JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2008**

Vol. 7, No. 6, July-August 2008

Revisiting Class Cohesion: An empirical investigation on several systems

Linda Badri, Mourad Badri & Alioune Badara Gueye
Software Engineering Research Laboratory, Department of Mathematics and Computer Science, University of Quebec at Trois-Rivières, Quebec, Canada

Abstract

Class cohesion is considered as one of most important object-oriented software attributes. Cohesion refers to the degree of relatedness between members in a class. High cohesion is a desirable property of classes. Several metrics have been proposed in literature in order to measure class cohesion in object-oriented systems. They capture class cohesion in terms of connections between members within a class. Most of these metrics have been experimented and widely discussed. They do not take into account some characteristics of classes as stated in several papers. We present, in this paper, an extention of the cohesion metric we proposed in a previous work. We introduce a new cohesion criterion based on common objects parameters. Our main goal in this work was: (1) to demonstrate, by analyzing many real systems that the introduced criterion is statistically significant and, (2) to validate our approach for class cohesion assessment by exploring empirically the relationship that may exist between our new cohesion metric and coupling. We developed a cohesion measurement tool for Java programs and performed an empirical study on several systems. The selected test systems vary in size and domain. The obtained results demonstrate that: (1) the new class cohesion metric captures several additional pairs of related methods and (2) there exists a significant correlation between the new cohesion metric and coupling.

1 INTRODUCTION

Software metrics have become an essential tool in software engineering [Som 04, Pre 05]. In the field of software quality, metrics are used for assessing several software attributes such as complexity, coupling, cohesion, and size. They provide, therefore, an important assistance to developers and managers in order to assess and improve software quality during the development process. Class cohesion is considered as one of most important object-oriented software attributes. As stated in [Hen 96], internal cohesion can be best understood as syntactic cohesion evaluated by examining the code of each individual module. It is argued that the modularization can be accomplished for a variety of reasons. There is several types of cohesion: functional cohesion, sequential cohesion, coicidental cohesion, etc. [Hen 96]. We focused, in this work, on functional cohesion. Cohesion refers to the degree of relatedness between members in a component. High cohesion is a desirable property of components. It is widely recognized that highly cohesive components tend to have high maintainability and reusability [Li 93, Bie 95, Bri 97, Cha 00]. The cohesion of a component allows the measurement of its structure quality. The cohesion degree of a component is high, if it implements a single logical function. All the parts of a component must contribute to this implementation.

Yourdon and Constantine introduced cohesion, in the context of traditional applications, as a measure of the extent of the functional relationships of the elements in a module [You 79]. Grady Booch describes high functional cohesion as existing when the elements of a component (such as a class) all work together to provide some well-bounded behaviour [Boo 94]. In the object paradigm, a class is cohesive when its parts are highly correlated. It should be difficult to split a cohesive class. A class with low cohesion has disparate and non-related members. Cohesion can be used to identify the poorly designed classes. Class cohesion is considered as one of most important characteristics in object-oriented design. Cohesion is an underlying goal to continually consider during the design process [Lar 03]. A large number of metrics have been proposed in literature in order to measure class cohesion in object-oriented systems. Major existing cohesion metrics have been presented in detail and are categorized in [Bri 98]. The majority of these metrics are based on attributes usage (sharing) criteria. These metrics capture class cohesion in terms of connections among members within a class. They count the number of instance variables used by methods or the number of methods pairs that share instance variables. Most of these metrics have been experimented and widely discussed in literature [Hen 96, Bas 96, Cha 98, Chi 98, Ema 99, Bri 00, Bad 03]. Several studies have noted that they fail in many situations to properly reflect the cohesiveness of classes [Kab 00, Cha 00, Ama 02]. According to many authors, they do not take into account some characteristics of classes, for example, sizes of cohesive parts as stated in [Ama 02] and connectivity among members as stated in [Cha 00].

Beyond these aspects, we believe that existing metrics fail to reflect properly the properties of class cohesion, particularly in terms of functional relationahips that may exist between methods. They are based on restricted criteria and could lead, as stated in some papers, to unexpected values of cohesion in many situations. We believe that class cohesion should not exclusively be based on common instance variables usage as stated in [Kab 01] and have to go beyond this aspect by considering other relationships between methods. We note that, in many situations, by analyzing the source code of several parts of many real systems, that several methods are functionally related together without sharing any instance variables and can not be separated in different classes. We extend the existing criteria by considering different ways of capturing class cohesion. In a first step, we introduced, in [Bad 03, Bad 04], a new criterion, which focused on interactions between class methods. We performed an experimental study on several Java systems. The obtained results demonstrated that the introduced criterion was statistically significant. It allows capturing a significant number of pairs of related methods, which are not captured by others existing cohesion metrics. Many empirical studies on object-oriented metrics have been conducted these last years. Among others, [Dag 03] focused on the prediction of maintaibility, [Zho 03] and [Agg 06] on the prediction of fautes, and [Kab 01] on exploring the relationship between cohesion and coupling in the one hand and the relationship between cohesion and changeability in the other hand. The domain gains in interest these last years.

Since our last papers [Bad 03, Bad 04] on the cohesion of object-oriented systems, we continued experimenting on cohesion metrics on several other systems. The analysis of the obtained results combined with code analysis of certain applications that we noticed allowed us to determine that there are certain particular situations where class' methods could be functionally connected without sharing instance variables or calling methods in common. This lead us to introduce a complementary criterion for cohesion and experiment on it. In this paper, we propose, on the first hand, an extension of the two metrics DC_D and DC_I that we proposed in [Bad 03, Bad 04]. This extension corresponds to the introduction of a new criterion related to common objects parameters (methods having a same objet as parameter). In fact, two methods of a given class can very well share a same object passed in parameter without being connected by sharing a method or an instance variable. In fact, if we consider existing cohesion metrics, the majority is based on shared instance variables. From the nature of object-oriented systems, these instance variables can be of a primitive type (basic Java types, for example) or be of object type. Furthermore, in the object context, objects collaborate to accomplish a given task. The collaboration between a group of objects, to accomplish a given task, is based on certain design principles (design patterns, among others) and implies the assignment of responsibilities to classes [Lar 03]. This collaboration can be located on two levels: a collaboration between a group of objects belonging to different classes, and a collaboration between a group of methods within a unique given class. This last type of collaboration can be seen, among other things, as the use of objects under the form of instance variables or passed as arguments, at the method level, public in particular. Cohesion may allow, in this context, to insure that assignment of responsibilities to classes is done in a cohesive manner. In this context, and starting from the conclusions we drew in our experiments since 2003, the introduction of the new criterion seemed relevant to us. The obtained results, from the experiment we conducted, confirm our hypotheses. They clearly demonstrate that the extended cohesion metrics, based on the addition of the proposed criterion, capture more pairs of connected methods that the old metrics DC_D and DC_I did. The statistical test we performed was positive for several systems.

