Quantitative Approaches in Object-Oriented Software Engineering

Report on the 11^{th} Workshop QAOOSE at ECOOP 2007

Yann-Gaël Guéhéneuc[1], Christian F.J. Lange[2], Houari A. Sahraoui[1], Giovanni Falcone[3], Michele Lanza[4], Coral Calero[5], and Fernando Brito e Abreu[6]

[1] Department of Computer Science and Operations Research, Université de Montréal — Canada
[2] Software Engineering and Technology Group, Eindhoven University of Technology — The Netherlands
[3] Lehrstuhl für Softwaretechnik Universität Mannheim — Germany
[4] Faculty of informatics, University of Lugano — Switzerland
[5] Department of Computer Science, Escuela Superior de Informatica of the Castilla-La Mancha University — Spain
[6] Department of Computer Science, Lisbon New University — Portugal

Abstract. The QAOOSE 2007 workshop brought together, for half day, researchers working on several aspects related to quantitative evaluation of software artifacts developed with the object-oriented paradigm and related technologies. Ideas and experiences were shared and discussed. This report includes a summary of the technical presentations and subsequent discussions raised by them. Exceptionally this year, one of the founders of the workshop, Horst Zuse, gave a keynote on the Theoretical Foundations of Object-Oriented Measurement. Three out of the four submitted position papers were presented, covering different aspects such as measuring inconsistencies, visualizing metric values, and assessing the subjective quality of systems. In the closing session, the participants discussed open issues and challenges arising from researching in this area and tried to forecast what will be hot research topics in the short and medium terms.

1 Introduction

Measures of software internal attributes have been extensively used to help software managers, customers, and users to characterize, assess, and improve the quality of software products. Many software companies have intensively adopted software measures to increase their understandability of how (and how much) software internal attributes affect the overall software quality. Estimation models based on software measures have successfully been used to perform risk analysis and to assess software maintainability, reusability, and reliability. Although most of known work applies to object-oriented software, it is also desirable to find measures for component-based software (CBS) and

M. Cebulla (Ed.): ECOOP 2007 Workshop Reader, LNCS 4906, pp. 166–170, 2008.

aspect-oriented development, and for web-based software (WBS), model-based development, in general.

Submissions were invited, but not limited, to the following topics, organized in four areas:

- Metrics collection, including support and standards for sharing research hypotheses, data and results; evaluation of metric collection tools; metric values visualization; evolutionary software metrics collection and validation.
- Quality assessment, including Measuring non-functional requirements of OO systems; metric-based reengineering; quantitative assessment of OO analysis/design patterns, frameworks, aspect-oriented systems, agent-based Web services.
- Metrics validation, including meta-level metrics; formal and empirical validation; measurement Theory; validation techniques and their limits.
- Process management, including reliability and rework effort estimates based on design measures; quantitative tracking of OO, web services, and CBS development; empirical studies on the use of measures for process management.

The workshop was specifically scheduled to increase fruitful interactions and discussions. Participants were requested to submit a contribution in advance. Each participant was expected to read the material submitted by the other participants, so that all participants are acquainted with the ideas that exist within the group and that the workshop could be devoted to discussions instead of presentations. After a short welcome session during which participants introduced themselves, Horst Zuse gave a one-hour keynote on the theoretical foundations of object-oriented measurement [1]. Then, three position papers were presented. The position papers are published in the workshop proceedings [2]. Finally, all participants discussed the presented work and future work.

2 Keynote: Horst Zuse

Theoretical concepts behind software measures and software measurement are necessary in order to have a precise qualitative interpretation of the numbers [3]. In this presentation it was shown, that the properties of software measures for imperative and object-oriented measurement are different. While measures for imperative languages very often assume the extensive structure, object-oriented measures do not assume this measurement structure. This shows, that behind the object-oriented concept of software development another paradigm is hidden than behind imperative languages. Software measures reflect these different paradigms.

3 Keynote: Giovanni Falcone

The goal of developing an ordering technique for a large scale software component search engine is mainly comprised of two majors steps. In a first step a search request is defined by a user. Several techniques are used in practice, reaching from the simple definition of keywords toward the definition of interfaces a particular component should fulfill. In general, the similarity of the components to the given search request are calculated and used as a primary step in the ordering of the components in the results list. In

the literature several solutions for measuring a similarity are given. However, depending on the level of abstraction a given request is made, two extremes are found:

- The result set is comprised of components where all of them have a different similarity and the primary ordering technique is sufficient.
- The result set is comprised of one large subset where all of the entities within have the same similarity measure.

