STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

GOD CLASS DESIGN FLAW DETECTION IN OBJECT ORIENTED DESIGN. A CASE STUDY

CAMELIA ŞERBAN

Abstract. In this article we present an experimental evaluation of our proposed methodology for object-oriented design (OOD) assessment introduced in previous work. A comparison with a related approach based on detection strategies is comprised in the proposed case study. It also discusses the limitations of existing approaches found in literature in the context of OOD assessment.

1. Introduction

Object-oriented systems that undergo repeated addition of functionality commonly suffer a loss of quality in their underlying design. A small change in one part of it may have unforeseen effects in completely other parts, leading to potential disasters. In order to prevent this, we aim to maintain a high quality design throughout the system lifecycle. This goal can be attained by repeated evaluation of the system design, with the goal of early identification of those design entities that are not conformable to rules, principles and practices of a good design [2].

As a result in this direction software metrics have been brought to the attention of many researchers. They measure different aspects of software and therefore play an important role in *understandig, controlling and improving* software quality.

Several approaches [2, 4, 5, 6] were found in literature that address the problem of metrics based assessment for OOD. However, these approaches encounter some limitations: i) how to set proper threshold values for metrics is not addressed; ii) they lack a standard terminology and formalism in order to define the contextual background which can also serve for metrics definition. To mitigate these limitations, a *Conceptual Framework for OOD Assessment (CFDA)* has been introduced in our previous work [1]. This paper presents

Received by the editors: October 10, 2011.

2000 *Mathematics Subject Classification.* 68N30, 68T37.

1998 *CR Categories and Descriptors.* code D.2.8 [**Software Enginnering**]: *Metrics – Product metrics*; code D.1.5 [**Pattern recognition**]: *Clustering –Algorithms.*

Key words and phrases. Software metrics, object oriented design, fuzzy clustering.

an experimental evaluation of this methodology for object-oriented design assessment.

The paper is organized as follows: Section 2 briefly describes our previous work. The proposed experimental evaluation is presented in Section 3. To emphasize the advantages of the CFDA, Section 4 presents a comparison with a related approach based on detection strategies [2]. Finally, Section 5 summarizes the contributions of this work and outlines directions for further research.

2. A conceptual framework for OOD Assessment

In our previous work [1] we have proposed a quantitative evaluation methodology for object-oriented design. The proposed methodology, based on static analysis of the source code, is described by a conceptual framework which has four layers of abstraction:

The first layer, *Object-Oriented Design Meta-Model*, formally defines *the domain of the assessment* $D(S) = (E, Prop(E), Rel(E))$ of an OOD corresponding to a software system S; this meta-model describes the design entities that are evaluated E, their properties $Prop(E)$ and the relationships between them $Rel(E)$.

The second layer, *Formal Definitions of OOD Metrics*, consists of a library of OOD metrics definitions. Metrics are formally defined using the context delineated for the meta-model presented by first layer and expressing them in terms of algebraic sets and relations, knowledge assumed as familiar since the first stages of our studies.

The third layer, *Specifications of the Assessment Objectives*, specifies in a formal manner the assessment objectives using a metrics based approach.

The last layer, *Mesurement Results Analysis*, uses the fuzzy clustering analysis method to interpret the measurement results obtained in the assessment process. This method overcome the limitations of the existing approaches which use threshold values for metrics. The selected algorithm, Fuzzy Divisive Hierarchic Clustering (FDHC) [3] produces a binary tree hierarchy that provides an in-depth analysis of the data set, by deciding on the optimal subcluster cardinality and the optimal cluster substructure of the data set.

3. Proposed Case-Study

In this section, we describe the steps needed to be performed to apply the proposed evaluation framework on an open source application, namely *log4net* [7]. It consists of 214 classes grouped in 10 packages.

Domain Assessment Identification. We parse the source code of *log4net* application and produce the domain of the assessment, $D(log4net) =$

$(E, Prop(E), Rel(E))$, i.e the design entities, their properties and the relations between them.

Setting the Assessment Objectives. The objective of the proposed assessment is to identify those classes AE (assessed design entities) from the *log4net* application affected by "God Class" [8] design flaw. In order to attain this goal we proceed as follows: *i*) a set of design principles, heuristics or rules DP are related to "God Class" design flaw, defining a flaws–principles graph $FPG = (DF, DP, G_{DF \mapsto DP} \subseteq DF \times DP)$, $DF = \{God\ Class\}$; *ii*) the elements of DP are than related to a set of software metrics M which quantify these principles, obtaining the principles–metrics graph $PMG = (DP, M, G_{DP \mapsto M} \subseteq DP \times M)$.

