A CASE STUDY ON STRUCTURAL CHARACTERISTICS OF OBJECT-ORIENTED DESIGN AND ITS STABILITY

Mahmoud O. Elish
Department of Computer Science
George Mason University
Fairfax, VA 22030-4400, USA
melish@gmu.edu

ABSTRACT

Design structural stability refers to the extent to which the structure of a design is preserved throughout the evolution of the software from one release to the next. This paper investigates whether there are some structural characteristics (metrics) of object-oriented design that are indicators of its structural stability. Investigated metrics are related to size, inheritance, cohesion, and coupling. Design structural stability was assessed from each software release to the next using two metrics: a class-based metric and a relationship-based metric. As a case study, measures were collected from 13 successive releases of Apache Ant.

KEY WORDS

Design structural stability, software evolution, software metrics, object-oriented designs.

1. Introduction

Software maintenance is inevitable if software systems need to remain useful in their environments. Changes are necessary to continue increasing or sustaining the value of software as it evolves over time. A well designed software system should be able to accommodate these changes without requiring changes to its structure as much as possible.

Design structural stability refers to the extent to which the structure of a design is preserved throughout the evolution of the software from one release to the next. In object-oriented software, classes and relationships between them define the design structure, which is depicted by class diagrams.

The availability of adequate metrics as indicators of design structural stability can give software managers early insight into trends in software evolution, and thus assist them in managing and controlling long-lived software systems. According to DeMarco's principle [4]: "you cannot control what you cannot measure."

Pervious research includes the following studies. Jazayeri [8] applied retrospective analyses to successive releases of a large telecommunication software system to evaluate its architectural stability with simple size measures, coupling measures, and color visualization to observe phenomena about the evolution of the software across releases. Bansiya [1] proposed a methodology to assess the stability of framework architectures over successive versions by determining the extent-of-change in the structural characteristics between versions. Grosser et al. [5, 6] proposed a case-based reasoning approach for predicting stability of Java classes. None of the pervious research studies investigated indicators of the structural stability of object-oriented designs.

The objective of this paper is to investigate whether there are some structural characteristics (metrics) of object-oriented design that are indicators of its structural stability. In other words, the paper aims to test for existence of significant correlations between measures of some object-oriented design metrics and measures of two design structural stability metrics over successive releases of a case study system.

The rest of this paper is organized as follows. Section 2 describes the metrics used to assess design structural stability. Section 3 defines the investigated object-oriented design structural characteristics. Section 4 states the hypotheses. Section 5 discusses the case study and its results. Section 6 concludes the paper and gives directions for future work.

2. Measuring Design Structural Stability

Classes and relationships between them are the two most fundamental building blocks of object-oriented designs. Classes are the units of modularity, and relationships between them define the architectural structure of a design.

Different kinds of relationships may exist between classes: generalization, aggregation, dependency, and association. A *generalization* is a relationship between a more general class (superclass) and more specific class (subclass). This relationship can be further classified into implementation inheritance and interface inheritance (a.k.a. realization). An *aggregation* relationship exists between two classes if one is part of the other, i.e. if one is the type of an attribute of the other. A *dependency* relationship exists between two classes if one is the return type of a method of the other or the type of a parameter of a method of the other. An *association* relationship exists between two classes if one invokes one or more methods of the other, and/or references one or more attributes of the other.

Design structural stability in this paper is assessed from each software release to the next. The classes and relationships between them in release *i* are compared with the classes and relationships in release *i+1* (following release) to determine the percentage of classes and relationships that remained stable from one release to the next.

Two design structural stability metrics are used to quantify stability from one release to the next: class-based metric, and relationship-based metric.

2.1. Class-based Metric

The class-based design structural stability (*CDSS*) metric calculates the percentage of classes that were not added, deleted or modified from one release to the next. A class is considered modified if at least one of its lines of code (excluding comment and blank lines) is deleted or modified, or at least one new line of code is added to it. Formally, the *CDSS* metric is calculated from release *i* to release *i+1* (following release) as follows:

$$CDSS_{i\rightarrow i+1} = \frac{(N_i + N_{i+1}) - (AC_{i\rightarrow i+1} + DC_{i\rightarrow i+1} + 2 * MC_{i\rightarrow i+1})}{N_i + N_{i+1}}$$

Where,

N_i	is the number of classes in release *i*
N_{i+1}	is the number of classes in release *i+1*
$AC_{i\rightarrow i+1}$	is the number of added classes between release *i* and release *i+1*
$DC_{i\rightarrow i+1}$	is the number of deleted classes between release *i* and release *i+1*
$MC_{i\rightarrow i+1}$	is the number of modified classes between release *i* and release *i+1*

The number of modified classes, unlike the numbers of added and deleted classes, in the above formula is multiplied by two because modified classes exist in both releases not just in one of them.