On the second hand, we explored, as a first attempt to validate our approach, the relationship that may exist between the extended cohesion metrics and coupling. A well established belief in the software engineering community states that a high cohesion is related to a low coupling, and vice-versa (the yin-yang principle) [Lar 03, Som 04, Pre 05]. However, and to the best of our knowledge, no empirical validation was done on this. A few papers, however, such as [Kab 01], attempted to validate the relationship between coupling and cohesion, but without any success. One of their conclusions called for a refinement of existing cohesion metrics. The experimental study we have performed uses our previous cohesion metrics as well as the new extended ones. The obtained results demonstrate that there exists a significant correlation between our cohesion metrics and the considered coupling metric (CBO, Coupling Between Objects) of Chidamber et al. [Chi 91, Chi 94]. The considered cohesion metrics present, however, different correlation degrees. The empirical investigation as well as the obtained results are discussed in Section 7. Our ultimate objective, which will be the subject of futur work, is to validate the proposed cohesion metrics as, for example, good indicators for changeability and testability.

The rest of the paper is structured as follows: Section 2 provides an overview of major class cohesion metrics. Section 3 presents the concept of coupling between objects and some well known coupling metrics. Section 4 presents some related work addressing the relationships which may exist between object-oriented metrics and some quality characteristics. Section 5 presents the new definition of class cohesion that we propose based on the new criterion that we introduce in this paper. Section 6 presents the first step of the experiment that we conducted (statistic test). Section 7 presents the empirical investigation that we conducted to explore the relationship between cohesion and coupling. Finally, conclusions and future work directions are given in section 8.

2 CLASS COHESION METRICS

Classes are considered as the basic units of object-oriented software. Classes should then be designed to have a good quality. However, improper modeling in the design phase can produce classes with low cohesion. In order to assess class cohesion in object-oriented systems, several metrics have been proposed in literature. Many authors have defined class cohesion by proposing their cohesion metrics. Most of the proposed cohesion metrics are inspired from the LCOM (Lack of COhesion in Methods) metric defined by Chidamber and Kemerer [Chi 91, Chi 94, Chi 98]. Many authors have redefined the LCOM metric. Many cohesion metrics have been presented in detail and are categorized in [Bri 98]. A class is more cohesive, as stated in [Cha 00], when a larger number of its instance variables are referenced by a method (LCOM5 [Hen 96], Coh [Bri 98]), or a larger number of methods pairs share instance variables (LCOM1 [Chi 91], LCOM2 [Chi 94], LCOM3 [Li 93], LCOM4 [Hit 95], Co [Hit 95], TCC and LCC [Bie 95], DC [Bad 95]). Some of these metrics take into account the interactions (at different levels) between methods.

3 COUPLING BETWEEN CLASSES

Coupling between classes allows evaluating in which proportion an entity uses other entities. Stevens & al. [Ste 74] define coupling as a measure of the strenght of the association established by a connection between two modules. According to [Pre 05], coupling is a measure of interconnection between modules forming the structure of the software. A module presenting a high coupling is a complex module. This complexity makes, among other things, the module difficult to understand, difficult to detect and correct errors, and to change that module. The complexity of a system can be reduced to the design of modules with low coupling. Well known practices in software engineering tend to promote low coupling between components to minimize interdependencies and facilitate evolution [Lar 03, Som 04, Pre 05]. In literature, several studies demonstrate that coupling metrics are good predictive indicators of OO systems maintainability. However, there exist empirical insufficiencies that clarify their significance for the prevision in maintainability [Dag 03]. Among coupling metrics, we cite CBO (Coupling Between Objects) of Chidamber and Kemerer [Chi 94], MPC (Message-Passing Coupling) and DAC (Data Abstraction Coupling) of Li and Henry [Li 93, Li 95] or OLC (Object Level Coupling) and CLC (Class Level Coupling) of Hitz and Montazeri [Hit 95]. Brian & al. counted 23 coupling metrics [Bri 97]. In the context that we are interesting in, we used the CBO metric proposed by [Chi 94], largely known as a good coupling metric between classes. In our future work, we plan on extending our study to integrate other coupling metrics.

4 RELATED WORK

During the three last decades, a large number of software metrics were proposed, among others, coupling, cohesion, and complexity metrics. But there is a little understanding of the empirical hypotheses of many of these measures [Agg 06]. It is often hard to determine which metric is the most useful and the one that predicts the most efficiently certain aspects relative to quality such as maintainability, testability, changeability, or other characteristics. Dag & al. mention in [Dag 03] the fact that there exists empirical insufficiencies clarifying their siginificance for the prediction of certain quality factors, in particular maintainability. Among papers addressing this question, the paper of Dagpinar & al. [Dag 03] is particularly interesting. The results obtained in their case demonstrate that metrics of size and direct coupling importation are significant indicators of class maintainability, opposed to inheritance, cohesion and indirect exportation coupling.

In [Agg 06], an empirical study regarding the majority of OO coupling metrics was realised. The objective of this study was to identify the most significant metrics in terms of provided information. The theoretical analysis of these metrics suggests that only 6 of them (NOA, NOM, MPC, DAC, LCOM and LCC) give enough information to be used, and the other metrics correspond to subsets of the retained metrics or give the same information but in another format. Aggarwal & al. [Agg 06] addressed the correlation between existing coupling metrics and their relationship to fault proneness. The prediction model proposed in [Agg 06b] shows that coupling metrics are highly correlated to fault proneness. Zhou & al. [Zho 06] focused on the relationship between design metrics (CBO, WMC, RFC, LCOM, etc.) and fault-proneness when taking fault severity into account. In this context, the fact of determining if there exists a relationship between cohesion and coupling could lead us to believe that there also exist a relationship between cohesion and, for example, maintainability, testability, as well as fault proneness. This needs, however, more investigations on the direct relationship that may exist between cohesion and those characteristics to draw strong conclusions. This last aspect will be the subject of our future work, and is out of the scope of this paper.

5 CLASS COHESION ASSESSMENT: A NEW MEASURE

Class cohesion in our approach, as stated initially in [Bad 95], is defined in terms of the relative number of related methods in a class. It is comparable to the approach adopted by Bieman and Kang in [Bie 95]. We have revised our initial definition of class cohesion [Bad 95] by extending the methods invocation criterion in the one hand and introducing the concept of indirect usage of attributes defined by Bieman & al. in [Bie 95] in the other hand. We have also extended this concept to the methods invocation criterion. The new definition of class cohesion and corresponding metrics have been experimented [Bad 03, Bad 04]. The obtained results have shown that the introduced criterion and the extension of the original criteria allow capturing more pairs of related methods than the others class cohesion metrics proposed in literature. The difference between the values obtained using the original and the extended metrics was statistically significant [Bad 04]. As stated previously, we conducted after that several experiments on various systems. The obtained results and particularly the analysis of the code of some programs allowed us to observe, in several situations, that methods of a class may be functionally connected in other ways. Our experiments also allowed us to observe the following. For several of the studied systems, a significant part of class attributes were, in fact, reference attributes. Those attributes were shared by methods and were the basic criterion (connection between methods) that was used by all existing class cohesion metrics. Among several analyzed systems, we observed that in some of them more than 20% of the attributes were, in fact, reference attributes. This is a natural thing in OO systems knowing that classes collaborate, according to their respective responsibilities, to implement a given task. Reference attributes are used to insure the necessary visibility between objects [Lar 03]. Then, a question arises: why use all the attributes of a class (common usage of attributes criterion) knowing that part of them can be reference attributes and not use the objects passed as parameter (non primitive) also as cohesion criteria ? It is in this context that we explored the introduction of a new criterion : Common Objects Parameters. We give, in what follows, the cohesion criteria used in our approach and the resulting cohesion metrics. The two first criteria have been used and extended in [Bad 03, Bad 04]. The third creterion is the new criterion that we introduce and experiment in this paper.