Therefore, the 3-tuple $AO(log4net) = (AE, FPG, PMG)$ represents *the assessment objectives specification* regarding the evaluation of the object oriented design $D(log4net) = (E, Prop(E), Rel(E))$ corresponding to *log4net* application.

An instance of a God Class performs most of the operations, delegates only minor details to a set of trivial classes, and uses the data from other classes. This design flaw violates the principle of *manageable complexity*, as god classes tend to capture more than one abstraction. Consequently, such pathological classes tend to be also *non-cohesive*.

The selected heuristics [8] related to God Class design flaw are: *i*) distribute system intelligence horizontally as uniformly as possible (it refers to *class complexity*); *ii*) beware of classes with much non-communicative behavior (it refers to intraclass communication – *low cohesion* principle); *iii*) beware of classes that access directly data from other classes. Thus, the selected metrics to quantify these heuristics are: Weighted Method per Class (WMC) [9], Tight Class Cohesion (TCC) [10], Access to Foreign Data (ATFD) [2].

To add more clarity for the above mentioned statements, Figure 1 presents the specification of the assessment objectives.

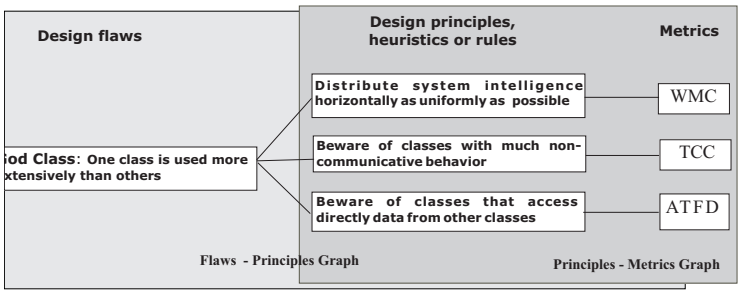

FIGURE 1. Specification of the assessment objectives.

Formal Definition of Selected Metrics. Due to space restriction, we present here only the formal definition of ATFD [2] metric. ATFD represents the number of external classes from which a given class accesses attributes, directly or via accessor-methods. Its formal definition being:

$$ATFD(c) = card(A - B - C), c \in Class(E) \ where :$$

$A = \{d \in Class(E) | \exists m_0 \in Meth(c) : ARefRel(m_0) \cap Attr(d) \neq \emptyset\}$,
$B = \{d \in Class(E) | \exists c_{i_0}, c_{i_1}, ..., c_{i_{l_i}} \in Class(E) : c_{i_0} = c, c_{i_{l_i}} = d, (c_{i_j}, c_{i_{j+1}}) \in IRel_{c_{i_j}}, j = \overline{0, l_i - 1}\}$,
$C = \{d \in Class(E) | d \in Inner(c))\}$.

Measurement Results Analysis. The results of the assessment are a set of design entities that were evaluated AE, in this case the set of classes from *log4net* application, together with their corresponding values of the selected metrics WMC, TCC, $ATFD$. Based on the metrics values, we will select from AE the "*suspect*" entities (those classess affected by "God Class" design flaw).

Following a classical approach we have to set thresholds values for metrics that we use. In order to overcome this limitation, we propose an alternative approach, based on fuzzy clustering analysis. Thus, an entity may be placed in more that one group, having different membership degrees.

The optimal fuzzy partition, $U = \{1.1.1, 1.1.2, 1.2.1, 1.2.2, 2.1, 2.2\}$ obtained by applying the FDHC [3] algorithm contains six clusters. From each of them we have eliminated some entities whose metrics values are very different from the majority of entities of this cluster. We name these entities "isolated data points", they are considered for further analysis.

We consider that clusters with "suspect" entities have to simultaneously meet the following two conditions: *i*) the member's average values for the WMC and ATFD metrics are greater than the average values of the entire set of analysed design entities; *ii*) the member's average value for the TCC metric is lower than the average value computed on the entire set of analysed design entities.

Taking into account the above mentioned criteria, clusters 1.1.1, 2.1, 2.2 have been identified as containing possible *suspect* entities.

Regarding the set of isolated points we can conclude the following: *i*) The entities with id 16, 26, 44, 158, 175, 180 are not considered *suspect* entities. They have low values for the WMC and ATFD metrics and they are cohesive. *ii*) One entity with class id 123 is complex but cohesive and with low values of ATFD metric, thus is not considered *suspect* entity. *iii*) The classes with the id 33, 36, 39, 42, 52, 73, 170 are considered *suspect* entities, some of them being cohesive but very complex and with high values for ATFD and some being less cohesive and lacking complexity but also having high values for ATFD.

