STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVIII**, Number 4, 2013

SHOTGUN SURGERY DESIGN FLAW DETECTION. A CASE-STUDY

CAMELIA ŞERBAN

Abstract. Due to the complexity of object oriented design, its assessment becomes a time-consuming activity. Consequently, methods and techniques are needed in order to assess the design in an *automatic manner*. As a result, software metrics represent a solution, a means for quantifying those aspects considered important for the assessment. They measure different aspects of software and therefore play an important role in *understandig*, *controlling* and *improving* software quality.

This article presents an experimental evaluation of our proposed framework for object-oriented design (OOD) assessment. To emphasize the relevance of this methodology, a comparison with similar approaches found in literature is also comprised in our case study.

1. Introduction

Object-oriented systems going through recurrent additions of functionality commonly suffer a loss of quality in their underlying design [6]. A minor change in one of its parts may have unpredictable effects over other parts and may generate possible disasters. Therefore, object-orientation should be basically flexible and easily adaptable to extending the functionality of a system, with limited alteration to existing modules [7].

To fulfill this goal, assessment of the software system design should observe well established design principles and heuristics, and should be continuing through the entire development life cycle. The evaluation results will further serve in identifying those design entities that need further investigation and possible refactorings.

Due to the complexity of object oriented design, its assessment becomes a time-consuming activity. Consequently, we need methods and techniques to

Received by the editors: October 10, 2013.

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.

assess the design in an *automatic manner*. As a result, software metrics represent a solution, a means for quantifying those aspects considered important for the assessment. They measure different aspects of software and therefore play an important role in *understandig*, *controlling* and *improving* software quality.

Several approaches [1, 2, 3, 4] 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 [8]. This paper aims to highlight the relevance of our results, presenting an experimental evaluation of the proposed 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 [1]. Finally, Section 5 summarizes the contributions of this work and outlines directions for further research.

2. Proposed methodology for object-oriented design assessment

In our previous work [8] 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: *Object-Oriented Design Model*, *Formal Definitions of OOD Metrics*, *Specifications of the Assessment Objectives* and *Mesurement Results Analysis*.

(1) **Object-Oriented Design Model.** The first layer formally defines *the domain of the assessment* $D(S) = (E, Prop(E), Rel(E))$ of an OOD corresponding to a software system S; this model describes the design entities E that are evaluated, their properties $Prop(E)$ and the relationships between them $Rel(E)$.
(2) **Formal Definitions of OOD Metrics.** The second layer consists of a library of OOD metrics definitions. Metrics are formally defined using the context delineated for the 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.

(3) **Specifications of the Assessment Objectives.** The third layer specifies in a formal manner the assessment objectives using a metrics based approach.
(4) **Mesurement Results Analysis.** The last layer 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) [9] 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.

Corresponding to these layers, the proposed framework will be associated with a 4-tuple, *CFDA=(OOD-Model, Metrics-Definition, Assessment-Objectives, Results-Analysis)*, where the acronym *CFDA* stands from **C**onceptual **F**ramework for **D**esign **A**ssessment.

Figure 1 presents the elements of CFDA, emphasizing their interactions. The proposed conceptual framework contains a predefined and customizable catalogue of software metrics. The definition of these metrics are written using the language which describes the OOD Model. The assessment objectives are specified using a predefined catalogue of design heuristics, rules and principles, and a set of design flaws. The Results Analysis gathers data from the three components of CFDA framework in order to establish the assessment results.

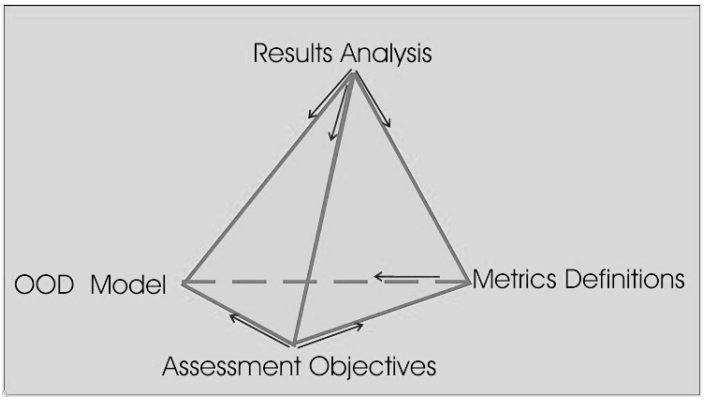

FIGURE 1. The four layers of OOD Assessment Framework

3. Shotgun Surgery design flaw detection. 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* [11]. It consists of 214 classes grouped in 10 packages.