Two methods can be connected, in fact, in many ways. The adopted approach for the estimation of class cohesion is based on different relationships that may exist between its methods. It takes into account different ways of capturing the functional cohesion in a class, by focusing on the proposed cohesion criteria: Attributes Usage Criterion, Methods Invocation Criterion, and Common Objects Parameters. Class cohesion refers essentially the relatedness of public methods of a class, which represent the functionalities used by its clients. The others methods of the class are included indirectly through the public methods.

Attributes Usage Criterion (UA)

Let us consider a class C. Let A = {A1, A2,…, Aa} be the set of its attributes and PUM = {M1, M2, …, Mn} be the set of its public methods. Let UA_{Mi} be the set of all the attributes used directly or indirectly by the public method Mi. An attribute is used directly by a method Mi, if the attribute appears in the body of the method Mi. The attribute is indirectly used by the method Mi, if it is used directly by another method of the class that is invoked directly or indirectly by Mi. There are n sets UA_{M1}, UA_{M2}, …, UA_{Mn}. Two public methods Mi and Mj are directly related by the UA relation if $UA_{Mi} \cap UA_{Mj} \# \Phi$. It means that there is at least one attribute shared (directly or indirectly) by the two methods.

Methods Invocation Criterion (IM)

Let us consider a class C. Let PUM = {M1, M2, …, Mn} be the set of its public methods and PRM = {I1, I2, …, Ik} be the set of its other (private and protected) methods. Let PUM_{Mi} be the set of all the public methods of the class C, which are invoked directly or indirectly by the public method Mi. A public method Mj is called directly by a public method Mi, if Mj appears in the body of Mi. A public method Mj is indirectly called by a public method Mi, if it is called directly by another method of the class C that is invoked directly or indirectly by Mi. There are n sets PUM_{M1}, PUM_{M2}, …, PUM_{Mn}. Let PRM_{Mi} be the set of all the other methods (private and protected) of the class C, which are invoked directly or indirectly by the public method Mi. There are n sets PRM_{M1}, PRM_{M2}, …, PRM_{Mn}. Let $IM_{Mi} = PRM_{Mi} \cup PUM_{Mi}$ be the set of all the methods of the class C, which are invoked by the public method Mi. There are n sets IM_{M1}, IM_{M2}, …, IM_{Mn}. Two public methods Mi and Mj are directly related by the IM relation if $IM_{Mi} \cap IM_{Mj} \# \Phi$. We also consider that Mi and Mj are directly related if $Mj \in IM_{Mi}$ or $Mi \in IM_{Mj}$.

Common Objects Parameters (CO)

Ler us consider a class C. Let PUM = {M1, M2, …, Mn} be the set of its public methods. Let UCO_{Mi} be the set of all the parameters (of object type) of the method Mi. There are n sets UCO_{M1}, UCO_{M2}, …, UCO_{Mn}. Two public methods Mi and Mj are directly related by the UCO relation if $UCO_{Mi} \cap UCO_{Mj} \# \Phi$. It means that there is at least one parameter of object type used by the two methods.

Cohesion based on the direct relation

Two public methods Mi and Mj may be directly connected in many ways: they share at least one instance variable in common (UA relation), or interact at least with another method of the same class (IM relation), or share at least one object passed as argument (CO relation). In this context, the two methods may be directly connected by one or more creteria. It means that the two methods are directly connected if: $UA_{Mi} \cap UA_{Mj} \# \Phi$ or $IM_{Mi} \cap IM_{Mj} \# \Phi$ or $UCO_{Mi} \cap UCO_{Mj} \# \Phi$.

Let us consider a class C with PUM = {M1, M2, …, Mn} the set of its public methods. The maximum number of public methods pairs, is n * (n – 1) / 2. Consider an undirected graph G_D, where the vertices are the public methods of the class C, and there is an edge between two vertices if the corresponding methods are directly related. Let E_D be the number of edges in the graph G_D. The degree of cohesion in the class C based on the direct relation between its public methods is defined as: $DC_{DE} = |E_D| / [n * (n-1) / 2] \in [0,1]$. DC_{DE} (as an extention of DC_D [Bad 03, Bad 04]) gives the percentage of public methods pairs, which are directly (as defined below) related. The Lack of Cohesion in the Class (LCC_{DE}) is than given by : $LCC_{DE} = 1- DC_{DE} \in [0,1]$.

Cohesion based on the indirect relation

Two public methods Mi and Mj can be indirectly related if they are directly or indirectly related to a method Mk. The indirect relation, introduced by Bieman and Kang in [Bie 95], is the transitive closure of the direct relation. We use this concept in our approach for identifying the indirect related methods. Consider now an undirected graph G_I, where the vertices are the public methods of the class C, and there is an edge between two vertices if the corresponding methods are directly or indirectly related (transitive closure of the graph G_D). Let E_I be the number of edges in the graph G_I. The degree of cohesion in the class C in this case (direct and indirect relations) is defined as: $DC_{IE} = |E_I| / [n * (n-1) / 2] \in [0,1]$. DC_{IE} (as an extention of DC_I [Bad 03, Bad 04]) gives the percentage of public methods pairs, which are directly or indirectly related. The Lack of Cohesion in the Class (LCC_{IE}) is than given by: $LCC_{IE} = 1- DC_{IE} \in [0,1]$.

6 EXPERIMENTAL STUDY

As a first experimentation of the new criterion and to achieve significant and general results, we have chosen several systems, which can be freely downloaded from the web. Our goal was to analyze a maximum number of Java classes from different systems in order to collect significant data for the experiment. The considered systems vary in size and domain. The following section gives some of their characteristics. The goal, at this step, was essentially to explore if the new criterion is statistically significant before more investigations. We extended the cohesion measurement tool (in Java) for Java programs, that we developed for [Bad 04], to automate the computation of our metrics (DC_D, DC_{DE}, DC_I and DC_{IE}).

Several classes in the considered systems have only one method or do not have any methods. These classes were considered as special classes and have been excluded from our measurements. We also excluded all abstract classes. Overloaded methods within the same class were treated as one method. Moreover, all special methods (constructors, destructors) were removed. We collected the values for all the selected metrics from the test systems. For each metric, we calculated some descriptive statistics (minimum, maximum, mean, median, and standard deviation).

