Babeş–Bolyai University
Faculty of Mathematics and Computer Science
1, M. Kogălniceanu Street
400084, Cluj–Napoca, România
http://www.ubbcluj.ro

Metrics in Software Assessment

PhD Thesis Abstract

PhD Student: Camelia Şerban
Advisor: Prof. Dr. Militon Frenţiu

2012

ACKNOWLEDGMENTS

I would like to thank all people which have made this work possible.

I express my sincere gratitude to my supervisor, Professor Militon Frenţiu, for his help during these years.

I also thank Professor Horia F. Pop for research opportunities offered and valuable ideas.

The passion for research, encouragements and many ideas came from my coleague Andreea Vescan to whom I am deeply grateful.

I thank from all my heart my parents and my sister for the enormous help that they offered. They helped me move forward when I have lost all hope. Throughout these years, my husband was beside me, and I thank him for his help and encouragements he offered. I dedicate this thesis to my daughter, Emilia, with all my love.

I thank all my colleagues from the Computer Science Department, which helped me and with whom I collaborated.

Contents

Contents of the Thesis

List of author's publications

- [Ser11] **C. Serban**. God Class Design Flaw Detection In Object Oriented Design. A Case–Study. *Studia Universitas Babes-Bolyai, Seria Informatica*, LVI(4):33-38, 2011. *(MathSciNet)*

- [Ser10] **C. Serban**. A Conceptual Framework for Object-Oriented Design Assessment. *In UKSim 4th European Modelling Symposium on Mathematical Modelling and Computer Simulation*, pages 90-95, 2010. *(IEEE paper)*

- [SVP10b] **C. Serban**, A. Vescan, and H. F. Pop. A Formal Model for Component-Based System Assessment. *In Second international conference on Computational Intelligence, Modelling and Simulation*, pages 261-266, 2010. *(IEEE paper)*

- [SVP10a] **C. Serban**, A. Vescan, and H. F. Pop. A conceptual framework for component-based system metrics definition. *In 9th RoEduNet International Conference*, Sibiu, Romania, pages. 73–78, 2009. *(ISI Proceeding)*

- [Ser09b] **C. Serban**. High coupling detection using fuzzy clustering analysis. *Knowledge Engineering: Principles and Techniques (Post-proceedings of KEPT 2009), International Conference*, Babes-Bolyai University, Presa Universitara Clujeana, pages 258-265, 2009. *(ISI Proceeding)*

- [SVP09] **C. Serban**, A. Vescan, and H.F. Pop. A new component selection algorithm based on metrics and fuzzy clustering analysis. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems*, LNCS Vol. 5572, pages 621-628, 2009. *(ISI Proceeding)*

- [Ser09a] **C. Serban**, A formal approach for OOD metrics definition. *Proceedings of the First International Conference on Modelling and Development of Intelligent Systems*, Lucian Blaga University Press, pages 262-269, 2009. *(MathSciNet)*

- [Ser09c] **C. Serban**. High coupling detection using fuzzy clustering analysis. *Special Issue of Studia Universitatis Babes-Bolyai Informatica: Proceedings of The International Conference on Knowledge Engineering: Principles and Techniques*, pages 223-226, 2009. *(MathSciNet)*

- [SP08] **C. Serban** and H.F. Pop. Software quality assessment using a fuzzy clustering approach. *Studia Universitas Babes-Bolyai, Seria Informatica,* LIII(2):27-38, 2008. *(MathSciNet)*

- [SVP08] **C. Serban**, A. Vescan, and H.F. Pop. Component selection based on fuzzy clustering analysis. *Creative Mathematics and Informatics*, 17(3):505-510, 2008.

- [SV07b] **C. Serban** and A. Vescan. Metrics for component-based system development. *Creative Mathematics and Informatics*, pages 143-150, 2007. *(MathSciNet)*

- [SV07a] **C. Serban** and A. Vescan. Metrics-based selection of a component assembly. *Special Issue of Studia Universitatis Babes-Bolyai Informatica: Proceedings of The International Conference on Knowledge Engineering: Principles and Techniques*, pages 324-331, 2007. *(MathSciNet)*

- [SC06] **C. Serban** and C. Cretu. Impact on design quality of refactorings on code via metrics. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 39-44, 2006.

- [Ser06] **C. Serban**. Coupling measurement for compiled .Net code. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 21-26, 2006.

- [SM05] **C. Serban** and A. Mihis. Software quality assurance. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 207-212, 2005.

List of keywords

Software metric, object oriented design, component based development, object oriented design assessment, component based systems assessment, fuzzy clustering analysis.

1 Introduction

This PhD thesis is the outcome of the research which I have conducted in Software Engineering, particularly in the field of Software Metrics, since 2004.

Programs have continuously increased in size and complexity leading to higher development costs and lower productivity. Software systems have become inflexible (difficulty in adding new functionality), monolythical (provided functionality not based on components), and difficult to maintain (any change comes with an unending chain of adjustments in multiple places) [Mih03].

The need for quality in the software system has become more and more evident on the market. Quality improvement is only possible through quality control. As De Marco underlines: "you cannot control what you cannot measure" [DeM82]. Therefore, the interest for software measurement has vitally increased.

Measurement is not only useful but also necessary. It is essential to check up the status of the projects being developed and to find out the resources and processes involved. It is also difficult to state whether a project is good or bad, if measurements of its features (goodness, health, quality, etc.) are not performed [Bar02]. In other words, projects need to be controlled rather than just developed.

So it is obvious that software measurement has become essential in the field of Software Engineering. Software developers measure software characteristics to make sure the requirements are consistent and complete, the design is of high quality, and the code is ready to be tested. Project managers are keen on constantly measuring the process and the product to be able to tell when the software is ready for delivery making also efforts not to exceed the budget. Informed customers measure different aspects of the final product to determine whether it meets the requirements and the quality standards. Maintainers should have the background required for the product assessment to decide what should be upgraded and improved [FP97].

As a consequence to the development above, software metrics have become a concern for many researchers. They measure different aspects of software and therefore play an important part in *understanding, controlling and improving* software quality.

A lot of metrics have been proposed so far and new metrics continue to appear in the literature regularly. In spite of their great number, there are still some relevant issues raised. Many of these issues will be tackled as they are relevant for this work.

Very often definitions of metrics are *incomplete*, *ambiguous* and *open to a variety of different interpretations* [BDW99, Mar97, MP08]. For instance, the definition of NOM metric given by Li and Henri [LH93b] sounds like this: "the number of local methods of a class". The problem is that they do not explain the term "local method". He makes no reference about inherited methods neither about class methods or re-defined methods. Has the method's visibility (public, private etc.) been considered? Many of these questions haven't been answered yet; however this metric was validated by their authors as a predictor for maintenance effort [Rei01].

Of greater importance is the issue related to *the interpretation of results' measurement* rather than the definition of the metrics. In most cases there are no interpretation models or they are very empirical, so that applicability and reliability of measurement results is seriously hampered. A very long standing issue related to interpreting measurements' results of any metrics-based approach is that of *setting the metrics threshold*

values [Mar02]. So, what are the "correct" threshold values to be used and how can they be determined?

As metrics is a widely elaborate issue in any software domain, our research shall be limited to approaching fields such as Object Oriented Design (OOD) and Component Based Systems (CBS).