2.2. Relationship-based Metric

The relationship-based design structural stability (*RDSS*) metric calculates the percentage of class relationships, including all kinds of relationships, that were not added nor deleted from one release to the next. Formally, the *RDSS* metric is calculated from release *i* to release *i+1* (following release) as follows:

$$RDSS_{i\rightarrow i+1} = \frac{(R_i + R_{i+1}) - (AR_{i\rightarrow i+1} + DR_{i\rightarrow i+1})}{R_i + R_{i+1}}$$

Where,

R_i	is the number of class relationships in release *i*
R_{i+1}	is the number of class relationships in release *i+1*
$AR_{i\rightarrow i+1}$	is the number of added relationships between release *i* and release *i+1*
$DR_{i\rightarrow i+1}$	is the number of deleted relationships between release *i* and release *i+1*

Both *CDSS* and *RDSS* metrics have a range from zero (maximum instability) to one (maximum stability).

3. Design Structural Characteristics

This paper adapts 14 object-oriented design metrics that are believed to capture some important dimensions of design structural characteristics as candidate indicators of its stability. These metrics are based on existing measures in the software metrics literature, such as [2, 3, 7]. The metrics are categorized into four groups: size, inheritance, cohesion, and coupling.

3.1. Size Metrics

- *Number of classes (NC)* – The total number of classes defined in a design.
- *Public classes ratio (PCR)* – The number of public classes to the total number of classes defined in a design.
- *Average number of methods per class (ANM*) – The average number of methods defined in a class.
- *Average number of attributes per class (ANA)* – The average number of attributes defined in a class.
- *Average weighted methods per class (AWMC)* – The average sum of the cyclomatic complexities of all methods in a class.
- *Average response for a class (ARFC)* – The average number of methods in the set of all methods that can be invoked in response to a message sent to an object of a class.

3.2. Inheritance Metrics

- *Abstractness (ABS)* – The number of abstract classes to the total number of classes defined in a design.
- *Reuse ratio (RR)* – The number of superclasses to the total number of classes defined in a design.
- *Specialization ratio (SR)* – The number of subclasses to the total number of superclasses defined in a design.

- *Average depth of inheritance tree per class (ADIT)* – The average depth of a class within its inheritance hierarchy.
- *Average number of children per class (ANOC)* – The average number of direct subclasses of a class.

3.3. Cohesion Metrics

- *Average lack of cohesion in methods per class (ALCOM)* – The average number of different methods within a class that reference a given instance variable.

3.4. Coupling Metrics

- *Average class coupling (ACC)* – The average number of classes that a class is related to through any kind of relationship.
- *Coupling factor (CF)* – The number of class couplings in a design to the maximum number of possible class couplings in a design.

4. Hypotheses

The hypotheses that are examined in this paper are listed in Table 1. For each one of the 14 design metrics described in the pervious section, the hypothesis is that there is a significant correlation between the measure of that design metric in release *i* and the design structural stability from release *i* to release *i+1* (following release) measured by both *CDSS* and *RDSS* metrics.