Even if in practice the abstract view on the constitution of the result set lies somewhere in between, it shows that the primary ordering based on a similarity measure is not sufficient. By mainly using an interface driven driven search, where an interface is used as basis for the search request, the first step of ordering the results is given by measuring the conformance of the functional properties of the components to the ones of the defined interface. In a second step non-functional properties need to be taken into account, where software measures seem to be the most valuable ones. Therefore a set of software measures have been calculated for more than 3 million Java based source code components and the results have been further investigated.

In the literature criticism that several software measures show a correlation to other measures is found. In general, the results described in the literature are mainly based on small to medium sized projects. For the purpose of building a second level ordering technique in a software component search we further investigated the behavior of the correlation between some basic software measures.

We presented an overview of the correlation coefficients of a set of about 30 software measures and have shown that a strong correlation between mostly all of the complexity measures investigated has been found (LOC, cyclomatic complexity, number of statements, number of executable statements, number of branch statements, and the number of methods) using the Brevais-Pearson as well as the Spearman rank correlation coefficient. The same results have been found for all of the measures of the Halstead suite where the Breavis-Pearson correlation coefficient was slightly below the Spearman correlation coefficient, except the Effort measure indicating a very weak correlation for the Breavis-Pearson turning in a strong correlation for the rank based correlation.

Based on the presented results, further steps of analyzing the data have been discussed comprising the dimensional reduction using a principal component and a clustering analysis based on the principal components. Next steps have be identified, one of them including a mapping of responsibilities to the clusters found.

4 Position Papers

4.1 Paper: Inconsistencies of Metrics in C++ Standard Template Library

Authors: *by Zoltán Porkoláb, Ádám Sipos, and Norbert* Pataki. Since McCabe's cyclometric measure, structural complexity have been playing an important role measuring the complexity of programs. Complexity metrics are used to achieve more maintainable code with the least bugs possible. C++ Standard Template Library (STL) is the most popular library based on the generic programming paradigm. This paradigm allows implementation of algorithms and containers in an abstract way to ensure the configurability and collaboration of the abstract components. STL is widely used in industrial

softwares because STL's appropriate application decreases the complexity of the code significantly. Many new potential errors arise by the usage of the generic programming paradigm, including invalid iterators, notation of functors, etc. In this position paper, the authors present many complexity inconsistencies in the application of STL that a precise metric must take into account, but the existing measures ignore the characteristics of STL.

4.2 Paper: Automatic Generation of Strategies for Visual Anomaly Detection

Authors: *Salima Hassaine, Karim Dhambri, Houari Sahraoui, and Pierre Poulin.* An important subset of design anomalies is difficult to detect automatically in the code because of the required knowledge. Fortunately, software visualization offers an efficient and flexible tool to inspect software data searching for such anomalies. However, as maintainers typically do not have a background in visualization, they often must seek assistance from visualization expert. This position paper proposes an approach based on taxonomies of low-level analytic tasks, interactive tasks, and perceptual rules to design an assistant that helps analysts to effectively use a visualization tool to accomplish detection tasks.

4.3 Paper: Perception and Reality: What Are Design Patterns Good for?

Authors: *Foutse Khomh and Yann-Gaël Guéhéneuc* This position paper presents a study of the impact of design patterns [4] on quality attributes. An empirical study was performed by asking respondents their evaluations of the impact of all design patterns on several quality attributes. Additionally, detailed results for three design patterns (Abstract Factory, Composite, and Flyweight) and three quality attributes (reusability, understandability, and expendability) were presented. The authors reported on a Null hypothesis test and concluded that, contrary to popular beliefs, design patterns do not always improve reusability and understandability, but that they do improve expendability.

5 Discussions

The informal discussions focused on the following subjects:

- Semantics of measures and how to take semantics of programs into account.
- In metrics visualization, mapping between graphical attributes and some measures, such as DIT.
- Study of community-based preferences that go against "common" sense and validation.
- Taking into account participants' experience and the languages used.

Future directions of research. The participants concluded on the need to perform an extensive survey of the literature on the use of metrics and how validation evolve over time. The agreed to divide the work among the participants to create tooling to identify the relevant papers (in PDF) and extract automatically pertinent information from these papers. This work is currently being pursued.

References

1. Zuse, H.: A Framework of Software Measurement. Walter de Gruyter, Berlin (1998)
2. e Abreu, F.B., Calero, C., Guéhéneuc, Y.G., Lange, C.F.J., Lanza, M., Sahraoui, H.A., Cebulla, M. (eds.): QAOOSE 2007. Proceedings of the 11the ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering Forschungsberichte der Fakultät IV – Elektrotechnik und Informatik. Technische Universität Berlin (2007)
3. Zuse, H.: Foundations of object-oriented software measures. In: METRICS 1996. Proceedings of the 3rd International Symposium on Software Metrics, p. 75. IEEE Computer Society Press, Los Alamitos (1996)
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of reusable Object-Oriented Software. Addison-Wesley, London (1994)