The goal of this work is that of investigating the way metrics can be applied to the assessment of software systems, mainly to the assessment of Object-Oriented Design and of Component-Based Systems. It will also be looking at methods and techniques with a view to offering a relevant interpretation of the obtained measurement results.

This thesis focuses on the activity of metrics based software assessment. It contains 132 bibliographical references and is divided into seven chapters (an introduction, a background chapter, four chapters containing original contributions and a concluding one) as follows.

Chapter 1 introduces the context, motivation and goals of this work. It summarizes also the contributions brought within it and provides an outline of its contents.

Chapter 2 introduces the background to be further used in this dissertation. It covers the fundamental concepts of the domain the thesis is related to (object-oriented paradigm and component-based systems), the key concepts of software measurement, a state of the art in the measurement field and quality assurance related to object-oriented design and component-based systems. The current chapter also presents the fuzzy clustering analysis method, used in measurement results interpretation. After pointing out the limitations of the current existing approaches, the chapter defines the goals and requisites of the thesis.

Chapter 3 introduces a new conceptual framework for object-oriented design assessment. The framework defines four abstraction layers that are used in object-oriented design assessment: the Object-Oriented Design Model – formally defines the assessment domain, specifying the elements to be evaluated, their properties and the relationships between them; Formal Definitions of OOD Metrics – consisting of a library of OOD metrics definitions; Specifications of the Assessment Objectives – that specifies in a formal manner the assessment objectives using a metrics based approach; Measurement Results Analysis – uses the fuzzy analysis method in order to interpret the measurement results obtained in the assessment process.

Chapter 4 evaluates the methods and techniques introduced in the previous chapter, based on a relevant case-study and presents the toolkit intended to support the automation of the design assessment methodology introduced in Chapter 3. It also includes a comparison with related approaches that use threshold values for software metrics, that are found in the literature;

Chapter 5 proposes an evaluation framework for component based systems, framework used in the problem of component selection with the aim of assessing the components for building a software system that meets the client's requirements. The approach defines a metamodel for CBS used as a contextual input in order to formally define metrics and to specify the assessment objectives. Additionally, it contains a robust method, based on fuzzy analysis, for the interpretation of the obtained measurement results. The steps for applying the evaluation model are also described. The proposed model is similar to the conceptual framework introduced in Chapter 3, but it has a different granularity level, of software components instead of that of classes.

Chapter 6 proposes an evaluation of the components assembly. In this chapter the system coupling is studied, because it captures the interactions between the components and is related to quality attributes. The selection of metrics is made and new metrics are proposed for quantifying the interaction issues between the components, by studying the influence of metrics values on quality attributes. Also, the assessment in question offers support in selecting the optimum solution from a set of possible solution configurations.

Chapter 7 concludes our work and offers an overview of all contributions reported within it.

The personal contributions introduced by this thesis are included in Chapters 3, 4, 5, and 6. They are divided into two research areas for the assessment of Object-Oriented Design and Component-Based Systems, and can be summarized as follows.

Contributions to object-oriented design assessment

- A new formalization of object-oriented design model [Ser09a] (Section 3.1). This formalization defines the theoretical background of OOD evaluation, to be used at formally defining metrics, stating the assessment objectives and interpreting the obtained assessment results.
- A new library of object-oriented design metrics formal definitions [Ser09a, Ser10] (Section 3.2). Metrics are formally defined by using the context delineated for the OOD model and expressing them in terms of algebraic sets and relations, knowledge assumed as familiar since the first stages of our studies.
- A new formalism for the specification of the assessment objectives (Section 3.3). The assessment objectives are specified using a metrics based approach.
- A new method for the interpretation of measurement results based on fuzzy clustering techniques [SP08] (Section 3.4). Fuzzy clustering analysis is used to overcome the limitations of the existing approaches which use threshold values for metrics.
- A new metric regarding design flaw entropy distribution [Ser09b, Ser09c] (Section 3.4.2).
- A complete framework for object-oriented design assessment [Ser10] (Chapter 3).
- A new case-study which experimentally validates the proposed methodology regarding object-oriented design assessment [Ser11, Ser09b, SP08] (Chapter 4). It also includes a comparison with related approaches to be found in literature, approaches that use threshold values for software metrics.

Contributions to component-based systems assessment

- A new method for component selection based on metrics and fuzzy clustering analysis [SVP08] (Section 5.2).

- A new component selection algorithm based on metrics and fuzzy clustering analysis [SVP09] (Section 5.3).
- A new formalization for the definition of metrics for component based systems [SVP10a] (Section 5.4.3).
- A new method for the interpretation of measurement results based on fuzzy clustering techniques [SVP10b] (Section 5.4.4).
- A complete framework for component-based system assessment [SVP10b] (Section 5.4). The conceptual framework formally addresses the problem of components' evaluation, including a model for CBS assessment, model that offers the contextual input for formal definitions of metrics and for formal specification of the assessment objectives; the steps involved in applying the proposed model are also included in this framework;
- A new formal approach of component assembly as a graph [SV07b] (Section 6.1.2).
- A set of new metrics to quantify quality attributes of component assembly [SV07b] (Section 6.1.3).
- A new set of metrics to best select a component assembly from a set of configurations [SV07a] (Section 6.2.2).
- A study of the influence of metrics values on quality attributes [SV07a] (Section 6.2.3).

2 Setting the context

This work is going to tackle the issue of using measurement in two major fields of software engineering: object-oriented design and component-based systems. Thus, the current chapter presents the fundamental concepts of these domains, given a particular emphasis to the manner in which measurement supports research in the software engineering area.

2.1 Software Measurement

Fenton's conceptual framework for measurement [Fen95] is widely accepted by the software engineering community and we believe it provides adequate support for this research.

2.2 Metrics in Object Oriented Design Assessment

Evaluation of software system design is a time-consuming activity due to its complexity. Therefore, special methods and techniques are needed in order to assess the design in an automatic manner. Software metrics are consequently an alternative solution, being a means for quantifying those aspects considered important for the assessment.

2.2.1 Object Oriented Design

In order to meet quality attributes an object-oriented application has to follow different *design principles* [Rie96, JF88], *design heuristics* [Mey88, Mar] and *design patterns* [GHJV94]. Object-oriented systems that does not observe to rules that ensure a good design and which were mentioned above, encounter a lot of problems which hamper their evolution.

Riel [Rie96] proposes a set of heuristic design guidelines and argues on some of the flawed structures that result when these guidelines are broken. Fowler in [FBB+99] completes with some deviations from a good design and their consequences. In [Mar] Martin comments on object orientation main design principles arguing that their breakdown causes a rotting design.

2.2.2 Metrics for Object Oriented Design

The literature offers a significant number of object-oriented metrics. Of these, the metrics proposed by Abreu [Abr93, AR94], those of Chidamber and Kemerer [CK94], Li and Henry [LH93a], the MOOD metrics [Abr95], Lorenz and Kidd [LK94] metrics are the most widely used. Marinescu [Mar02] has classified these metrics according to four internal characteristics that are essential to object-orientation: - i.e. *coupling, inheritance, cohesion and structural complexity.*

Several authors [Mar02, MP08, Rei01, BBA02, MSL06, BDW99, WBD98] discussed some important problems encountered when a metrics based approach is used in software assessment. The most important of these issues are related with *imprecise definition of metrics* and *measurements results interpretation.*

2.3 Metrics in Component-Based Development

At present, Component-Based Software Development (CBSD) has been accepted as a new effective development paradigm in the industry. It brings forward the design and construction of a software system by selecting and integrating appropriate components, previously developed, and then assembling them to obtain the functionality expected.