Table 1. Hypotheses

H1a	Significant correlation between NC_i and $CDSS_{i\rightarrow i+1}$
H1b	Significant correlation between NC_i and $RDSS_{i\rightarrow i+1}$
H2a	Significant correlation between PCR_i and $CDSS_{i\rightarrow i+1}$
H2b	Significant correlation between PCR_i and $RDSS_{i\rightarrow i+1}$
H3a	Significant correlation between ANM_i and $CDSS_{i\rightarrow i+1}$
H3b	Significant correlation between ANM_i and $RDSS_{i\rightarrow i+1}$
H4a	Significant correlation between ANA_i and $CDSS_{i\rightarrow i+1}$
H4b	Significant correlation between ANA_i and $RDSS_{i\rightarrow i+1}$
H5a	Significant correlation between $AWMC_i$ and $CDSS_{i\rightarrow i+1}$
H5b	Significant correlation between $AWMC_i$ and $RDSS_{i\rightarrow i+1}$
H6a	Significant correlation between $ARFC_i$ and $CDSS_{i\rightarrow i+1}$
H6b	Significant correlation between $ARFC_i$ and $RDSS_{i\rightarrow i+1}$
H7a	Significant correlation between ABS_i and $CDSS_{i\rightarrow i+1}$
H7b	Significant correlation between ABS_i and $RDSS_{i\rightarrow i+1}$
H8a	Significant correlation between RR_i and $CDSS_{i\rightarrow i+1}$
H8b	Significant correlation between RR_i and $RDSS_{i\rightarrow i+1}$
H9a	Significant correlation between SR_i and $CDSS_{i\rightarrow i+1}$
H9b	Significant correlation between SR_i and $RDSS_{i\rightarrow i+1}$
H10a	Significant correlation between $ADIT_i$ and $CDSS_{i\rightarrow i+1}$
H10b	Significant correlation between $ADIT_i$ and $RDSS_{i\rightarrow i+1}$
H11a	Significant correlation between $ANOC_i$ and $CDSS_{i\rightarrow i+1}$
H11b	Significant correlation between $ANOC_i$ and $RDSS_{i\rightarrow i+1}$
H12a	Significant correlation between $ALCOM_i$ and $CDSS_{i\rightarrow i+1}$
H12b	Significant correlation between $ALCOM_i$ and $RDSS_{i\rightarrow i+1}$
H13a	Significant correlation between ACC_i and $CDSS_{i\rightarrow i+1}$
H13b	Significant correlation between ACC_i and $RDSS_{i\rightarrow i+1}$
H14a	Significant correlation between CF_i and $CDSS_{i\rightarrow i+1}$
H14b	Significant correlation between CF_i and $RDSS_{i\rightarrow i+1}$

5. Case Study

Apache Ant [9] is the case study system, which is an open source Java-based build tool. Apache Ant software project is more than four years old. Its most recent release (1.6.2) has more than 200K lines of code, and more than 1200 classes.

In order to test the hypotheses, 13 successive releases of Apache Ant were analyzed, from its first release (1.1) to its most recent release (1.6.2). Precisely, the releases are 1.1, 1.2, 1.3, 1.4, 1.4.1, 1.5, 1.5.1, 1.5.2, 1.5.3, 1.5.4, 1.6, 1.6.1, and 1.6.2.

Three tools were used for data collection: JStyle [10] metrics tool, ExamDiff Pro [11] file comparison tool, and a prototype metrics tool that has been developed as part of this research. JStyle was used to collect measures of the 14 design metrics under investigation from each release of Apache Ant. ExamDiff Pro was used to calculate the *CDSS* stability metric from each release to the next by comparing classes between releases. Comment and blank lines were excluded in class comparison. A class was considered modified if at least one of its lines of code was deleted or changed, or at lease one new line of code was added to it. The *RDSS* stability metric was calculated with the developed prototype metrics tool.

5.1. Results

Tables 2 and 3 provide descriptive statistics for the measures of the two stability metrics and the 14 design metrics that were collected from the releases of Apache Ant respectively. Wide variation in the *CDSS* metric measures throughout the evolution of Apache Ant can be observed. In general, class relationships were more stable than classes over the releases of Apache Ant. This can be observed by comparing the measurement results of the *CDSS* metric with the *RDSS* metric.

Table 2. Descriptive statistics of the measures of the two stability metrics

	Minimum	Maximum	Average	Std. Dev.
CDSS	0.0841	0.9819	0.6149	0.3580
RDSS	0.6265	0.9975	0.8683	0.1582

Table 3. Descriptive statistics of the measures of the 14 design metrics

	Minimum	Maximum	Average	Std. Dev.
NC	117	1204	745.08	349.44
PCR	0.7692	0.8630	0.8365	0.0296
ANM	6.74	8.25	7.64	0.36
ANA	2.52	3.59	2.83	0.32
AWMC	12.02	16.18	14.28	1.02
ARFC	19.06	23.34	20.73	1.00
ABS	0.0317	0.0536	0.0474	0.0078
RR	0.0684	0.1064	0.0982	0.0115
SR	7.9789	13.1250	8.8666	1.5269
ADIT	1.70	2.02	1.93	0.09
ANOC	0.85	1.04	0.98	0.07
ALCOM	31.14	47.16	41.03	3.93
ACC	8.74	9.53	8.97	0.28
CF	0.0073	0.0815	0.0198	0.0218