Selected systems

The experiment concerned more than 800 classes. The followed methodology and the obtained results are presented in the following sections. The selected systems are :

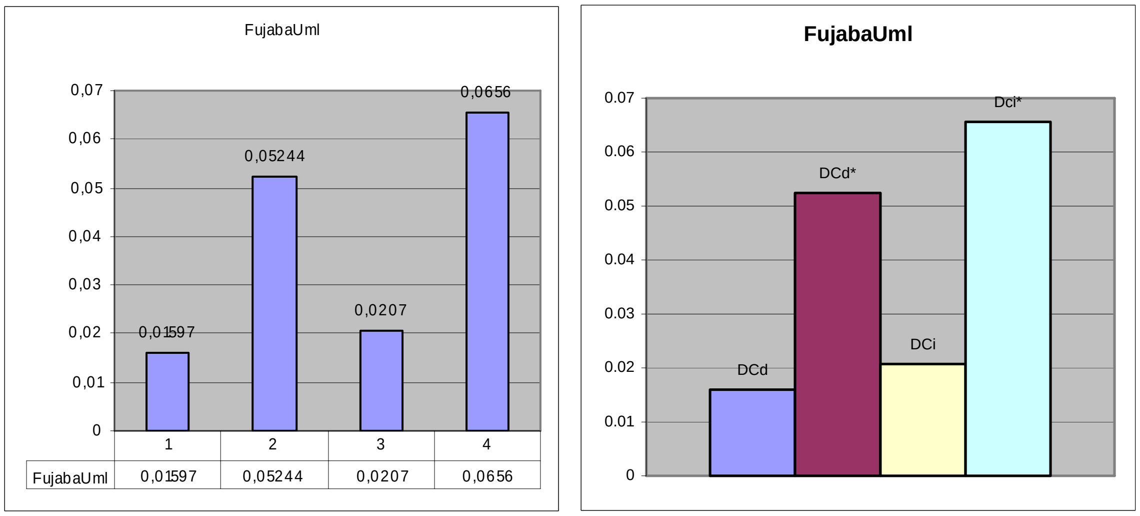
- System1: JIU0.10 (Java Imaging Utilities) is a library in Java for the change, the edition, the analysis and the backup of pixels of image files (http://sourceforge.net/projects/jiu). This system contains 180 classes.
- System2: JIU0.11 (Java Imaging Utilities) is an evolution of the first system (http://sourceforge.net/projects/jiu) and contains 191 classes.
- System3: FujabaUML is a software development tool allowing the easy extention of UML and the development with Java with the addition of plug-ins (http://www.fujaba.de). This system contains 186 classes.
- System4: Wbemservices is a Java open source implementation of Web Based Enterprise Management (WBEM) for commercial and non commercial applications. It is composed of API, of servers, client applications and tools (http://wbemservices.sourceforge.net/). It contains 463 classes.

Systems	Des. Stat	DC_D	DC_{DE}	DC_I	DCI_E
Jiu1	Moyenne	0,16027	0,17384	0,1922	0,2178
	Sdt.dev	0,13686	0,1378	0,1638	0,2178
Jiu2	Moyenne	0,2497	0,2635	0,3102	0,3350
	Sdt.dev	0,16466	0,1714	0,2292	0,2246
Fujaba	Moyenne	0,01597	0,05244	0,0207	0,0656
	Sdt.dev	0,01479	0,05861	0,0201	0,0739
Wbemservices	Moyenne	0,08138	0,2286	0,1013	0,2747
	Sdt.dev	0,14164	0,2051	0,1678	0,2332

Table 1: Average values of cohesion.

Results

We measured class cohesion values for the 4 selected systems. Table 1 shows the mean values of the metrics for the selected systems. The obtained results for DC_{DE} et DC_{IE} show clairly that they capture more pairs of connected methods than DC_D et DC_I. Figures 1 and 2, for example, give the mean values of the metrics for systems 3 and 4. The two metrics DC_{DE} and DC_{IE} seem capturing an additional aspect of characteristics of classes that the other do not. The main goal of this work is to demonstrate the relevance of the new criterion. For this raison, we will not discuss the cohesion values of the selected systems. The results given in table 1 show however that these systems are not cohesive.

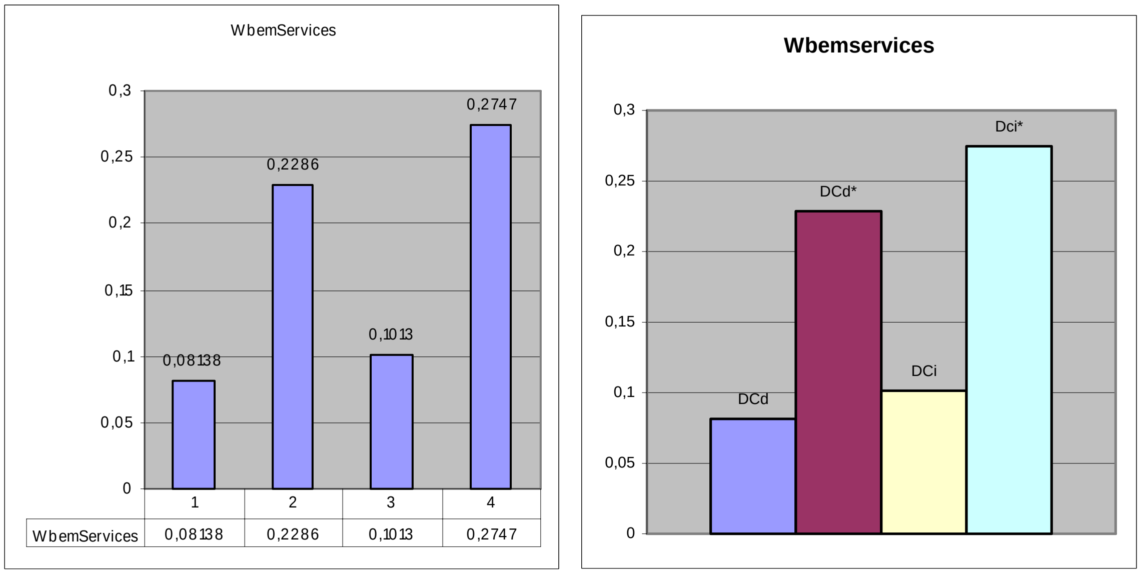


	1	2	3	4
FujabaUml	0,01597	0,05244	0,0207	0,0656

Figure 1: Representation and comparison of the average values for FujabaUml.

Validation of the new criterion

The objective of this section is to compare the results of DC_D and DC_{DE} on one side and the results of DC_I and DC_{IE} on the other. The objective is to see if the difference brought by the introduced criterion is statistically significant. Our goal is then to demonstrate that DC_{DE} and DC_{IE} are more significant than DC_D and DC_I and that they allow capturing more pairs of connected methods. To validate our hypotheses, we use an appropriate statistical test: the PAIRED t-TEST [Hin 03]:



	1	2	3	4
WbemServices	0,08138	0,2286	0,1013	0,2747

Figure 2 : Representation and comparison of the average values for Wbemservices.