2.3.1 Component Definition and Specification

Several definitions of a component are found in literature [Spa00, Mic, DW98, Szy98] over time. Although these definitions may contain dissimilarities, they generally state that a component is a software module providing functionality through well specified interfaces.

2.3.2 Metrics for component-based development

When building a component-based software system, there may be different components to provide the same needed functionality. In order to differentiate components with similar functional properties, knowledge about quality attributes is generally regarded as information of utmost importance, if specified. As a result in this direction, software metrics are very important, being a mean to quantify those aspects that are considered important for the assessed entities (software components) in order to obtain high quality solution [SKG07]. On the other hand, software metrics can be used to assess a system solution in order to select beetwen alternative solutions.

The realm of traditional software metrics are not applicable to component based systems. This is due mainly to the black-box nature of a component [GG03a]. Gill [GG03a] proposes some aspects that need to be considered when we define component based metrics.

2.4 Fuzzy Analysis in Measurement Results Interpretation

As we have argued before software metrics pose some problems, the most important being the ability to give relevant interpretation of the measurement results which in turn is due to the fact that threshold values for the metrics are difficult to set. In order to overcome this limitation, we propose an alternative approach for the problems regarding measurement results interpretation and setting up the software metrics threshold values. The proposed approach uses fuzzy clustering analysis. This allows us to place the assessed software entity in more than one group, with different membership degrees.

In our approach, the objects considered to be clustered are the design entities of the software that we aim to assess or the software components selected in order to build a component based system. The attributes or the relevant features of the analyzed objects are the corresponding values of the metrics selected for the assessment.

3 A conceptual framework for Object Oriented Design Assessment

The current chapter is focused on developing a methodology for quantitative evaluation for object-oriented design. The methodology we are proposing is based on static analysis of the source code and is described by a conceptual framework of four abstraction layers: Object-Oriented Design Model, Formal Definitions of OOD Metrics, Specifications of the Assessment Objectives, Mesurement Results Analysis. These elements are briefly described in the next sections.

3.1 A model for object oriented design

The OOD Model proposed by Marinescu [Mar02] is formalized in our approach for design assessment using terms of algebraic sets and relations.
Definition 3.1.1 ([Ser09a, Ser10]) *The 3-tuple $D(S) = (E, Prop(E), Rel(E))$ is called a model for an object-oriented design corresponding to a software system S, where:*

- *E represents the design entities set of S;*
- *$Prop(E)$ defines the properties of the elements from E;*
- *$Rel(E)$ represents the relations between the design entities of the set E.*

The elements $E, Prop(E), Rel(E)$, referred in Definition 3.1.1 will be detailed in what follows.

3.1.1 Design entities

Let $E = \{e_1, e_2, ..., e_{noE}\}$ be the set of *design entities* of the software system S, where $e_i, 1 \leq i \leq noE$ can be a *package*, a *class*, a *method* from a class, an *attribute* from a class, a *parameter* from a method, a *local variable* declared in the implementation of a method or a *global variable*. Thus, the design entities set is defined by the following formula:

$$Design\ Entities = Package(E) \cup Class(E) \cup Meth(E) \cup Var(E), \text{ where:}$$

$$Var(E) = Attr(E) \cup Param(E) \cup LocVar(E) \cup GVar(E).$$

Each notations are described in details in the thesis.

3.1.2 Properties of design entities

As we have mentioned before, the second element of our model is the set of properties of the design entities, denoted by $Prop(E)$. Because current approach refers to six types of design entities: class, method, attribute, parameter, local variable, global variable, each type having its own set of properties, we define a model in order to specify the properties of entities of a generic type T. Then, we apply this model for our concrete types of design entities enumerated above.

As an example, we apply this model to specify the properties for design entities of type class, the abbreviation used for this type is C.

Definition 3.1.4 ([Ser09a, Ser10]) *The 4-tuple* $Prop_{C,Class(E)} = [C, Class(E), Prop_C, PropVal_C]$ *is called specification of properties for entities of type "class", where:*

- $Prop_C = \{Abstraction, Visibility, Reusability\}$;
- $Abstraction = \{concrete, abstract, interface\}$;
- $Visibility = \{package, inner, public\}$;
- $Reusability = \{user - defined, user - extended, library\}$.

3.1.3 Relations between design entities

In this section we summarize the type of relations that exist between the entities of object oriented design model. We mention here that for each entity, we consider only those relations in which the entity directly interacts with other entities.

Inheritance relations between classes. Consider $a, b \in Class(E)$. There are two types of direct relations among classes:

- "**a extends b**", if class a is a specialization of class b (class a inherits the structure and behavior of class b);
- "**a implements b**", if class a is an implementation of the interface class b (class a implements the behavior of the interface class b).

Method invocation relation. Coupling metrics for a class c can be defined, on condition we know the set of methods called by every method $m \in Meth(E)$ and the set of variables referenced by any method of the class c [Mar02]. Briand in [BDW99] has introduced the definition of method call relation and this is adapted to our thesis framework.

Attributes reference relation. Attributes may be referenced by methods. As attribute references are not determined dynamically, it is enough to consider the static type of the object for which an attribute is referenced.

3.2 Formal definition of OOD Metrics

As a proof of concept regarding the applicability of our proposed model for OOD, we present the definitions of metric *Method Coupling – MC*.

Definition 3.2.1 ([RL92]) ***Method Coupling – MC***

Informal statement. *MC metric is defined as "the number of non-local references" in a method.*

Formal Definition. $\forall c \in Class(E)$, $\forall m_0 \in Meth(c)$, $MC(c, m_0)$ *is defined as follows:*

$$MC(c, m_0) = card(A \cup B), \ where:$$

$$A = \{m \in Meth(E) - Meth(c) | m_0 \ call \ m\})$$

$$B = \{a \in ARefRel(m) | a \notin Attr(c)\}$$

Comments. *"A non-local reference" is defined by the authors as that reference of method or variable which are not defined in the class to which the method belong.*

3.3 The problem of design assessment. Setting the objectives

Object-oriented software assessment is aimed at verifying whether the built system meets quality factors such as maintainability, extensibility, scalability and reusability. Fenton's axiom [Fen94] states that good internal structure should provide good external quality. According to this axiom, the assessment objectives are reduced to verifying if there is conformity between the software system internal structure and the principles and heuristics of good design, which are related with the internal quality attributes of the system design (such as coupling, cohesion, complexity and data abstraction).

The community of researchers was interested in setting a relation between the principles of good design with the design flaws. They wanted to seek what the violated principles or rules were for a certain design flaw or vice versa, what were the design flaws that could propagate in code if a design principle was violated.

Our objective is to develop a quantitative assessment approach and for this reason we also have to correlate design flaws or design principles with metrics that quantify those design flaws or design principles. Consequently, the evaluation objectives may be defined by two approaches:

1. In the first approach, we start from a set of design principles, for each of them we identify the corresponding design flaws, and for each design flaw a set of appropriate metrics; this approach will be referred in what follows as Principle-Flaw-Metric Objectives Specification;

2. In the second approach, we start from a set of design flaws, for each of them we identify the corresponding design principles, and than for each design principle we establish a set of appropriate metrics; this approach will be reffered in what follows as Flaw-Principle-Metric Objectives Specification.

