Quantitative Approaches in Object-Oriented Software Engineering

Report on the WS QAOOSE at ECOOP'06

Fernando Brito e Abreu[1], Coral Calero[2], Yann-Gaël Guéhéneuc[3], Michele Lanza[4], and Houari Sahraoui[5]

[1] Univ. of Lisbon, Portugal
[2] Univ. of Castilla, Spain
[3] Univ. of Montreal, Canada
[4] Univ. of Lugano, Switzerland
[5] Univ. of Montreal, Canada

Abstract. The QAOOSE 2006 workshop brought together, for a full 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. 12 out of 14 submitted position papers were presented, covering different aspects such as metrics, components, aspects and visualization, evolution, quality models and refactorings. In the closing session the participants were able to discuss open issues and challenges arising from researching in this area, and they also tried to forecast which will be the hot topics for research in the short to medium term.

1 Historical Background and Motivation

QAOOSE 2006 is a direct continuation of nine successful workshops, held at previous editions of ECOOP in Glasgow (2005), Oslo (2004), Darmstadt (2003), Malaga (2002), Budapest (2001), Cannes (2000), Lisbon (1999), Brussels (1998) and Aarhus (1995).

The QAOOSE series of workshops has attracted participants from both academia and industry that are involved / interested in the application of quantitative methods in object-oriented software engineering research and practice. Quantitative approaches in the OO field is a broad but active research area that aims at the development and/or evaluation of methods, techniques, tools and practical guidelines to improve the quality of software products and the efficiency and effectiveness of software processes. The workshop is open to other technologies related to OO such as aspects (AOP), component-based systems (CBS), web-based systems (WBS) and agent-based systems (ABS). The relevant research topics are diverse, but include a strong focus on applying empirical software engineering techniques.

2 Workshop Overview

23 people attended the workshop. They were representing 18 different organizations from 10 different countries.

Among the attendants, 6 people were not authors, as it is normally the case in these kind of workshops. They have asked the organizers to attend the workshop, which is an additional evidence of the interest raised by this area.

This workshop encompassed 4 sessions, the first 3 bring presentation sessions, while the last one was a pure discussion session. The topics of each presentation session were, respectively: (1) Metrics, Components, Aspects (chaired by H. A. Sahraoui), (2) Visualization, Evolution (chaired by F. Brito e Abreu), and (3) Quality Models, Metrics, Detection, Refactoring (chaired by C. Calero.

Each presentation, plus corresponding discussion, took around 25 minutes. Those presentations were based on submitted position papers that went through an evaluation process conducted by the organizers.

In the final session, a collective effort was performed to discuss open issues that rose from the three previous sessions and to identify future trends for this workshop.

In the next three sections (one per session), we will present for each discussed paper an abstract and a summary of the consequent discussion.

2.1 Session 1: Metrics, Components, Aspects

"Measuring the Complexity of Aspect-Oriented Programs with Multiparadigm Metric" - N. Pataki, A. Sipos, Z. Porkoláb

The position of the authors, presented by Norbert, is that nowadays multiparadigm metrics are necessary to compare programs written using different paradigms or to measure programs that use multiple paradigms. They defined a complexity metric, called AV, that can be extracted from programs written in AOP, OOP or procedural programs. They use this metric to compare the complexity of design patterns when implemented in Java or in AspectJ. The conclusion was that thanks to this multiparadigm metric, we can claim that AOP is well suited for some patterns, but not for others, i.e., increases the complexity in comparison with the OO version. The discussion that took place after the presentation concerned two points. First, a participant asked whether it is interesting to adapt a paradigm specific metric to other paradigms or to define new ones. The position of Norbert is that existing metrics are paradigm dependent. The adaptation can be biased by this dependence. for the second point, a participant asked if the metric is defined using a model that contains the important concepts (from the concerned paradigms) involved in the measurement. Norbert explained that this was the basis of the work. The paper presents the part that concerns AOP.