3.1. **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 (E), their properties ($Prop(E)$) and the relations between them ($Rel(E)$). These elements define a model for object oriented design [1] and they are described in a formal manner in our previous work [8].

3.2. **Setting the Assessment Objectives.** The objective of the proposed assessment is to identify those classes $AE \subset E$ (AE - assessed design entities) from the *log4net* application affected by "Shotgun Surgery" [10] design flaw. In order to attain this goal we proceed as follows:

- a set of design principles, heuristics or rules DP are related to "Shotgun Surgery" design flaw, defining a flaws–principles graph $FPG = (DF, DP, G_{DF \mapsto DP})$,$G_{DF \mapsto DP} \subseteq DF \times DP$, $DF = \{ShotgunSurgery\}$. FPG represents the second element of the assessment objectives specification definition, introduced in [8]; so, we aim at this step to establish what are those design principles, heuristics or rules which could better define the DP set.
- the elements of DP are then related to a set of software metrics M which quantify these principles, obtaining the principles–metrics graph $PMG = (DP, M, G_{DP \mapsto M})$, where $G_{DP \mapsto M} \subseteq DP \times M$.

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.

In order to identify the elements of the assessment objectives specification components, $AO(log4net) = (AE, FPG, PMG)$, we analyse in detail the "Shotgun Surgery" design flaw.

A class that is coupled to a large number of other classes and that produces a large number of changes throughout the system in case of an internal change, can be considered a possible suspect of Shotgun Surgery design flaw [10]. Shotgun Surgery means that a change on a given class entails many changes to a lot of different classes. In brief, this design flaw approaches the issue of "strong implementation coupling" [1]. It is difficult to spot changes spreading over many places. Therefore, an excessive coupling has a negative

effect on many external quality attributes like *reusability*, *maintainability* and *testability*.

The "open-closed principle" [12] is one of the most important principles of object oriented design. According to this, software entities (classes, modules, functions, etc.) "*should be open for extension, but closed for modification*". It is correlated with the principle of "low coupling". They are both concurrent in stating that those design entities that could propagate a lot of changes in the system when something was changed inside them, have to be identified and reviewed.

Another important design heuristic, related to *Shotgun Surgery* design flaw may be "*Minimize the Number of Messages in the protocol of a class*" [5].

In order to identify metrics which capture the meaning of this design flaw, further analysis of these principles and heuristics are needed. In this respect, Marinescu [1] have identified three potential "victims" of changes in a class:

- *methods directly accessing an attribute that has been changed*;
- *methods calling for a method whose signature has been changed*;
- *methods which override a method whose signature has been changed.*

Therefore, two metrics are selected [1] to quantify the above mentioned aspects. These metrics are briefly described below:

- Changing Methods (CM) [1] is defined as the number of distinct methods in the system that would be potentially affected by changes operated in the measured class. The methods potentially affected are all those that access an attribute and/or call a method and/or redefine a method of the given class.
- Changing Classes (CC) [1] is defined as the number of client-classes where the changes must be operated as the result of a change in the server-class.

To add more clarity for the above mentioned statements, Figure 2 presents the specification of the assessment objectives.

3.3. **Formal definitions of the selected metrics.** The metrics selected to quantify the "Shotgun Surgery" design flaw, CM, CC, are formally defined using the context delineated for the model presented by first layer of CFDA [8]. Their definitions are expressed in terms of algebraic sets and relations.

3.4. **Measurement Results Analysis.** The results of the assessment are a set of design entities that were evaluated $AE = AE_{Shotgun\ Surgery} = Class(E)$, in this case the set of classes from *log4net* application, together with their corresponding values of the selected metrics $M_{Shotgun\ Surgery} = \{CM, CC\}$. Based on the metrics values, we will select from AE the "*suspect*" entities (those classess affected by "Shotgun Surgery" design flaw).

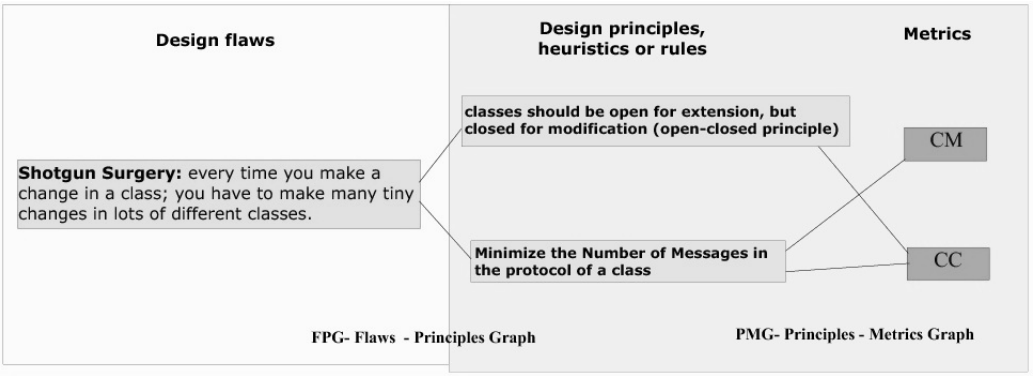

FIGURE 2. Specification of the assessment objectives.