The problem of design assessment

Informally, *the problem of design assessment or design assessment problem (DAP)* is to identify a list of design entities, called "suspects" which capture deviations from a specified design principle/heuristic/rule or are affected by a specified design flaw. The notations used for formally defining DAP are described in the thesis.

3.4 Assessment Results Analysis

We have already stated in the previous chapter that the most critical step in any assessment activity is probably the result analysis or, to put it otherwise, the interpretation of measurement results. This is because this activity cannot be fully automatized, like for instance that of metrics computation. Interpretation is based on specificities of the analyzed software system.

For the interpretation of measurement results, Marinescu used an automated mechanism, called "detection strategy" which combines different code metrics, filters the result and associates this information in order to detect problems in the design architecture. He defined a detection strategy as: "the quantifiable expression of a rule by which design fragments that conform to that rule can be detected in the source code" [Mar02].

A limitation of this approach is that of choosing proper threshold values for metrics, which is not addressed. In order to overcome this limitation, we propose a new approach for measurement result interpretation based on fuzzy clustering analysis. Thus, we started from the approach proposed by Marinescu, using the first part of so called detection strategy which correlates design flaw or design principles with metrics, and instead of the filtering and composition mechanisms we propose fuzzy analysis with the main goal of avoiding the thresholds for metrics.

3.4.1 Using fuzzy clustering analysis in the interpretation of measurement results

Consider the problem of design assessment previously mentioned. The results of the assessment are a set of design entities that were evaluated AE_p, together with their corresponding values of the selected metrics $M_p = \{m_1, m_2, ..., m_{noM_p}\}$. Formally, the results of the assessment can be described as follows:

- $AE_p = \{e_1, e_2, ..., e_n\}$, the assessed entities set $AE_p \subseteq AE$,
- $e_i = (e_{i1}, e_{i2}, ..., e_{i(noM_p)})$, the corresponding values of metrics $m_1, m_2, ..., m_{noM_p}$, $m_j \in M_p$, $1 \leq i \leq n$, $1 \leq j \leq noM_p$,
- $(e_{ij})_{i=\overline{1,n};j=\overline{1,noM_p}}$, *the assessment results matrix.*

Based on the metrics values, we will select from AE_p the *suspect* entities. Fuzzy clustering analysis is used to partition the entities in clusters, each entity having different membership degree to a cluster.

Definition 3.4.1 ([Ser10]) *A set $U_{AE_p,M_p} = \{U_1, U_2, ..., U_c\}$ is called a fuzzy partition of the design entities set $AE_p = \{e_1, e_2, ..., e_n\}$, entities characterized by the values of metrics the $M_p = \{m_1, m_2, ..., m_{noM_p}\}$ iff:*

- $U_i = (u_{i1}, u_{i2}, ..., u_{in})$, $1 \leq i \leq c$;
- $u_{ij} \in [0..1]$, $1 \leq i \leq c$, $1 \leq j \leq n$, u_{ij} *represents the membership degree of the design entity e_j to cluster i;*
- $\sum_{i=1}^{c} u_{ij} = 1$, $1 \leq j \leq n$, *the sum of each column of U is constrained 1.*

The fuzzy clustering genetic algorithm applied to obtain this partition, named Fuzzy c-means clustering, is described in [Bez81]. This algorithm has the drawback that the optimal number of classes corresponding to the cluster substructure of the data set, is an input data. To avoid this drawback, we considered the Fuzzy Divisive Hierarchic Clustering (FDHC) algorithm [Dum88]. The FDHC algorithm produce a binary tree hierarchy that provides an in-depth analysis of the data set, by deciding on the optimal number of clusters and the optimal cluster substructure of the data set. The partitions obtained by applying the FDHC algorithm are presented below.

Definition 3.4.2 ([Ser10]) *A set $BTFP_{AE_p,M_p} = \{N_1, N_2, ..., N_l\}$ is called binary tree fuzzy partition of the design entities set $AE_p = \{e_1, e_2, ..., e_n\}$, entities characterized by the corresponding values of metrics $M_p = \{m_1, m_2, ..., m_{noM_p}\}$ iff:*

- $\forall i \in \{1,2,...,l\}$, $N_i = (f_i, U_i)$, *where* f_i *is the father of* i^{th} *cluster,* $f_i \in \{0...l\}$ *and* $U_i = (u_{i1}, u_{i2}, ...u_{in})$, $u_{ij} represents the membership degree of the design entities e_j$ *to cluster* i, $u_{ij} \in [0..1]$, $j \in \{1,2,...,n\}$;
- $\exists!\ i \in \{1,2,...,l\}$ *such that* $f_i = 0 \wedge u_{ij} = 1, \forall j \in \{1,2,...,n\}$, N_i - *the root node;*
- $\forall i \in \{1...l\}$, $f_i \neq 0$, $\exists! j,k$ *such that* $f_i = k \wedge f_j = k \wedge U_i + U_j = U_k$, *where* $U_i + U_j = (u_{i1} + u_{j1}, u_{i2} + u_{j1}, ..., u_{i(n)} + u_{j(n)})$.

Definition 3.4.3 ([Ser10]) *Consider a binary tree fuzzy partition* $BTFP_{AE_p,M_p} = \{N_1, N_2, ..., N_l\}$. *A node* $N_i = (f_i, U_i) \in BTFP_{AE_p,M_p}$ *is called terminal or leaf node iff:*

$$\forall j \in \{1,2,...,l\}, j \neq i \text{ and } N_j = (f_j, U_j) \in BTFP_{AE_p,M_p} \Rightarrow f_j \neq i.$$

Definition 3.4.4 ([Ser10]) *Consider a binary tree fuzzy partition* $BTFP_{AE_p,M_p}$. *A subset* $OFP_{AE,M} = \{G_1, G_2, ..., G_k\} \subset BTFP_{AE_p,M_p}$ *is called optimal fuzzy partition of the design entities set* AE_p *with respect to a set of software metrics* M_p *iff:* G_j *is a terminal(leaf) node from* $BTFP_{AE_p,M_p}$, $\forall j \in \{1,2,...,k\}$.

The next section introduces a new metric, called *Design Flaw Entropy (DFE)*, which measures the dispersion of analyzed design principles or design flaws on a set of design entities.

3.4.2 Design Flaw Entropy Metric

DFE metric is defined considering the notion of *entropy* adapted from communication information theory of Shannon [SW49]. Starting from this concept many researchers [EGH02, BDE99, MBA04, KSW95] have developed new measures for the assessment of software products.

Let us consider a fuzzy partition $U_{AE_p,M_p} = \{U_1, U_2, ..., U_c\}$ of the design entities $AE_p = \{e_1, e_2, ..., e_n\}$, entities characterized by the values of metrics $M_p = \{m_1, m_2, ..., m_{noM_p}\}$, metrics selected in order to quantify a specified design principle or design flaw p.

Definition 3.4.5 ([Ser09b]) *We say that an entity* $e_j \in AE$, $1 \leq j \leq n$, *have dominant membership degree to cluster* U_i, $1 \leq i \leq c$, *if* $u_{ij} = \max\{u_{rj} | r = \overline{1,c}\}$.