“On the Influence of Practitioners’ Expertise in Component Based Software Reviews” - M. Goulão, F. Brito e Abreu

Fernando presented his (and his colleague) position on the importance of expertise in component code inspection. This position is supported by an empirical study performed using student subjects. Expertise was determined based on their independently assessed academic record. Inspection outcome was evaluated by the diversity of defects found at two levels of abstraction. As a result, a significant correlation was found between the two variables. Some participants questioned the fact that the expertise was determined based on the student academic results in general and not on components specifically. Although Fernando recognized that this is an issue, he explained that they consider only software engineering results which attenuates the impact. Another possible threat that was discussed is the influence of the domain knowledge of the inspected program on the results. This threat is in fact circumvented by the training on the control domain (application domain) that was given to the subjects. Finally one participant wondered if the age of the expertise is important. As all the students have recent expertise, it is difficult to evaluate the importance of this factor.

“A Substitution Model for Software Components” - B. George, R. Fleurquin, and S. Sadou

The position of the authors, as reported by Bart, concerned the problem of searching components in libraries. Indeed, in the context of CBSD, a software component is described by its functional and non-functional properties. Then, the problem is to know which component satisfies a specific need in a specific composition context, during software design or maintenance. The position is that this is a substitution problem in any of the two cases. From this statement, they propose a need-aware substitution model that takes into account functional and non-functional properties. Following this presentation, a participant asked for a clarification on if the matching concerns at the same time the functional and non-functional properties of the components. Bart explained that the non-functional matching is evaluated only if there is a functional matching. The search strategy was also discussed to see if it is better to compare concrete component between them than to compare each component with an ideal one. The conclusion was that as the search is done in a specific context, direct comparison is not suitable. Many questions concerned the definition and the measurement of component quality. More specifically, a participant questioned the use of ISO9126 to define a single metric for each property. Some properties can be better measured using more than one metric. Another participant highlighted the fact that the quality of a component is different from the quality of the system after the composition, i.e., finding the best component using local properties doesnt mean that this will lead to the best option from the system point of view. Bart however, minimized the impact of the locality by the argument that the composition is performed sequentially. Finally, some participant warned the presenter on the risks of using expertise-base weighting and on the need of evaluating the approach on real component libraries.

2.2 Session 2: Visualization, Evolution

Four position papers were presented and discussed in this session. All of them dealt with software visualization and/or software evolution aspects.

"Towards Task-Oriented Modeling Using UML" - C.F.J. Lange, M.A.M. Wijns, M.R.V. Chaudron

The authors claim that since software engineering is becoming increasingly model-centric, tasks such as the analysis or prediction of system properties require additional information, namely of quantitative nature, that must be easily visualized along with the diagrams. For this purpose they have developed a prototype tool, named MetricView Evolution tool, that allows 2D views of UML diagrams superimposed with a 3rd dimension for representing metrics values. Industrial case studies and a light-weight user experiment to validate the usefulness of the proposed views were reported. The presentation ended with a tool demo.

Several issues were raised in the discussion that followed the presentation. Questioned on the forecasted improvements on the demonstrated tool, Christian referred that there is an undergoing work with an industrial partner regarding tool usability and integration with other UML tools. Another issue pointed out was that of the ability to scale up to large systems. Christian recognized that they are indeed having problems of this sort for systems above a few hundred classes. However, he claimed that this problem can be somehow mitigated with the appropriate usage of model partitioning (packaging) and zooming. Another participant questioned which was the model input format used for the tool, since it has no editing capabilities. Since the answer was XMI, a question was then raised on the positioning information of model elements, which is only available on the new XMI 2 version. Regarding this issue, Christian argued that they are using Rose and Together tools, which generate extended (non-standard) XMI files with positioning information. Finally, someone questioned if the new views causes, or not, an increased navigation difficulty. While recognizing that UML 2 has already 13 diagram types and that adding more information may actually reduce the navigability, Christian argued that integrating / tracing information in the diagrams is a way of mitigating this problem.

"Animation Coherence in Representing Software Evolution" - G. Langelier, H.A. Sahraoui, and P. Poulin

The authors start by recognizing that the study of software evolution requires the analysis of large amounts of data, which is not easily tractable in a manual fashion. To mitigate this problem they propose a semi-automatic approach based on diachronic visualization to represent software versions. Animation is used to represent the transitions (code modifications) between versions. This representation allows the user to perceive visually the coherence between two successive versions. The presentation included a demo of a 3D city-like class visualization tool, where evolution situations provoked (re)placements plus characteristics modifications (e.g. building twist or height modification).