Let μ1 be the mean value of DC_{DE} (or DC_{IE}) and μ2 be the mean value of DC_D (or DC_I). We present the following two statistical hypotheses :

- H0 : μ1= μ2 The metrics are equivalent.
- H1 : μ1> μ2 DC_{DE} (or DC_{IE}) is more significant than DC_D (or DC_I).

Let Diff be the value of (μ1- μ2). The above test is equivalent to:

- H0 : Diff = 0.
- H1 : Diff >0.

The test statistic is: *Z =d/ [Sd / sqrt(N)]*
With *d* : the mean value of sample Diff
Sd : the standard deviation of sample Diff and
N : the number of classes in sample Diff.

Tables 2 and 3 present respectively the comparison between DC_D and DC_{DE} on one side and DC_I and DC_{IE} on the other.

Systèmes	Des. Stat	DC_D	DC_{DE}	Diff	Z	$Z\alpha$
Jiu1	Moyenne	0,16027	0,17384	0,01356	1,799	1,645
	Sdt .dev	0,13686	0,1378	0,01685		
Jiu2	Moyenne	0,2497	0,2635	0,0228	2,4635	1,645
	Sdt.dev	0,16466	0,1714	0,0207		
Fujaba	Moyenne	0,01597	0,05244	0,03646	2,6547	1,645
	Sdt.dev	0,01479	0,05861	0,05663		
Wbemservices	Moyenne	0,08138	0,2286	0,1472	4,7917	1,645
	Sdt.dev	0,14164	0,2051	0,1869		

Table 2 : Comparison between DC_D and DC_{DE}

Systèmes	Des. Stat	DC_I	DC_{IE}	Diff	Z	$Z\alpha$
Jiu1	Moyenne	0,1922	0,2178	0,0255	1,620	1,645
	Sdt.dev	0,1638	0,2178	0,0352		
Jiu2	Moyenne	0,3102	0,3350	0,02485	1,5498	1,645
	Sdt.dev	0,2292	0,2246	0,0358		
Fujaba	Moyenne	0,0207	0,0656	0,0448	2,6494	1,645
	Sdt.dev	0,0201	0,0739	0,0697		
Wbemservices	Moyenne	0,1013	0,2747	0,1734	5,7969	1,645
	Sdt.dev	0,1678	0,2332	0,1819		

Table 3 : Comparison between DC_I and DC_{IE}

The procedure consists on comparing Z, for each system, to a value Zα (the value of α is 0.05). If the value of Z is higher than Zα, we refuse hypothesis H0 : Diff = 0 and accept H1 : Diff > 0. In this case, the statistical test is significant and we can conclude that metric DC_{DE} (or DC_{IE}) is more significant than metric DC_D (or DC_I). This means that the added criterion is significant and allows capturing an additional aspect of classes' properties. We collected data on the metrics from the selected systems and calculated Diff and Z for these systems. These results are presented in tables 2 and 3. They clearly show that, for the majority of the tested systems, Z is higher thant Zα. The systems for which Z is lower thant Zα are the systems for which N is low. Globally, the results show that DC_{DE} (or DC_{IE}) is more significant than DC_D (or DC_I). This statistical validation demonstrate the relevance of the new cohesion criterion for capturing new pairs of connected methods. The obtained results show that the extended cohesion metrics, based on the introduction of the last proposed criteria, capture more pairs of connected methods than metrics DC_D and DC_I.

7 EXPLORING EMPIRICALLY THE RELATIONSHIP BETWEEN COHESION AND COUPLING

Introduction

A widely known belief in the software engineering community states that, intuitively, a high cohesion is related to low coupling, and vice-versa [Lar 03, Som 04, Pre 05]. However, to the best of our knowledge, no validation of this principle was proposed to this day. It is in this context that we explore in this section the relationship that may exist between our cohesion metrics and coupling. This appears to us as a first lead for the validation of our metrics if the relationship is really confirmed. Of course, to draw a final conclusion on this relationship, complementary and deeper investigations should be performed. These studies could eventually consider the exploration of the relationship that our metrics could have directly with high level quality characteristics such as testability, changeability and maintainability.

Selected Systems

The experiment we performed considered six systems that vary in size (number of classes) and domain. The selected systems are (more than 500 classes):

- System 1 : Gnujsp 1.0.1, GNUJSP is a free implementation of Java Server Pages of Sun (http://klomp.org/gnujsp). This system contains 56 classes.
- System 2 : JIU 0.12, JIU (Java Imaging Utilities) is a library in Java for loading, editing, analyzing and saving pixels in image files (http://sourceforge.net/projects/jiu). This system has 77 classes.
- Systeme 3 : fujabaUml.4, FujabaUML is a software development tool allowing the easy extension of UML and Java development with the use of plug-ins (http://www.fujaba.de). This system contains 60 classes.

- System 4 : jexcelapi 2.6, JExcelApi is a Java library that grants the possibility of reading, writing and modifying Microsoft Excel Worksheets (http://sourceforge.net/projects/jexcelapi). It contains 110 classes.
- System 5 : moneyjar 0.8, Moneyjar is a Java library for financial applications. It simplifies treasury management, currency exchange, tax calculations and invoice management (http://sourceforge.net/projects/moneyjar). It contains 20 classes.
- System 6 : wbemservices 1.0.0, Wbemservices is an open source Java implementation of Web Based Enterprise Management (WBEM) for commercial and non commercial applications. It is a project composed of APIs, of servers, of client applications and of tools (http://wbemservices.sourceforge.net/). This system contains 180 classes.

Experimental Process: First phase

We collected, from the set of considered systems, 6 in all, the data corresponding to our four cohesion metrics, as well as data corresponding to CBO metric. We used the Together tool to calculate CBO. The study of the obtained results, in a visual form first, lead us to believe that there could be a link between cohesion and coupling according to the considered metrics. The graphs of figures 6, 7 and 8 show the distribution of the values of cohesion and of coupling for, for example, five of the analyzed systems, allowing us to observe what seems to be a negative link between cohesion and coupling. We can clearly observe, in a global manner, that when cohesion increases, coupling decreases. The inverse is also true.

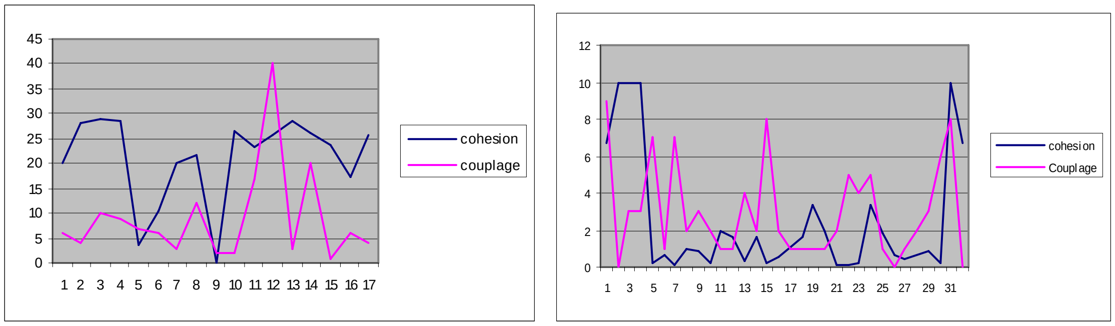


Figure 6: Distribution graphs for Coupling – Cohesion in Gnujsp and Fujaba.