Definition 3.4.6 ([Ser09b]) *The relative frequency of occurrence or the probability of a cluster* $U_i \in U_{AE_p,M_p}$, *denoted by* $p(U_i)$, *represents the ratio between the number of entities from* AE_p *that have dominant membership degree to cluster* U_i *and the total number of entities from* AE_p.

We will denote by $P_{U_{AE_p,M_p}} = \{p(U_1), p(U_2), ..., p(U_c)\}$ the probability distribution per clusters of the partition U_{AE_p,M_p}.

Definition 3.4.7 ([Ser09b]) *A measure of the information (self-information) contained in a cluster* $U_i \in U_{AE_p,M_p}$ *is defined as* $I(U_i) = -\log_2 p(U_i)$.

In the context of the previous definitions and notations, we can now introduce the definition of the proposed metric.

Definition 3.4.8 ([Ser09b]) ***Design Flaw Entropy (DFE) Metric***.
Design Flaw Entropy (DFE) corresponding to fuzzy partition U_{AE_p,M_p} *is defined as the*

average of the self-information associated to each cluster $U_i \in U_{AE_p,M_p}$. Formally:

$$DFE : FP(AE_p, M_p) \to [0..\infty],\ DFE(U_{AE_p,M_p}) = \sum_{i=1}^{c} p(U_i) \cdot I(U_i)$$

where $FP(AE_p, M_p)$ is the set of all fuzzy partitions of the design entities set AE_p, entities characterized by the values of metrics $M_p = \{m_1, m_2, ..., m_{noM_p}\}$, metrics selected in order to quantify a specified design principle or design flaw p.

As we have mentioned earlier, this metric measures the distribution or the variety of the analyzed design principle or design flaw p on a set of design entities AE_p. It also offers a deep analysis in the interpretation of measurement results regarding the assessment of an object-oriented design $D(S) = (E, Prop(E), Rel(E))$ corresponding to a software system S.

3.5 Conclusions

In the current chapter we have proposed an 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. We have approached two major issues regarding object oriented design assessment and the proposed solutions have provided. The chapter is based on the papers [Ser09a, Ser10, Ser09c, Ser09b, SP08].

4 Experimental Evaluation of the Proposed Model

The current chapter presents the experimental evaluation of our proposed model for OOD assessment introduced in Chapter 3 and the tool developed by us that supports the automation of metrics computation and fuzzy partitions determination.

4.1 Design Flaws Detection. Case - Study

Our starting point, i.e., the input data, is the source code of an open source object-oriented software system, called log4net [log]. It consists of 214 classes grouped in 10 packages.

4.1.1 Domain Assessment Identification

We parse the source code with our own developed tool and produce the domain of the assessment $D(log4net) = (E, Prop(E), Rel(E))$, an object-oriented model of the software system *log4net*.

4.1.2 Setting the Assessment Objectives

The objective of the proposed assessment is to identify those design entities affected by "God Class" [FBB+99] and "Shotgun Surgery" [FBB+99] design flaws. Consequently, the assessed entities AE are the set of classes from our system. In what follows, we aim to identify the other two components from the assessment objectives specification, $AO_2(log4net) = (AE, FPG, PMG)$, the elements of flaws-principles graph FPG and the elements of principles-metrics graph PMG.

God Class Design Flaw. To detect a God Class, Salehie et al. [MSL06] have related this design flaw to a set of three heuristics from Riel's book [Rie96]: *distribute system intelligence horizontally as uniformly as possible*, *beware of classes with much non-communicative behavior*, *beware of classes that access directly data from other classes*.

The selected heuristics are then related with the following metrics: Weighted Method per Class (WMC) [CK94], Tight Class Cohesion (TCC) [BK95] and Access to Foreign Data (ATFD) [Mar02].

Analyzing the definitions of these metrics, we can conclude that a possible "God Class" suspect will have *high* WMC and ATFD metric values and *low* TCC metric values. Fuzzy clustering analysis is used to solve the problem of setting software metric threshold values. Thus, an entity may be placed in more than one group, with various membership degrees.

Shotgun Surgery. 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 supect of Shotgun Surgery design flaw [FBB+99]. In brief, this design flaw approaches the issue of "strong implementation coupling" [Mar02].

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 [Mar02] have identified three potential "victims" of changes in a class: *methods directly*

accessing an attribute that has changed, methods calling for a method whose signature has changed, methods which override a method whose signature has changed.

Therefore, the selected metrics [Mar02] which quantify the above mentioned aspects are Changing Methods (CM) [Mar02], Weighted Changing Methods (WCM) [Mar02] and Changing Classes (CC) [Mar02].

In order to better emphasize the specification of the assessment objectives regarding the analyzed design flaws, Figure 1 describes the above mentioned sets.

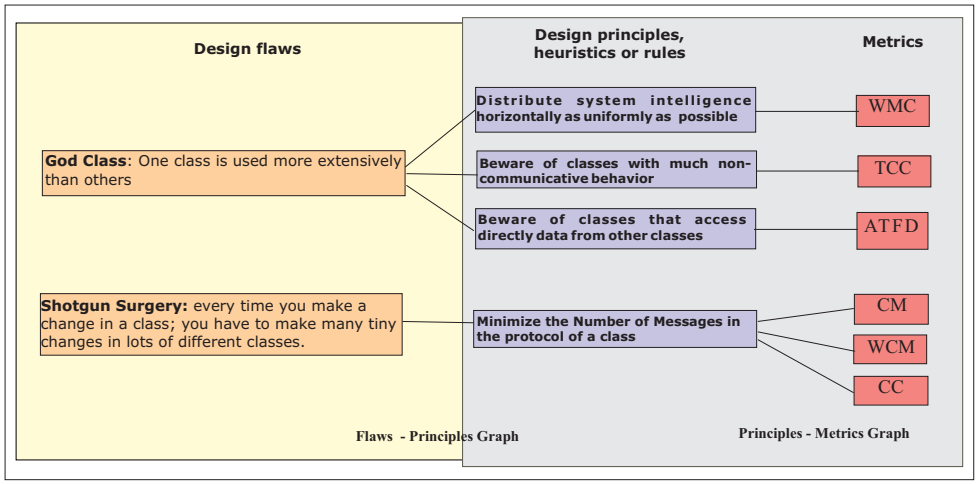

Figure 1: Assessment objectives specification for the proposed case-study.

4.1.3 Formal Definitions of Selected Metrics

The formal definitions of metrics used in this case study are presented in the thesis.

4.1.4 Fuzzy Partitions Determination. Results Analysis for "God Class" Design Flaw

The third step in the proposed evaluation consists of applying the fuzzy clustering analysis with the goal of identifying those design entities affected by "God Class" design flaw. The manner in which we apply the fuzzy clustering analysis method can be described in two steps as follows:

- *Binary tree fuzzy partition determination* using Fuzzy Divisive Hierarchic Clustering (FDHC) algorithm [Dum88].
- *The analysis of the obtained partitions* in order to determine those clusters that contain suspects entities.

A comparison with similar approach proposed by Marinescu [Mar02], based on detection strategies, is proposed in Section 4.1.4.3.

4.1.5 Fuzzy Partitions Determination for "Shotgun Surgery" Design Flaw

While the previous section deals with fuzzy partition determination for "God Class" design flaw, the current one aims to determine the fuzzy partition related with the second assessment objective that of "Shotgun Surgery" design flaw and starting from this partition our goal is to identify "suspect" design entities. In order to attain this goal, we proceed in the same manner as in "God Class" case.