After the presentation, the participants were able to raise several questions. Both the presenter and another co-author also present (Houari Sahraoui) answered them. The first question related to the representation (in the tool) of a typical basic refactoring operation class renaming. The authors replied that they did not consider it, because they want to reduce automatic processing by just allowing the user to detect the movement. Nevertheless this could be performed by matching new classes (appearing) with discarded ones, by simply comparing their contents (renamed classes keep the same features). Another participant asked if some kind of patterns of modification were identified. Guillaume answered that indeed several patterns such as the already mentioned appear/disappear pattern, have been identified, but are still being systematized. A final question tackled the suitability of visual attributes for evolution analysis. The presenter mentioned that if we take the psychological perspective, some attributes such as texture or colour can in fact be much less appropriate than, for instance, twist. However, further investigation on this issue must still be performed.

"Computing Ripple Effect for Object Oriented Software" - H. Bilal and S. Black

The work presented aims at proposing a quantification of the ripple effect, that is, the impact that a local change causes on the remaining parts of a software system. Keeping this effect low is typically a maintenance desideratum. The authors propose to calculate the ripple effect based upon the effect that a change to a single variable has on the rest of a program and consider its applicability as a software complexity measure for object oriented software. Extensions to the PhD work of Sue Black (2001) are proposed to the computation of ripple effect to accommodate different aspects of the object oriented paradigm.

As in previous presentations, we had a period for questions after the oral presentation. The presenter was first asked to prognosticate the deployment of this ripple effect detection in industry. He was not able to produce such a prognosis because, at the current stage of their research and tool support, they cannot yet scale-up to real-world examples. Another participant asked what kind of ripple detection model the authors are using. Haider replied that they have not yet developed a ripple estimation model for object-oriented software. They are planning to do so while extending a locally developed tool for calculating ripple effect for C++ code (currently it only supports C programs).

"Using Coupling Metrics for Change Impact Analysis in Object-Oriented Systems" - M.K. Abdi, H. Lounis, and H.A. Sahraoui

The authors start by recalling that maintenance costs are usually much larger than development ones, and as such, systems modifications and their effects should be assessed rigorously. They propose an analytical and experimental approach to evaluate and predict change impacts in object-oriented systems. This approach uses a meta-model based impact calculation technique. They presented the conclusions of an empirical study, upon a real system, in which a correlation hypothesis between coupling and change impact was evaluated, using three machine-learning algorithms. This session ended with a final period for questions. A participant remarked that the authors apparently have only considered one granularity level the class one which was corroborated by Houari. Then, the question was raised on the justification for the absence of other granularity levels. Houari replied that considering other levels like package, subsystem or system will, in the end, imply, in this case, computing coupling values at class level.

2.3 Session 3: Quality Models, Metrics, Detection, Refactoring

"A Maintainability Analysis of the Code Produced by an EJBs Automatic Generator" - I. García, M. Polo, M. Piattini

The paper presents the tool the authors have built for the automatic generation of multilayer web components- based applications to manage databases. The goal is to deal with the problem of the decrease of its quality and maintainability of web applications due to the successive changes on the code and databases. The source code of these applications is automatically generated by the tool, being optimized, corrected and already pre-tested and standardized according to a set of code templates. The paper makes an overview of the code generation process and, then, shows some quantitative analysis related to the obtained code, that are useful to evaluate its maintainability. Someone on the audience asked about the ability of the tool for the detection of the errors of the database. This issue is not considered yet but will be. The authors were also asked about the prediction of the complexity of the program based on the complexity of the database and they answered that this was considered. Also from a question the authors explained that the business logic is considered in the solution.