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.

3.4.1. *Fuzzy partitions determination.* Applying the $FDHC$ algorithm, using as input data the set of classes from *log4net* application, each class being identified with a vector of metrics values, we obtain an optimal fuzzy partition, described in Table 1. In what follows, this partition will be denoted through $U_{AE_{Shotgun\ Surgery},M_{Shotgun\ Surgery}} = \{1.1.1, 1.1.2, 1.2.1, 1.2.2, 2.1, 2.2\}$

From each cluster of the partition $U_{AE_{Shotgun\ Surgery},M_{Shotgun\ Surgery}}$, we determined the list of isolated data points. In the case of "Shotgun Surgery" design flaw the list of entities considered isolated data points are as follows:

- 72, 183, 199 (from cluster 1.1), 63, 70, 71, 73, 170, 210 (from cluster 1.2); these entities have the CC metric value equal to 1;
- 6, 19, 24, 74, 82, 101, 112, 122, 131, 139, 172 (from cluster 2.1); these entities have the CC metric value greater than 1, the rest of entities having the CC metric value equal to 1;
- 48, 78 (from cluster 2.2.1); these entities have metrics values very disimilar with the rest of entities from this cluster; they have CM=5 and CC=3, while the rest of entities have CC=1;
- from cluster 2.2.2 no isolated data points were identified.

The entities considered isolated points are removed from the clusters and considered for further analysis. The partition obtained after the elimination of isolated data points is denoted as $U'_{AE_{Shotgun\ Surgery},M_{Shotgun\ Surgery}}$ and the cardinality of this partition is 192.

Cluster	Members (ClassId)	No. of members
1.1	1 183 8 72 199	5
1.2	2 20 63 71 73 91 92 135 170 210 64 70 127 194	14
2.1	3 19 21 24 37 57 74 112 126 139 140 181 213 42 52 75 163 173 175 176 179 189 192 7 13 17 28 31 40 49 60 87 101 115 123 131 171 182 184 188 204 206 68 69 162 197 201 4 39 146 174 190 202 6 33 82 122 172	58
2.2.1	14 30 36 43 48 78 141 169 180 200 15 26 34 41 45 55 59 77 85 132 142 205 209 5 9 12 16 18 25 27 29 35 44 47 56 61 65 67 84 86 88 94 116 128 130 151 158 177 185 186 193 195 198 203 208	55
2.2.2	22 32 58 62 76 79 81 107 114 118 124 125 133 137 138 147 148 149 154 167 178 187 191 196 207 10 11 23 38 46 50 51 53 54 66 80 83 89 90 93 95 96 97 98 99 100 102 103 104 105 106 108 109 110 111 113 117 119 120 121 129 134 136 143 144 145 150 152 153 155 156 157 159 160 161 164 165 166 168 211 212 214	82
	Total number of entities	214

TABLE 1. Optimal fuzzy partition of a set of 214 class objects.

3.4.2. *Criteria used for "suspect" design entities identification.* All clusters have been examined in order to identify which of them contain classes that are suspects of the "Shotgun Surgery" design flaw. As we have discussed earlier, a possible suspect will have *high* values for the CM and CC metrics. Thus, to avoid the problem of setting up the software metrics threshold values in order to establish which are the clusters that contain "suspects" entities, we consider that a cluster with suspect entities has to meet the following condition: the member's average values for the CM and CC metrics are greater than the average values of the entire set of the analyzed design entities.

Table 2 presents the average values for the CM and CC metrics, computed for each cluster of the fuzzy partition $U'_{AE_{Shotgun\ Surgery},M_{Shotgun\ Surgery}} = \{1.1, 1.2, 2.1, 2.2.1, 2.2.2\}$, partition obtained after the elimination of the isolated data points. Thus, taking into account the above mentioned criterion, clusters 1.1 and 1.2 have been identified as containing possible suspect entities.

Cluster	Avrg(CM)	Avrg(CC)
1.1	94.0	1.35
1.2	40.63	5.75
2.1	10.98	1
2.2.1	3.36	1.07
2.2.2	0.3	0.30
Class(E)	8.52	1.17

TABLE 2. The average values of the CM and CC metrics, computed for each cluster. Clusters 1.1 and 1.2 are marked as containing suspect entities. $Class(E)$ represents the set of all classes from the analized system.