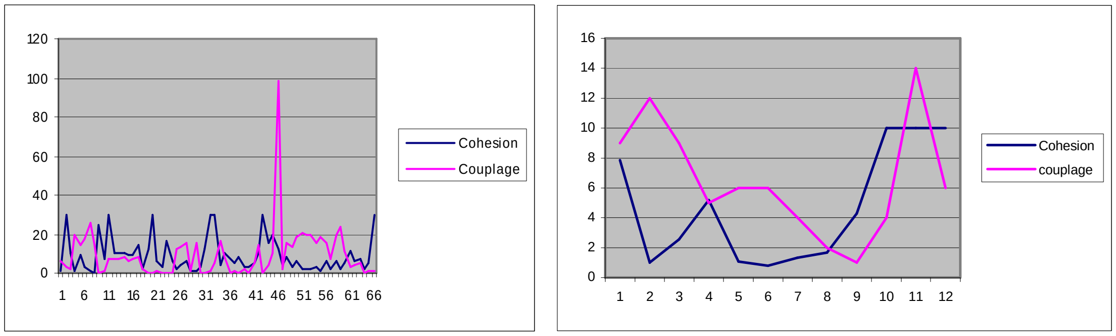


Figure 7 : Distribution graphs for Coupling – Cohesion in Jiu and Moneyjar.

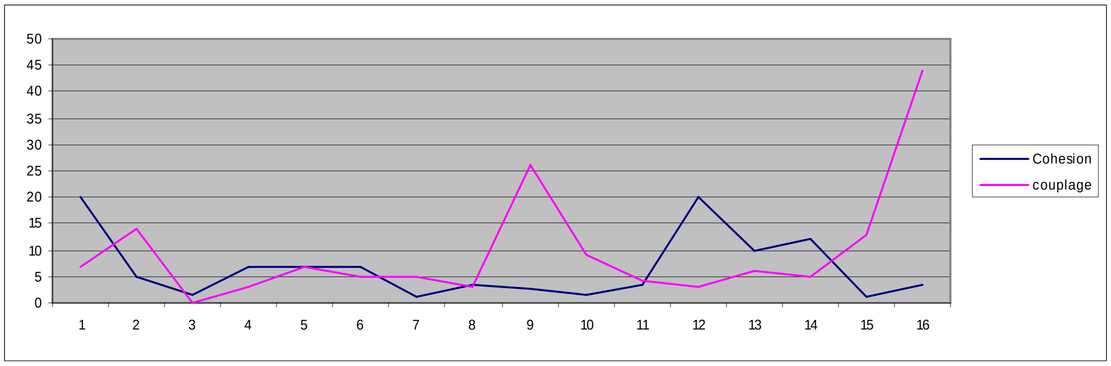


Figure 8 : Distribution graphs for Coupling – Cohesion in JexcelApi.

Experimental Process: Second phase

The objective of this second step of our experiments consists on an attempt to explain the observations given previously and, eventually, confirm the hypothesis of a relationship between coupling and cohesion such as it was introduced in this section. To test the hypothesis, if cohesion is correlated with coupling, we consider the four cohesion metrics: DC_I, DC_D, DC_{DE}, and DC_{IE}, as well as CBO coupling metric.

In our experiments, we collected data on the selected metrics from each of the considered systems, and then we used the Spearman coefficient (rank statistics) to test the correlation. This test is appropriate since the dependence seems to be non linear according to the previous graphs. Analysis of the data sets are done by calculating the Spearman dependence coefficients for each pair of metrics (a metric of cohesion, CBO). The Spearman statistic is based on ranks of the observations. The value of the Spearman statistic is a number between -1 and 1, -1 being a perfect negative dependence and +1 a perfect positive dependence.

Results

Regression Study

First comes a regression study between coupling and the different cohesion metrics. Each cohesion metric was associated to the retained coupling metric to do a regression analysis between the two variables. The goal of this first statistical analysis is to verify if there exists a linear relationship between the cohesion metrics and coupling. Here are a few terms used in this part of paper:

- Regression model: It is the regression model used. The independent variables are the cohesion metrics DC_{DE}, DC_{IE}, DC_D, DC_I and the dependant variable is the coupling metric CBO;
- Dependant variable: A random variable to predict;
- Independant variable: A predictive variable;
- R2 (r-square): The percentage of variance of the dependent variable explained by the independent variables in the regression model for the given sample of the population.

- Population : The set of classes taken into consideration at the test level;
- Adjusted R-square: The percentage of variance of the dependent variable explained by the independent variables in a regression model of the population;
- Sum of squares of regression: The variance of the dependent variable explained by the regression model;
- Sum of squares of residual: The variance not explained by the dependent variable;
- Mean squares of residual: The sum of squared residues divised by the number of freedom degrees of the residues;

Table 4 shows the values of R2 obtained.

System	Cohesion Metric	R2 vs Coupling
FujabaUml	DC_{DE}	0.0118
	DC_{IE}	0.0081
	DC_{D}	0.0081
	DC_{I}	0.0054
System	Cohesion Metric	R2 vs Coupling
Gnujsp	DC_{DE}	0.2835
	DC_{IE}	0.2676
	DC_{D}	0.4657
	DC_{I}	0.4506
System	Cohesion Metric	R2 vs Coupling
JIU	DC_{DE}	0.0228
	DC_{IE}	0.0267
	DC_{D}	0.0186
	DC_{I}	0.0221
System	Cohesion Metric	R2 vs Coupling
Moneyjar	DC_{DE}	0.0226
	DC_{IE}	0.0237
	DC_{D}	0.032
	DC_{I}	0.0331

Table 4 : Values of R2 in the different systems.

To study another variant of this relation between the metrics of cohesion and the coupling metric, the logarithm of the coupling value was defined. A regression between this logarithm and the cohesion value is done. The results obtained are shown in table 5.

System	Cohesion Metric	R2 vs logCouplage
FujabaUml	DC_{DE}	0.0027
	DC_{IE}	0.0012
	DC_{D}	0.0019
	DC_{I}	0.0008
System	Cohesion Metric	R2 vs logCouplage
Gnujsp	DC_{DE}	0.0032
	DC_{IE}	0.0066
	DC_{D}	0.0628
	DC_{I}	0.0504
System	Cohesion Metric	R2 vs logCouplage
JIU	DC_{DE}	0.0341
	DC_{IE}	0.043
	DC_{D}	0.0459
	DC_{I}	0.0545
System	Cohesion Metric	R2 vs logCouplage
Moneyjar	DC_{DE}	0.015
	DC_{IE}	0.0148
	DC_{D}	0.0168
	DC_{I}	0.0162

Table 5 : R2 obtained with the log of a coupling value.