4. A Comparative Analysis

Marinescu [2] defined metric-based detection strategies for capturing design flaws of object-oriented design. In the definition of detection strategy he used threshold values for metrics. The current section makes a comparison of our approach based on fuzzy analysis with that proposed by Marinescu regarding the identification of "God Class" design flaw. The detection strategy defined by Marinescu is presented in what follows.
$GodClasses := ((ATFD, TopValues(20\%)) \wedge (ATFD, HigherThan(4))) \wedge ((WMC, HigherThan(20)) \vee (TCC, LowerThan(0.33))$.

We have applied the above mentioned detection strategy on the same data set of measurement results. In this respect, we will make a comparison beetwen the two sets of *suspect* entities: the list of entities identified as *suspect* using the approach proposed by us, and the list of entities identified as *suspect* using the approach proposed by Marinescu. The conclusions are presented in what follows.

From the 30 entities which are identified as suspects by the approach of Marinescu, only two are not also identified using our approach, entities 94 and 151. These entities are classes which are non-cohesive and access external data but have a relative low complexity in comparison to the other suspect entities.

From the 46 entities identified by our approach, 18 were not identified using the detection strategy based approach. The reason why these were not considered as suspects by the approach of Marinescu is because of their low ATFD value (lower than 4). Let us analyse two entities identified as suspects by our approach: One with classId=13 and metric values WMC=19, TCC=0.07 and ATFD=4 and the other with classId=3 and metric values WMC=44, TCC=0.07 and ATFD=3. The first one is considered suspect by the detection strategy based approach and the second one is not. We ask ourselves if the difference of 1 on the ATFD metric is relevant when deciding between the two, which one is suspect and which one isn't. We do observe that the two entities have the same value for TCC but the WMC value is much higher for the second entity. Also, the membership degree of the second entity to the cluster of suspect entities is much higher than the one of the first entity.

All the above mentioned statements argue that our approach overcome the limitations which the approaches based on the establishement of thresholds values for metrics have encountered. We recall here these limitations: they are *inflexible*, the selected entities have metrics values that belong to a strictly defined interval, and they are *opaque*, providing information only about the suspect entities and they do not provide an overview of all entities in the

system, thus we do not know if a suspect entity is not very similar to many more other entities who have not passed the threshold values.

5. Conclusion and Future Work

We have presented in this paper a case-study to experimentally validate our proposed methodology for object-oriented design assessment [1]. The case-study addressed the issue of "God Class" design flaws detection. The approach is based on metrics and on fuzzy clustering analysis method. To highlight the advantages of using our model, a comparative study with a similar approach which uses threshold values for metrics has been made.

As one of our further work we aim to propose other comparisons with similar approaches regarding OOD assessment based on metrics.

References

[1] Camelia Serban, *A Conceptual Framework for Object-oriented Design Assessment*, Computer Modeling and Simulation, UKSim Fourth European Modelling Symposium on Computer Modelling and Simulation, 90–95, 2010.

[2] R. Marinescu, *Measurement and quality in object-oriented design*, Ph.D. thesis in the Faculty of Automatics and Computer Science of the Politehnica University of Timisoara, 2003.

[3] Dumitrescu, D., *Hierarchical pattern classification*, Fuzzy Sets and Systems 28, 145–162, 1988.

[4] S. Mazeiar, Li. Shimin, and T. Ladan. *A Metric-Based Heuristic Framework to Detect Object-Oriented Design Flaws* Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC06), 2006.

[5] P.F. Mihancea and R.Marinescu. *Towards the optimization of automatic detection of design flaws in object-oriented software systems*, In Proc. of the 9th European Conf. on Software Maintenance and Reengineering, 92-101, 2005.

[6] L. Tahvildari and K. Kontogiannis. *Improving design quality using meta-pattern transformations : A metric-based approach*, Journal of Software Maintenance and Evolution: Research and Practice, 16, 331-361, 2004.

[7] Open source project: log4net, http://logging.apache.org/log4net.

[8] A.J. Riel. *Object-Oriented Design Heuristics*, Addison-Wesley, 1996.

[9] S. Chidamber and C. Kemerer, *A metric suite for object- oriented design*, IEEE Transactions on Software Engineering, 20(6), 476–493, 1994.

[10] J.M. Bieman and B.K. Kang, *Cohesion and Reuse in an Object-Oriented System*, ACM Symposium on Software Reusability, 1995.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

E-mail address: camelia@cs.ubbcluj.ro