Regarding the set of isolated points which were identified earlier, we can conclude the following:

- The entities with id 63, 70, 71, 72, 73, 170, 183, 199, 210 are not taken into account for further analysis. They have the value of CC metric equals to 1.
- The entities with id 6, 19, 24, 48, 74, 78, 82, 101, 112, 122, 131, 139, 172 are considered suspect entities and they are selected from the cluster 2.1, 2.2.1.

In conclusion, the entities considered for further analysis are those classes with the value of CC metric greater than 1. Their cardinality is 23 out of 214 design entities which were analyzed. They are presented in Figure 3.

The following statement argues the decision in the establishement of final suspects list: if, by changes, two classes affect n methods, the class that spreads its changes over more classes is worse than the one that spreads them all in one class. Thus, the final list of suspect entities contains classes with the value of CC metric greater than 3 (see Figure 3.

4. Related Approaches Comparison

The current section aims to present a comparative study between our approach on the identification of design entities affected by "Shotgun Surgery" design flaw and the similar approach proposed by Marinescu [1] which is based on setting the threshold values for metrics used. To identify those classes from the system, affected by the above mentioned design flaw, Marinescu introduced so called "detection strategy" defined by by the formula 1.

$$(1) \qquad ShotgunSurgery(c) = \begin{cases} 1, & ((CM \in TopValues(20\%))and(CM \geq 10))and(CC \geq 5) \\ 0, & else \end{cases}$$

ClassId	CM	CC	1.1.	1.2.	2.1.	2.2.1.	2.2.2.
8	65	2	0.8806	0.0607	0.0335	0.0132	0.0119
19	26	2	0.0334	0.4276	0.3775	0.0917	0.0698
20	35	2	0.0093	0.8036	0.1201	0.0368	0.0302
64	53	2	0.3653	0.6296	0.0030	0.0011	0.0010
127	54	2	0.4220	0.5702	0.0046	0.0017	0.0015
131	7	2	0.0011	0.0046	0.7715	0.1763	0.0464
135	28	2	0.0315	0.5162	0.3092	0.0805	0.0625
139	24	2	0.0329	0.3432	0.4494	0.1002	0.0743
194	51	2	0.2598	0.7385	0.0010	0.0004	0.0003
6	11	3	0.0060	0.0288	0.9315	0.0227	0.0110
24	16	3	0.0165	0.0993	0.7496	0.0828	0.0518
48	5	3	0.0006	0.0023	0.3942	0.5027	0.1002
74	24	3	0.0329	0.3458	0.4471	0.1000	0.0742
78	5	3	0.0006	0.0023	0.3942	0.5027	0.1002
101	6	3	0.0009	0.0037	0.5822	0.3324	0.0809
122	10	3	0.0045	0.0208	0.9451	0.0204	0.0091
172	9	4	0.0038	0.0170	0.8812	0.0687	0.0294
2	29	6	0.0314	0.5665	0.2702	0.0739	0.0580
112	18	6	0.0236	0.1566	0.6390	0.1080	0.0728
82	13	7	0.0131	0.0673	0.7466	0.1078	0.0651
91	40	12	0.0590	0.8353	0.0649	0.0221	0.0187
92	35	18	0.0951	0.6506	0.1558	0.0535	0.0450
1	123	25	0.5093	0.2080	0.1513	0.0675	0.0639

FIGURE 3. The list of "Shotgun Surgery" suspect entities considered for further analysis.

Applying the detection strategy defined before, we can conclude that:

- all entities identified as suspects by the approach of Marinescu are also in our list of suspects;
- one entity with classId 172 identified by our approach, was not identified using the detection strategy based approach. The reason why this was not considered as suspect by the approach of Marinescu is because of its CC metric value (lower than 5).

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 [8]. The case-study addressed the issue of "Shotgun Surgery" 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] 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.

[2] 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.

[3] 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.

[4] 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.

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

[6] M. O'Keeffe and M. Cinnide: Search-based refactoring: an empirical study, *Journal of Software Maintenance and Evolution: Research and Practice*, 20, 345–364, 2008.

[7] A. Chatzigeorgiou and G. Stephanides: Entropy as a Measure of Object-Oriented Design Quality, *1st Balkan Conference on Informatics (BCI'2003)*, 21–23, 2003.

[8] C. 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.

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

[10] M. Fowler and K. Beck and J. Brant and W. Opdyke and and D. Roberts: *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999.

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

[12] R. Martin. Design Principles and Patterns: http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf, 2006.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

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