"Validation of a Standard- and Metric-Based Software Quality Model" - R. Lincke and W. Löwe

This paper describes the layout of a project of the authors. The objective was to validate the automated assessment of the internal quality of software according to the ISO 9126 quality model. In selected real world projects, automatically derived quality metric values shall be compared with expert opinions and information from bug and test databases. As a side effect, the authors create a knowledge base containing precise and reusable definitions of automatically assessable metrics and their mapping to factors and criteria of the ISO quality model. After the exposition, and after a question, the authors explained that the approach was proven on different languages. Also was remarked the fact that the separation of models automated and manual is strange because ones need the others. About the state of the quality model, the authors explained that they were improving it. The authors also clarified that the metrics on their proposal are calculated on the source code. Finally, somen one suggested to the authors that to consider the semantics of the languages when translating programs in different languages the values of the metrics are calculated but perhaps they are capturing wrong information.

"A Proposal of a Probabilistic Framework for Web- Based Applications Quality" - G. Malak, H.A. Sahraoui, L. Badri and M. Badri

In this work, the authors try to introduce into the web-based applications quality some key issues inherent to this field such as causality, uncertainty and subjectivity. They propose a framework for assessing Web-based applications quality by using a probabilistic approach. The approach uses a model including most factors related to the evaluation of Web-based applications quality. A methodology regrouping these factors, integrating and extending various existing works in this field is proposed. A tool supporting the assessment methodology is developed. Some preliminary results are reported to demonstrate the effectiveness of the proposed model. During the discussion the authors explained that they use the uncertainty (that is different to weighting) because the result is not affected even if some errors appear into the probabilities. The authors were asked about some criteria of the proposal that are not automated and the problems derived from this fact. Authors think that there are some aspects on the web that are very difficult to automate. However, the important is not to automate but the time needed to do the calculation. In any case, they assume that if we want to measure a big amount of web sites, the subjective measures must be avoided.

"Investigating Refactoring Impact Through a Wider View of Software" - M. Lopez, N. Habra

On this work, the authors work about refactoring and the fact that the activity of refactoring is practiced by many software developers and when it is applied well, refactoring should improve the maintainability of software. To investigate this assumption, they propose a wider view of the software, which includes the different wellknown artifacts (requirements, design, source code, tests) and their relationships. This wider view helps analyzing the impact of a given refactoring on software quality. In this study, authors analyze the impact of the refactoring Replace Conditional with Polymorphsm by using this wider view of software. And, at the light of this global view of software, it is more difficult to accept that the analyzed refactoring Replace Conditional with Polymorphsm improves well the maintainability of software. Unfortunately the authors were not able to come to the session and there was no discussion.

"Relative Thresholds: Case Study to Incorporate Metrics in the Detection of Bad Smells" - Y. Crespo, C. López, and R. Marticorena

In order to detect flaws, bad smells, etc, quantitative methods: metrics or measures are usually used. It is common in practice to use thresholds setting the correctness of the measures. Most of the current tools use absolute values. Nevertheless, there is a certain concern about threshold applications on obtained values. Current work tries to accomplish case studies about thresholds on several products and different versions. By other side, product domain and size could also affect the results. The authors tackle if it is correct to use absolute vs. relative thresholds, seeing that effects could have in metric collection and bad smell detection. In the questions time the danger of using threshold values as a first step an after made them relative was discussed. The fact that if you use a threshold value wrong for a system, then the relative threshold will be worst, was remarked. So, some suggestion on the use of the use of data for calculating the threshold accompanied by some techniques as fuzzy logic, probability models, etc were suggested although the authors thought that their approach is also correct.

2.4 Session 4: Discussion

In this session five (5) open questions were selected and discussed. This section summarizes the discussions question by question.

1. Is it interesting to conduct empirical studies to prove obvious hypotheses about software quality? The majority of the participants agreed on the following aspects:
   - It is always worth to replicate previous studies that proved a particular hypothesis although replication studies are still not well considered in our community.
   - Studying obvious hypotheses can be a good mean to train students.
   - An obvious hypothesis can be transformed into a less obvious hypothesis if we target a particular context. For example, it seems obvious that design patterns improve the quality of object-oriented programs. However, is it really obvious that they improve a specific quality characteristic like performance?
   - The evaluation of an obvious hypothesis can be interesting is the results of the study are not a simple Yes/No answer. Detailed discussions of the results, treats to validity, etc. can be very valuable.
   - There is a gap between taught skills and required/performed skills in industry. What can seem obvious in academia may be not trivial in an industrial context.
   - In conclusion, proving obvious hypotheses can be interesting if well motivated, discussed and documented.
2. How to deal with threshold values in general and for anomaly detection in particular? The majority of the participants agreed on the following aspects:
   - Threshold values have a relative impact if the goal is not to decide automatically what is good and what is bad but just what is suspicious. Indeed, they can help to order suspected elements to prioritise the workload (for example testing activities).
   - We need to have explicit thresholds to explain analysis results to developers in industrial context, i.e., to have clear criteria. In particular, anti-pattern detection is hard to explain without clearly defined threshold values.