A comparison with similar approach proposed by Marinescu [Mar02], based on detection strategies, is proposed in Section 4.1.5.3.

4.2 Tool Support

In order to reduce the time spent on design assessment we need tools that automatize the problem of design flaws. In this respect, we have developed a tool called Metrics.

4.3 Conclusions

In this chapter we have presented a new case-study in order to experimentally validate the theoretical methodology regarding object-oriented design assessment, methodology introduced in Chapter . The proposed case-study addressed the issue of "God Class" and "Shotgun Surgery" design flaws detection and it is based on metrics and on fuzzy clustering techniques. The chapter is based on the paper [Ser11].

5 Metrics-Based Approach for Component Selection

This chapter addresses the problem known in literature as the Component Selection Problem (CSP). Informally, our goal is to select a subset of components from the available component set satisfying the system requirements. Selecting the best component alternative among components that sometimes provide similar functionalities needs to appeal to knowledge on quality attributes which often is the most significant information for this operation. So, evaluation of these components is of utmost importance and software metrics are a means to quantify those attributes considered important for the system that will be built.

5.1 Component selection problem. Formal statement

An informal definition of a Component Selection Problem is given below: considering a repository of components and a specification of the component system to be built (set of system requirements), components should be chosen and connected in such a way that component system resulted fulfills the specification. A system requirement is in fact a service provided by a component.

In what follows, we will describe the notations used to formally define CSP, according to [FBR04, Ves08b] with a few minor changes. Consider $SR = \{r_1, r_2, ..., r_n\}$ the set of system requirements (target requirements) and $CR = \{c_1, c_2, ..., c_m\}$ the repository of components available for selection. Each component $c \in CR$ is specified as a set of provided interfaces (functionalities, services) $PI_c = \{pi_{c,1}, pi_{c,2}, ..., pi_{c,noPI_c}\}$, a set of required interfaces $RI_c = \{ri_{c,1}, ri_{c,2}, ..., ri_{c,noPI_c}\}$ and the dependencies (context) between the services provided and required by component c. A dependency states that in order to offer a service, a component requires a service from other component.

The goal is to find a set (subset) of components Sol in such a way that to every requirement r from the set SR, a component c from Sol can be assigned, where r is in PI_c.

5.2 Metrics-based Component Evaluation. Theoretical Background

Let us consider the component selection problem defined in previous section. As we already mentioned, selecting the best component alternative among components that sometimes provide similar functionalities needs to appeal to knowledge on quality attributes which often is the most significant information for this operation. So, evaluation of these components is of utmost importance and software metrics are a means to quantify those attributes considered important for the system that will be built. Based on the interpretation of the measurements results obtained, we can decide what component best satisfies the system need.

The current research proposes a new method for components selection, which is based on metrics and fuzzy analysis.

5.3 A New Component Selection Algorithm based on Metrics and Fuzzy Clustering Analysis

5.3.1 Algorithm Description

The current section presents a new algorithm for constructing a software system by assembling components. In this context we address the problem of component selection. Informally, our problem is to select a subset of components satisfying the system requirements. The difficulty resides in the fact that each component had a related set of components that share similar functionalities and our goal is to select the best solution. Metrics are defined to quantify some important attributes of components (criteria used for selection). Fuzzy clustering analysis is used for components classification based on metrics values and an algorithm for the decision process is proposed.

Two alternative approaches are possible for the proposed algorithm:

- one that uses only one partition based on initial system requirements SR;
- the second alternative recomputes metrics based on the update of system requirements SR and reclassifies the candidate components at each step of a component selection (from a set of candidates).

This algorithm is described in the thesis in Section 5.3.1.

5.3.2 Comparative analysis of the obtained solutions by other approaches

To compare our approach with similar approaches found in literature, we have proposed a case study [SVP09] described in the thesis in Section 5.3.2. Table 1 comparatively presents the obtained solutions with all approaches. The solutions obtained by our proposed algorithm are comparable with the ones provided by the others approaches, the main advantages being: the search space dimension is drastically reduced, the obtained partition suggesting the component that should be selected at a given step; the execution time needed for selecting the best component is reduced due to the reduced search space; the selection criteria of the components are based on several characteristics of components (several metrics may be defined).

Algorithm	*Solution*	*cost*	*reusability*
Greedy	c_4, c_0, c_7, c_1	35	5
Branch and Bound	c_4, c_2, c_6, c_1	34	3
Genetic algorithm (only cost)	c_2, c_6, c_8	28	2
Genetic algorithm (only PSU and RSU)	c_0, c_7, c_8	29	4
MFbCSwSPA	c_0, c_1, c_5, c_8	32	5
MFbCSwCPA	c_2, c_6, c_8	28	2

Table 1: Obtained solutions using different approaches

5.4 A conceptual framework for CBS Assessment

The current section proposes a formal approach concerning component based systems assessment. More precisely, we aim to define a general, scalable and integrated framework for a quantitative evaluation regarding both individual components and the system obtained by connecting components. This framework is a formalization and a generalization of our previous work [SVP08, Ves09, SVP09, SVP10a]. It is composed of four layers of abstraction: A model for CBS, Assessment objectives, Formal definitions of metrics, A measurement results analysis method. These elements are briefly described in the next sections.

5.4.1 A model for Component-Based System

Every measurement activity have to be preceded by the specification of the elements that will be evaluated, their properties and relationship between them. These elements are grouped into a model of the analyzed system. In what follows, we will define a model for a component-based software system.

Definition 5.4.1 ([SVP10b]) *The 3-tuple* $(E, Prop(E), Rel(E))$ *is called a model for Component Based System, where:*

- E *represents the set of system entities;*
- $Prop(E)$ *defines the properties of the elements from* E*;*
- $Rel(E)$ *signifies the set of relations between the entities of* E *set.*

The elements $E, Prop(E), Rel(E)$, referred in Definition 5.3.1 will be detailed in what follows, using terms of algebraic sets and relations.

System entities. Consider a repository of components $CR = \{c_1, c_2, ..., c_{noCR}\}$. A subset of components $Comp(S) = \{c'_1, c'_2, ..., c'_{noComp}\}$, $Comp(S) \subseteq CR$ are selected in order to construct a software system S that has to provide a set of services $Serv(S) = \{s_1, s_2, ..., s_{noServ}\}$.

Each component $c \in CR$ is specified as:

- a set of *provided interfaces* $PI_c = \{pi_{c,1}, pi_{c,2}, ..., pi_{c,noPI_c}\}$;
- a set of *required interfaces* $RI_c = \{ri_{c,1}, ri_{c,2}, ..., ri_{c,noPI_c}\}$;
- the dependencies (context) between its interfaces provided and required.

Properties of system entities. We identify three types of system entities (component, interface and parameter). To specify their properties we use the model proposed in Section 5.4.1.2 in the thesis.

Relations between system entities. Two types of relations (dependency and connection) between the system entities are defined in this approach. These relations are described in the thesis in Section 5.4.1.3.

5.4.2 Assessment Objectives

When building a component-based software system two kinds of requirements must be fulfilled: firstly the system has to provide a set of functionalities/services and secondly it has to meet certain non-functional properties or quality attributes, such as security, performance and reliability. Thus, the main objectives in component based system assessment are related with these two kinds of requirements.