Figure 1 plots the cumulative measures of *CDSS* and *RDSS* stability metrics over the successive releases of Apache Ant. Segments of these line plots with steep slope represent periods of stability throughout the evolution of Apache Ant. The closer the measure of a stability metric to 1.0 between two successive releases the steeper the slope of the line plot segment between these two releases for this measure. For example, a steep slope for the *CDSS* metric measures can be observed between every two successive releases from release 1.5 to release 1.5.4 indicating a period of stability according to the *CDSS* metric. In contrast, periods of instability can be easily observed from release 1.4.1 to release 1.5 and from release 1.5.4 to release 1.6 according to the *CDSS* metric.

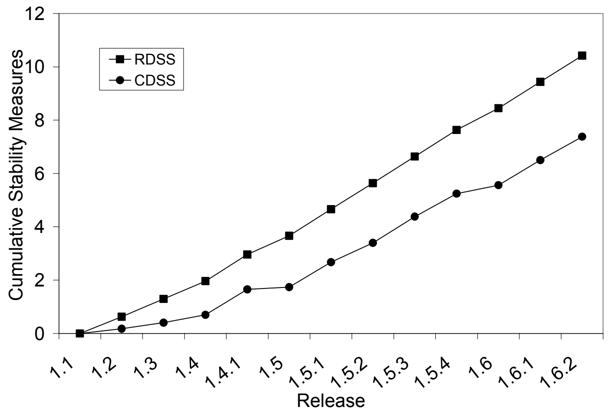

Figure 1. Cumulative measures of *CDSS* and *RDSS* stability metrics over the successive releases of Apache Ant

5.2. Correlation analysis

Correlation analysis was performed at 0.05 level of significance (95% confidence level) to test for existence of significance correlations between measures of each one of the 14 design metrics and measures of both *CDSS* and *RDSS* metrics. Most of the collected measures have skewed distribution, and accordingly Spearman's rank correlation analysis method was used to compute correlation coefficients.

Table 4 reports the correlation coefficients between measures of each one of the 14 design metrics and measures of the *CDSS* metric. Only 7 out of the 14 design metrics (*NC*, *PCR*, *ABS*, *ADIT*, *ANOC*, *ACC*, and *CF*) were found to have significant correlations with the *CDSS* metric.

Table 4 Spearman's correlation coefficients between each of the 14 design metric and the *CDSS* metric

Candidate Indicator	Correlation coefficient	Hypothesis	Conclusion
NC	0.511*	**H1a**	**Accepted**
PCR	0.567*	**H2a**	**Accepted**
ANM	0.126	H3a	Rejected
ANA	-0.322	H4a	Rejected
AWMC	-0.273	H5a	Rejected
ARFC	-0.210	H6a	Rejected
ABS	0.616*	**H7a**	**Accepted**
RR	0.231	H8a	Rejected
SR	-0.371	H9a	Rejected
ADIT	0.588*	**H10a**	**Accepted**
ANOC	0.525*	**H11a**	**Accepted**
ALCOM	0.175	H12a	Rejected
ACC	-0.510*	**H13a**	**Accepted**
CF	-0.545*	**H14a**	**Accepted**

* Correlation is significant at the 0.05 level (1-tailed).

Table 5 reports the correlation coefficients between measures of each one of the 14 design metrics and measures of the *RDSS* metric. Only 6 out of the 14 design metrics (*NC*, *PCR*, *ABS*, *ADIT*, *ACC*, and *CF*) were found to have significant correlations with the *RDSS* metric. The *ANOC* metric is only significantly correlated with the *CDSS* metric, and not with the *RDSS* metric.