   - To determine threshold values, two approaches are used. The first deals with expert knowledge or consensuses (practical approach). The second abstracts threshold values starting from a representative set of data. We need to define representative samples with respect to given sets of particular data, using an adequate sampling technique, when possible.
   - In any case, we need to take into account the used technologies, company-wide standards, etc.
3. Generic model and uniform metrics, is it enough/possible? The majority of the participants agreed on the following aspects:
   - It is important to define metrics independently prom the programming language.
   - Achieving independence from programming languages means that any intermediate representation must take into account the operation semantic of each targeted language. In other words, Durant the mapping of a program to the intermediate representation, we must interpret syntactic constructs such as inheritance according to the used lookup algorithm.
   - An additional direction that seems promising is the consideration of domain ontologies to derive metric definitions automatically.
   - Finally, traceability can be also an avenue as it bridges the gap between models and source code.
4. Visualisation: what could be the paradigms to visualise data together? The majority of the participants agreed on the following aspects
   - It is important to go beyond tables andor aggregation of metrics (averages, summations, etc.) to visualize and exploit metrics. Visualization technique must allow displaying metric values for a large set of elements.
   - It is important to find tradeoffs between visualizing little information on large systems and visualizing lot of information on small systems. To this respect, cognitive science and scientific data visualisation are two domains were we can find inspiration.
   - We must find a balance between contextual information that is supposed to be present all the time and elements related to a specific problem that must be in first plan.
   - We have to find the balance also between offering customization capabilities and overriding the users with too many options.
   - Using metaphors is a promising direction for data visualization and navigation.
5. Beyond evaluation, how to provide feedback to improve programs? The majority of the participants agreed on the following aspects:
   - It is important to relate design/code anomalies (e.g., suspicious classes) and opportunities of corrections (e.g., refactorings). This can be done using metric definitions or general software engineering principles.
   - It is also important to consider metric-based quality evaluation as a day-to-day tool rather then a one-shot tool in an evaluation phase. After such an evaluation phase, it is generally too late. The cost of corrections is too high.

3 QAOOSE 2006 Participants

Organizers

Name	Affiliation
Fernando Brito e Abreu	Univ. of Lisbon, Portugal
Coral Calero	Univ. of Castilla, Spain
Yann-Gaël Guéhéneuc	Univ. of Montreal, Canada
Houari Sahraoui	Univ. of Montreal, Canada

Other Participants

Name	Affiliation
Regis Fleurquin	Univ. de Bretagne Sud, France
Carlos Lopez	Univ. Nova de Lisboa, Portugal
Guillaume Langelier	Univ. de Montréal, Canada
Rüdiger Lincke	Växjö Univ., Sweden
Macario Polo	Univ. Castilla-La Mancha, Spain
Haider Bilal	London South Bank Univ., UK
Javier Perez	Univ. de Valladolid, Spain
Gabriela Arevalo	LIRMM - Univ. de Montpellier, France
Thomas Fritz	Univ. of British Columbia, Canada
Jaqueline McQuillan	NUI Maynooth, Ireland
Norbert Pataki	Eötuös Lorand Univ. Budapest, Hungary
Christian Lange	TU Eindhoven, The Netherlands
Raul Marticorena	Univ. of Burgos, Spain
Yann Prieto	Ecole Centrale de Nantes, France
Cédric Bardet	SNCF, France
Naouel Moha	Univ. of Montreal, Canada
Christine Havart	SNCF, France
Olivier Beaurepaire	SNCF, France
Susanne Jucknath	TU Berlin, Germany