The objectives regarding system functionalities/services are simply to verify if they are attained by the final system. Therefore, this research thesis proposes a quantitative approach related to the nonfunctional assessment objectives, on the basis of Factor Criteria Metric (FCM) model [BR88]. Thus, each quality attribute is refined into several criteria each suggesting relevant metrics. Consequently, each quality attribute will be linked to one or more metrics that best capture its meaning.

It is easily noticeable that the quality of individual components influences, directly or indirectly, the quality of the final system. The compositional reasoning reveals that the system's properties are influenced to a greater extent by the interaction of its components than by the properties of a single component. A better and more worthy evaluation of components may be obtained in the context of the system in which they are to be integrated. As a consequence, the assessment objectives we propose are reported to the target product: the component assembly [GA04b].

Another issue related to the quality attributes is that they may influence each other in many different ways; for instance, the increase of one attribute (maintainability) can diminish another attribute (performance). This means that for each of the selected metrics, we have to establish a weight factor (wf) with the role of defining a priority in the component selection process. Thus, between two components that offer similar services it is possible to select a component with a low reusability value instead of a component with a high value of this attribute, if other quality attribute is more important for the client.

5.4.3 Formal Definition of Metrics

In the thesis in Section 5.4.3 are presented some metrics for CBD, their formal definitions are based on the model proposed in Section 5.4.1.

5.4.4 Mesurement Results Analysis

Formally, we described the results of the proposed assessment as follows:

- CR, the set of assessed entities;
- each component $c_i \in CR$ being identified as a vector $c_i = (c_{i_1}, c_{i_2}, ..., c_{i_{noM}})$ with the corresponding values of metrics $m_1, m_2, ..., m_{noM}$, $m_j \in Metrics(S)$, $1 \leq j \leq noM$.

An interpretation of these results is needed, in order to use them as input data in the component selection algorithm. As we have stated before with regard to the interpretation of measurement results there is an issue which resides in the difficulty to set threshold values for metrics. In order to overcome this limitation we have used fuzzy clustering analysis. This allows us to place an object in more than one

group, with different membership degrees, giving us the possibility to take into account, when choosing between two similar components, some specific elements of the analyzed system by reducing the rigidity of the threshold values.

In what follows, we will introduce some definitions, in order to emphasize how we apply the fuzzy analysis in the proposed assessment.

Definition 5.4.3 *A set* $U = \{U_1, U_2, ..., U_k\}$ *is called a fuzzy partition of a set of components* $CR = \{c_1, c_2, ..., c_{noComp}\}$*, each component* $c_i \in CR$ *being identified as a vector* $c_i = (c_{i_1}, c_{i_2}, ..., c_{i_{noM}})$ *with corresponding values of metrics* $m_1, m_2, ..., m_{noM}$*,* $m_j \in Metrics(S)$*,* $1 \leq j \leq noM$*, iff:*

- $U_i = (u_{i1}, u_{i2}, ..., u_{i(noComp)})$*,* $1 \leq i \leq k$*;*
- $u_{ij} \in [0..1]$*,* $1 \leq i \leq k$*,* $1 \leq j \leq noComp$*,* u_{ij} *represents the membership degree of the component* c_j *to cluster* i*;*
- $\sum_{i=1}^{k} u_{ij} = 1$.

Due to the fact that the number of clusters is an input data for a partitional clustering algorithm, we considered the Fuzzy Divisive Hierarchic Clustering (FDHC) algorithm [Dum88]. This algorithm produces a binary tree fuzzy partition that provides an in-depth analysis of the data set, deciding the optimal subcluster cardinality and the optimal fuzzy partition of the data set. In the following we formally define these two partitions.

Definition 5.4.4 ([SVP10b]) *A set* $BTFP = \{N_1, N_2, ..., N_l\}$ *is called a Binary tree fuzzy partition of a components set* $CR = \{c_1, c_2, ..., c_{noComp}\}$*, each component* $c_i \in CR$ *being identified as a vector* $c_i = (c_{i_1}, c_{i_2}, ..., c_{i_{noM}})$ *with corresponding values of metrics* $m_1, m_2, ..., m_{noM}$*,* $m_j \in Metrics(S)$*,* $1 \leq j \leq noM$*, iff:*

- $\forall i \in \{1, 2, ..., l\}$*,* $N_i = (f_i, U_i)$*,* f_i*-the father of* i^{th} *cluster,* $f_i \in \{0...l\}$*;* $U_i = (u_{i1}, u_{i2}, ..., u_{i(noComp)})$*,* $u_{ij} \in [0..1]$*,* u_{ij} *represents the membership degree of the component* c_j *to cluster* i*;*
- $\exists i, i \in \{1, 2, ..., l\}$ *such that* $f_i = 0 \wedge u_{ij=1}, \forall j \in \{1, 2, ..., noComp\}$*,* N_i *- the root node (cluster);*
- $\forall i \in \{1...l\}$*,* $f_i \neq 0$*,* $\exists!\ j, k : f_i = k \wedge f_j = k \wedge U_i + U_j = U_k$*, where* $U_i + U_j = (u_{i1} + u_{j1}, u_{i2} + u_{j1}, ..., u_{i(noComp)} + u_{j(noComp)})$.

Definition 5.4.5 ([SVP10b]) *Consider a binary tree fuzzy partition* $BTFP = \{N_1, N_2, ..., N_l\}$*. A node* $N_i = (f_i, U_i) \in BTFP$ *is called terminal or leaf node iff:*

$$\forall j \in \{1, 2, ..., l\}, j \neq i \text{ and } N_j = (f_j, U_j) \in BTFP \Rightarrow f_j \neq i.$$

Definition 5.4.6 ([SVP10b]) *Consider the binary tree fuzzy partition* $BTFP$*. A subset* $OFP = \{G_1, G_2, ..., G_k\} \subset BTFP$ *is called an optimal fuzzy partition iff:* G_j *is a terminal(leaf) node from* $BTFP$*,* $\forall j \in \{1, 2, ..., k\}$.

Starting from the fuzzy partitions which are described by the definitions above, in our previous work [SVP09] we have constructed an algorithm for component selection problem. This algorithm is presented in the thesis in Section 5.3.

5.5 Conclusions

In this chapter we have proposed a new metrics based approach concerning component-based systems assessment. The proposed approach aims to overcome the problems encountered in a software measurement activity:

- *metrics definition*; the metrics that can be expressed using our framework have definitions that are *unambiguous*, *simple* and *language independent*.
- *measurement results interpretation*; fuzzy clustering analysis is used in order to overcome the problem of setting up the software metrics threshold values and to select appropriate components.

Some of the original results presented in this chapter have been reported in the papers [SVP08, SVP09, SVP10a, SVP10b].

6 Components Assembly Evaluation

The previous chapter introduced the issue of metrics based component selection with the aim of obtaining a system solution to better satisfy customer requirements. The solution obtained by us has some limitations, for instance, it does not take into account the interactions between components, because we have full knowledge of these interactions only after the system has already been built. The overall quality of the assembly depends on the interaction among components. Consequently, this is topic of ongoing analysis in order to assess the system as a whole, to assess its quality and to make comparisons between possible alternative solutions.

In this respect the current chapter proposes an evaluation, mainly focused on the components assembly. We have looked at the coupling of the system, because coupling captures the interactions between components and it is related with quality attributes. We have selected the metrics and proposed new metrics in order to quantify the interactions between components and study the influence of metrics values on quality attributes. The proposed assessment provides support for the selection of an optimum solution from a set of configurations representing possible solutions/candidates.