Concerning this first experiment, we based our study on the values of the R2 statistic to interpret the relation that eventually links coupling and cohesion. For example, for system JIO in table 4, values 0.0228 and 0.0267 of metrics DC_{DE} and DC_{IE} respectively actually represent percentages of the variance of coupling explained by the cohesion metrics. Therefore, 2.28% and 2.67% of the variance of coupling, respectively, is explained by the cohesion metrics DC_{DE} and DC_{IE}. Concerning table 5, for the same JIU system, values 0.0341 and 0.0430, respectively for cohesion metrics DC_{DE} and DC_{IE}, represent the percentages of the logarithm of the variance explained by the cohesion metrics. Therefore, 3.41% and 4.3% of the logarithm of variance is explained respectively by cohesion metrics DC_{DE} and DC_{IE}. Given the obtained values in this experiment and taking into account the noted observations in previous section (the relationship seems to be non linear), we conducted a second experiment using the Spearman correlation.

Spearman Correlation study (rank statistic)

As a second step, we calculated the correlation degree (according to Spearman) between the cohesion metrics and coupling in the selected systems. Table 6 presents the obtained results.

System	Statistic	DC_{IE}-CBO	DC_{DE}-CBO	DC_{I}-CBO	DC_{D}-CBO
Gnujsp	Spearman Coeff.	-0.354545	-0.35892	-0.35455	-0.35892
	Test statistic	-2.786373	-2.8258	-2.78637	-2.8258
	P-value	0.0036697	0.003299	0.00367	0.003299
jiu	Spearman Coeff.	-0.50857	-0.47888	-0.50337	-0.47584
	Test statistic	-5.11527	-4.72409	-5.04502	-4.68533
	P-value	1.17E-06	5.27E-06	1.53E-06	6.11E-06
fujabaUml	Spearman Coeff.	-0.425590442	-0.43809	-0.29225	-0.31195
	Test statistic	-3.5817696	-3.71155	-2.3273	-2.5005
	P-value	0.000349459	0.000232	0.011731	0.007625
jexcelapi	Spearman Coeff.	-0.18723	-0.22039	-0.19318	-0.21987
	Test statistic	-1.98076	-2.34805	-2.04612	-2.3423
	P-value	0.02508	0.010346	0.021587	0.010499
moneyjar	Spearman Coeff.	-0.00602	-0.01955	-0.02105	-0.03459
	Test statistic	-0.02552	-0.08295	-0.08934	-0.14683
	P-value	0.48996	0.467402	0.4649	0.442451
wbemservices	Spearman Coeff.	-0.242732	-0.295322901	-0.26708	-0.3055
	Test statistic	-3.338282	-4.124041493	-3.69767	-4.28049
	P-value	0.0005133	2.85E-05	0.000145	1.52E-05

Table 6 : Results of the Spearman rank statistic method.

The goal of this experiment was to find a correlation (negative) between the cohesion metrics and coupling metric we selected. The experiment consisted on verifying if the correlation is significatively lower than 0 (in the statistical sense) for a negative dependence. A statistical test was executed. The statistical test must then be compared to a Student variable calculated with n-2 freedom degrees, and where n is the size of the sample.

The P-value indicates the probability of obtaining such a value under the null hypothesis of absence of dependence. In general, if P-value < 0.05 (error margin), we conclude that a negative dependence is significant. Therefore, for the set of tested systems and according to the values of table 6, only the *moneyjar* system has P-values > 0.05 for all combinations (cohesion metric – coupling metric). We observe values of 0.48996, 0.46740, 0.4649, and 0.442451 for, respectively, cohesion metrics DC_{IE}, DC_{DE}, DC_{I}, DC_{D} compared to the coupling metric CBO. For the rest of the studied systems, the P-values are all < 0.05 for the entire set of combinations (cohesion metric – coupling metric). Therefore, all systems, according to table 6, present a significative negative dependence between cohesion and coupling, with the exception of the *moneyjar* system. One possible explanation is that only that system actually has a relatively low number of classes (20) compared to the other systems. Therefore, to observe a significant negative dependence, it would be interesting to consider systems having a high number of classes.

The obtained results demonstrate that there is a relationship between the cohesion metrics and the coupling metric. It seems also possible that the more the number of classes of a system is high, the more the dependence relation between cohesion and coupling is visible (non linear dependence relations). Even if the obtained results, considering the different systems selected for our study, clearly confirm that the relation that exists between coupling and cohesion (when one increases, the other decreases), it is however necessary to continue the exploration of this relation for other systems before drawing any global conclusions.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we have revised our definition of class cohesion [Bad 03, Bad 04] and proposed a new cohesion criterion. Our main goal in this work was to validate the introduced criterion (Common Objects Parameters) and our approach for class cohesion assessment. We developed a cohesion measurement tool for Java programs to automate the computation of the class cohesion metrics that we propose. In order to demonstrate the effectiveness of the new criterion and the proposed metrics for class cohesion, we performed an empirical study on several systems. In our experiments, several hundred of classes were analysed. The selected systems vary in size and domain. The obtained results show that the extented metrics, based on the introduced criterion, capture more pairs of connected methods. Furthermore, and with the goal of validating our new metrics, we explored the eventual relationship that theses metrics could have with coupling. The experiment we conducted in this second step allowed us to analyse several hundreds of classes The obtained results demonstrated that there is in fact a significative negative correlation between cohesion and coupling for the studied systems. The results also seem to indicate that the more the number of classes in a system is high, the more the dependence relation between cohesion and coupling is confirmed.

We believe that the present work constitutes an improvement of class cohesion measurement. During our experiment, we collected several data on the analyzed classes. An important part of the collected data has been treated during this work. Actually, we are analyzing the rest of the collected data. As future work we plan to: (1) study in detail the weakly cohesive classes; (2) study the proposed metrics by including other aspects of object-oriented design such as inheritance between classes; (3) continue to explore the cohesion-coupling relation by integrating in the experiment other coupling metrics as well as other cohesion metrics to refine our study and draw more global conclusions; and finally (4) explore the relationship (without going through coupling) between our cohesion metrics and some high level quality characteristics such as testability, changeability and maintainability.

ACKNOWLEDGEMENTS

This work was supported by a NSERC (Natural Sciences and Engineering Research Council of Canada) grant.

REFERENCES

[Agg 06] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, RuchikaMalhotra, Empirical study of object-oriented metrics, In Journal of Object Technology, vol. 5. no. 8, November-December 2006, pp. 149-173.

[Agg 06] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, RuchikaMalhotra, Investigating the effect of coupling metrics on fault proneness in object-oriented systems, SQP, vol. 8 no. 4, 2006.

