrical Approach. *Lecture Notes in Computer Science*, 2425. 409-421.
- [20] Henry, S. and Selig, C. Predicting Source-Code Complexity at the Design Stage. *IEEE Software*.
- [21] Karner, G. *Metrics for Objectory*. Master Thesis, Linkuping University, Linkuping, 1993.
- [22] Li, W. and Henry, S. Object-Oriented Metrics that Predict Maintainability. *Journal of Systems and Software*, 23 (2). 111-122.
- [23] Marchesi, M., OOA Metrics for the Unified Modeling Language. In *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering* (CSMR'98) (Florence, Italy, 1998), 67-73.
- [24] Miranda, D., Genero, M. and Piattini, M., Empirical Validation of Metrics for UML Statechart Diagrams. In *Proceedings of the 5th International Conference on Enterprise Information Systems* (ICEIS'03) (Angers, France, 2003), 87-95.
- [25] OMG. *UML 2.0 Infrastructure Final Adopted Specification*, Object Management Group Inc., 2003.
- [26] OMG. *Unified Modeling Language: Superstructure - Version 2.0 - Final Adopted Specification*, Object Management Group Inc., 2003.
- [27] Saeki, M. Embedding Metrics into Information System Development Methods: An Application of Method Engineering Technique. *Lecture Notes in Computer Science*, 2681. 374-389.
- [28] Smith, J. The Estimation of Effort Based on Use Cases, *White Paper of Rational Software*, 1999.
- [29] Tang, M.H., Chen, M.H. and Kao, M.H., Investigating Test Effectiveness on Object-Oriented Software - A Case Study. In *Proceedings of the 12th Annual International Software Quality Week* (San Jose / Silicon Valley, California - USA, 1999).
- [30] Weyuker, E.J. *Evaluating Software Complexity Measures*. Technical Report, Courant Institute of Mathematical Sciences, New York, NY, EUA, 1985.