Table 5. Spearman's correlation coefficients between each of the 14 design metric and the *RDSS* metric

Candidate Indicator	Correlation coefficient	Hypothesis	Conclusion
NC	0.497*	**H1b**	**Accepted**
PCR	0.574*	**H2b**	**Accepted**
ANM	-0.070	H3b	Rejected
ANA	-0.217	H4b	Rejected
AWMC	-0.378	H5b	Rejected
ARFC	-0.294	H6b	Rejected
ABS	0.630*	**H7b**	**Accepted**
RR	0.154	H8b	Rejected
SR	-0.336	H9b	Rejected
ADIT	0.497*	**H10b**	**Accepted**
ANOC	0.455	H11b	Rejected
ALCOM	0.007	H12b	Rejected
ACC	-0.524*	**H13b**	**Accepted**
CF	-0.524*	**H14b**	**Accepted**

* Correlation is significant at the 0.05 level (1-tailed).

In the set of design metrics that were found having significant correlations with the stability metrics, the coupling metrics (*ACC* and *CF*) are the only design metrics that are negatively correlated with design structural stability, whereas the others are positively correlated.

Cohesion is the only design metric category that was not found to be significantly correlated with any of the two stability metrics. Additional metrics of cohesion will be investigated in future research.

5.3. Analysis of Results

Further analysis of the obtain results from this case study reveals a number of interesting observations. The significant positive correlations between the *NC* metric and both *CDSS* and *RDSS* stability metrics suggest that as the design size increases in terms of the number of classes its structural stability increases.

In addition, all coupling metrics (*ACC* and *CF*) were found to have significant negative correlations with both stability metrics. This result suggests that low coupling between design classes sustains the structural stability of the design.

Moreover, the significant positive correlations between inheritance metrics (*ABS*, *ADIT*, and *ANOC*) and the stability metric(s) suggest that high, and presumable appropriate, utilization of inheritance mechanism leads to more structurally stable design.

6. Conclusions

Structural stability of object-oriented designs refers to the extent to which the structure of a design is preserved throughout the evolution of the software from one release to the next. Classes and relationships between them define the design structure. This paper has investigated 14 object-oriented design metrics related to size, inheritance, cohesion, and coupling as candidate indicators of design structural stability. Stability was assessed from each software release to the next using two metrics: a class-based metric that calculates the percentage of classes that remained stable between two releases, and a relationship-based metric that calculates the percentage of class relationships that remained stable between two releases.

The major results of this case study can be summarized as follows. Only 6 out of the 14 investigated design metrics (*NC*, *PCR*, *ABS*, *ADIT*, *ACC*, and *CF*) were found to have significant correlations with both stability metrics (*CDSS* and *RDSS*). This means that these six design metrics represent good indicators of design structural stability. Releases of Apache Ant whose designs highly utilize the mechanism of inheritance were more structurally stable than those with less utilization of inheritance. In addition, releases whose designs have low class couplings were found to be more structurally stable than those with high class couplings.

As future work, additional case studies are needed to further support the findings of this paper. Confirmed results can then be used to develop predictive models for design structural stability. Another research direction is to utilize the results to develop guidelines for building structurally stable designs.

References

[1] J. Bansiya, "Evaluating Framework Architecture Structural Stability," *ACM Computing Surveys*, vol. 32, 2000.

[2] L. Briand, J. Daly, and J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91-121, Jan/Feb 1999.

[3] S. Chidamber and C. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, June 1994.

[4] T. DeMarco, *Controlling Software Projects: Management, Measurement & Estimation*, Prentice-Hall, 1982.

[5] D. Grosser, H. Sahraoui, and P. Valtchev, "Predicting Software Stability Using Case-Based Reasoning," *Proc. 17th IEEE International Conference on Automated Software Engineering*, pp. 295-298, 2002.

[6] D. Grosser, H. Sahraoui, and P. Valtchev, "An analogy-based approach for predicting design stability of Java classes," *Proc. 9th International Software Metrics Symposium*, pp. 252-262, 2003.

[7] R. Harrison, S. Counsell, and R. Nithi, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," *IEEE Transactions on Software Engineering*, vol. 24, no. 6, pp. 491-496, June 1998.

[8] M. Jazayeri, "On Architectural Stability and Evolution," *Lecture Notes in Computer Science*, *Springer-Verlag*, pp. 13-23, 2002.

[9] Apache Ant Website, http://ant.apache.org/

[10] JStyle Website, http://www.mmsindia.com/jstyle.html

[11] ExamDiff Pro Website, http://www.prestosoft.com/ps.asp?page=edp_examdiffpro