[Ama 02] H. Aman, K. Yamasaki, H. Yamada and MT. Noda, A proposal of class cohesion metrics using sizes of cohesive parts, *Knowledge-Based Software Engineering*, T. Welzer et al. (Eds), pp. 102-107, IOS Press, September 2002.

[Bad 95] L. Badri, M. Badri and S. Ferdenache, Towards Quality Control Metrics for Object-Oriented Systems Analysis, Proceedings of TOOLS (*Technology of Object-Oriented Languages and Systems*) *Europe'95*, Versailles, France, Prentice-Hall, March 1995.

[Bad 03] L. Badri and M. Badri, A New Class Cohesion Criterion: An empirical study on several systems, Proceedings of QAOOSE'03, 2003.

[Bad 04] L. Badri and M. Badri, A proposal of a new class cohesion criterion: An empirical Study, In Journal of Object Technology, 2004.

[Bas 96] V.R. Basili, L.C. Briand and W. Melo, A validation of object-oriented design metrics as quality indicators, *IEEE Transactions on Software Engineering*, 22 (10), pp. 751-761, October 1996.

[Bie 95] J.M. Bieman and B.K. Kang, Cohesion and reuse in an object-oriented system, *Proceedings of the Symposium on Software Reusability (SSR'95)*, Seattle, WA, pp. 259-262, April 1995.

[Boo 94] G. Booch, Object-Oriented Analysis and Design With Applications, Second edition, Benjamin/Cummings, 1994.

[Bri 97] L. C. Briand, J. Daly, V. Porter, and J. Wuest, The Dimensions of Coupling in Object-Oriented Design, OOPSLA'97,1997.

[Bri 98] L.C. Briand, J. Daly and J. Wusr, A unified framework for cohesion measurement in object-oriented systems, *Empirical Software Engineering, 3 (1)*, pp. 67-117, 1998.

[Bri 00] L. Briand, J. Wuest, J. Daly and V. Porter, Exploring the relationships between Design Measures and software quality in object-oriented Systems, *Journal of Systems and Software*, No. 51, pp. 245-273, 2000.

[Cha 98] H. S. Chae and Y.R. Kwon, A cohesion measure for classes in object-oriented systems, *Proceedings of the fifth International Software Metrics Symposium*, Bethesda, MD, pp. 158-166, November 1998.

[Cha 00] H. S. Chae, Y. R. Kwon and D H. Bae, A cohesion measure for object-oriented classes, *Software Practice and Experience*, No. 30, pp. 1405-1431, 2000.

[Chi 91] S.R. Chidamber and C.F. Kemerer, Towards a Metrics Suite for Object-Oriented Design, *Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*, Special Issue of SIGPLAN Notices, Vol. 26, No. 10, pp. 197-211, October 1991.

[Chi 94] S.R. Chidamber and C.F. Kemerer, A Metrics suite for object Oriented Design, *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, pp. 476-493, June 1994.

[Chi 98] S.R. Chidamber, David P. Darcy, and C.F. Kemerer, Mangerial use of metrics for object-oriented sofytware : An exploratory analysis, *IEEE Transactions on Software Engineering*, Vol. 24, No. 8, pp. 629-639, August 1998.

[Dag 03] Melis Dagpinar and Jens H. Jahnke, Predicting maintenability with object-oriented metrics – An empirical comparaison, Proceedings of the 10th working conference on reverse engineering (WCRE'03), IEEE computer society, 2003.

[Ema 99] K. El Emam and W. Melo, The prediction of faulty class using object-oriented design metrics, *National Research Council of Canada NRC/ERB 1064*, 1999.

[Hen 96] B. Henderson-sellers, *Object-Oriented Metrics Measures of Complexity*, Prentice-Hall, 1996.

[Hin 03] W. W. Hines, D. C. Montgomery, D. M. Goldsman and C. M. Borror, Probability and statistics in engineering, Fourth edition, John Wiley & Sons, Inc., 2003.

[Hit 95] M. Hitz and B. Montazeri, Measuring coupling and cohesion in object oriented systems, *Proceedings of the Int. Symposium on Applied Corporate Computing*, pp. 25-27, October 1995.

[Kab 00] H. Kabaili, R.K. Keller, F. Lustman and G. Saint-Denis, Class Cohesion Revisited: An Empirical Study on Industrial Systems, *Proceeding of the Workshop on Quantitative Approaches Object-Oriented Software Engineering*, QAOOSE'2000, France, June 2000.

[Kab 01] H. Kabaili, R.K. Keller and F. Lustman, Cohesion as Changeability Indicator in Object-Oriented Systems, *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering (CSMR 2001)*, Estoril Coast (Lisbon), Portugal, March 2001.

[Lar 03] G. Larman, *Applying UML and Design Patterns, An introduction to object-oriented analysis and design and the unified process*, Second edition, Prentice Hall, 2003.

[Li 93] W. Li and S. Henry, Object oriented metrics that predict maintainability, *Journal of Systems and Software*, Vol. 23, pp. 111-122, February 1993.

[Li 95] W. Li, S. Henry, D. Kafura and R. Schulman. Measuring Object-Oriented Design. In *Journal of Object-Oriented Programming*, Vol. 8, No. 4, pages 48-55, July/August 1995.

[Pre 05] R. S. Pressman, *Software Engineering, A practitioner's approach*, Fifth edition, Mc Graw Hill, 2005.

[Som 04] I. Sommervile, Software Engineering, 2004.

[Ste 74] W.P. Stevens, G.J. Myers and L.L. Constantine. Structured Design. In *IBM Systems Journal*, Vol. 13, No. 2, pages 115-139, May 1974.

[You 79] E. Yourdon and L. Constantine, *Structured Design*, Prentice Hall, Englewood Cliffs, N.J., 1979.

[Zho 06] Yuming Zhou, Hareton Leung, Empirical analysis of object-oriented design metrics for predicting high and low severity faults, IEEE Transactions on software engineering, vol. 32, no. 10, October 2006.

About the authors

Linda Badri (Linda.Badri@uqtr.ca) is professor of computer science at the Department of Mathematics and Computer Science of the University of Quebec at Trois-Rivières. She holds a PhD in computer science (software engineering) from the National Institute of Applied Sciences in Lyon, France. Her main areas of interest include object and aspect-oriented software engineering, software quality attributes, maintenance, and reverse engineering.

Mourad Badri (Mourad.Badri@uqtr.ca) is professor of computer science at the Department of Mathematics and Computer Science of the University of Quebec at Trois-Rivières. He holds a PhD in computer science (software engineering) from the National Institute of Applied Sciences in Lyon, France. His main areas of interest include object and aspect-oriented software engineering, software quality attributes, and formal methods.

Alioune Gueye (Alioune.Gueye@uqtr.ca) is a student of computer science at the Department of Mathematics and Computer Science of the University of Quebec at Trois-Rivières. He is currently finishing his master in computer science from the University of Quebec at Trois-Rivières. His main areas of interest include object-oriented programming and metrics as well as various topic of software engineering.