6.1 Components Assembly as a Graph

Definition 6.1.1 ([SV07b]) ***Components Assembly.*** *An **assembly** is a binary relation denoted by $DR = (C, D), D \subseteq C \times C$, where C is a set of components and D is the relation graph that contains the dependences between components. There is a component $c_0 \in C$ with a special role: to start the system execution.*

Definition 6.1.2 ([SV07b]) A **dependency** is a pair $d = (c_1, c_2) \in D$ with the meaning that the execution of c_1 needs some services provided by c_2 (in other words, c_1 depends of c_2).

We model a component based system (an assembly of components) as a directed graph (DR), in which the vertices are the components (the set C) and the edges are the dependences (the set D) between components. In this way we map each assembly to a directed graph.

Using directed graph view of the assembly is difficult to provide the depth and breath of the dependences between involved components. A better view implies the transformation of the directed graph into a tree. The dependency tree construction [SV07b] is described in detail in the thesis by **D**ependency **T**ree **A**lgorithm (**DTA**).

To obtained an optimal tree that contains all the paths from the directed graph representation, an additional algorithm that completes the tree is required. The proposed algorithm is called ***C**omplete **D**ependency **T**ree **A**lgorithm* (**CDTA**).

6.2 Adapted and defined metrics

In the following the existing metrics for object-oriented design [LK94, CK93] that were adapted for component assemblies are presented. Metrics for the component-based system hierarchy are proposed, based on the same new approach of component interaction. The new metrics help us to assess quality attributes of the system.

In an object-oriented design, coupling is "the interconnectedness between its pieces" [CY91]. By declaring an object of a remote class, a potential collaboration between

the two classes is created. This is measured by CBO metric [CK93].

We consider an assembly of components $DR = (C, D)$, where C is a set of components and D is the relation graph that contains the dependences between components.

Definition 6.2.1 ([SV07b]) *A component c_1* ***is coupled*** *with component c_2 if $(c_1, c_2) \in D$.*

Definition 6.2.2 ([SV07b]) ***Coupling Between Components (CBC)*** *metric measures the number of components with which a given component is coupled.*

$$CBC(c) = card(\{d \in C | (c, d) \in D\}), c \in C.$$

We study the coupling between components in order to assess the system quality. It is known that an excessive coupling has a negative impact many on external quality attributes [Mar02]: *reusability*, *modularity*, *understandability* and *testability*.

Definition 6.2.3 ([SV07b]) ***Depth Dependence Tree (DDT) metric.*** *Let us consider a component $c_n \in C$ and the corresponding elementary chain $c_0, c_1, ..., c_n$, where c_0 is the root node. The metric value is $DDT = n$. In other words, DDT measures the length chain dependences from a given component to the root.*

Definition 6.2.4 ([SV07b]) ***Breadth Dependence Tree (BDT)*** *metric represents the number of chains dependences from the root to all the leafs.*

These metrics are evaluated considering the impact on quality attributes. From this point of view, a high value of DDT metric makes the component difficult to be reused in a different context. In addition, understandability, maintainability and testability are also affected. Understanding of an entity is based on a recurrent understanding of all components the entity depends on. Moreover any change in a component may require changes in all components depending on this. Maintainability and some of its criteria (understandability) are influenced by the high value of these metrics. The system records a tendency to increase in complexity.

Subsection 6.1.4 contains a system example, PDA - Personal Digital Assistant, to study the applicability of the adapted and proposed metrics.

6.3 Metrics-based selection of a component assembly

When the construction of a system is based on a set of components, there are two approaches used:

1. building only the solution that meets the system requirements, the nearest possible to the best solution, by adopting heuristics methods;

2. building all possible configurations and then analyzing each of these assemblies in order to select the best solution to meet our goal.

While the previous chapter dealt with the first approach, the current chapter adopt the second solution. Software metrics based on the assembly-centric evaluation approach are used in the selection (from among the obtained assemblies) of the solution that best represents the system requirements.

6.3.1 Proposed Metrics

We have proposed the following two metrics for measuring coupling between components.

Definition 6.3.1 ([SV07a]) ***Component Coupling Grade.*** *The Component Coupling Grade (CCG) of a component X which is dependent on a component Y, represents the number of services provided by Y that X uses. In what follows we will denote this value by CCG(X,Y).*

Definition 6.3.2 ([SV07b]) ***Component Coupling Total Grade.*** *The Component Coupling Total Grade (CCTG) of a component X which is dependent by a set of components C_1,C_2,...,C_n, represents the number of services provided by all these components that X uses,*

$$CCTG = CCG(X, C_1) + CCG(X, C_2) + ... + CCG(X, C_n). \quad (1)$$

6.3.2 The influence of metrics values on quality attributes

We stated before that our aim is to define metrics that are relevant in measuring the quality attributes which we are interested in. We need these informations for choosing the solution that best represents the system requirements. Table 2 presents the influence of metrics values on the quality attributes which we consider important for the assembly evaluation. We use the following notations: m for metric low value, M for high value of the metric, $+$ for positive influence and $-$ for negative influence. For example a low value of IDC influences positively the reusability of the component.

	Reusability	Functionality	Understandability	Maintainability	Testability
PSU	m/+	m/-	m/+	m/+	m/+
RSU	m/+	-	m/+	m/+	m/+
CPSU	m/+	m/-	m/+	m/+	m/+
CRSU	m/+	-	m/+	m/+	m/+
IDC	m/+	m/-	m/+	m/+	m/+
IIDC	m/+	-	m/+	m/+	m/+
OIDC	m/+	m/-	m/+	m/+	m/+
AIDC	m/+	-	m/+	m/+	m/+
CCG	M/-	M/+	M/-	M/-	M/-
CCTG	M/-	M/+	M/-	M/-	M/-

Table 2: The influence of metrics values on quality attributes

A threshold is a limit (high or low) placed on a specific metric. All the above metric values scale between 0 and 1, except the CCTG and CCG. We set the value of the threshold at 0.5.

An example to illustrate the above metrics and our approach based on metrics for the best solution selection is presented. In this example, nine components have been found as candidates. We add two more components to complete the final system: a *Read* (R) and a $Write$ (W) component. The used algorithms [FMD06, VM06] provided

several solutions. We only present two of them and discuss the different metrics values for each system-solution and their influences on the quality attributes.

The first solution contains only six of them (without taking into consideration the R and W components). It has the values for the metrics around the medium value, for all quality attributes. For example, the majority of the components have a very high functionality in the system (PSU, CPSU, OIDC and IDC values are very close to 1) and at the same time they can offer new functionalities for the future improvement by adding new provided services (which influences the maintainability attribute). Regarding the coupling metrics we can remark that there is a maximum limit that is not very high and we can say that the maintainability and reusability are not strongly influenced. The assembly values metrics suggest that the solution is not considered to be the "best" for every quality attribute, but a medium "best" solution for the overall system. The value of the $AIDC$ metric is close to 1 but we must take also into consideration the $CCTG$ metric to decide which solution best represents our future needs (if we would like to improve and add new functionality or if we just want to have a good functionality for the system).

The second solution contains only three internal components form the set of candidate-components. The metrics values that influence the functionality attribute are close to 1 revealing a good functionality of each component inside the system, but the other metrics values influence negatively the other quality attributes. The 0.50 chosen threshold is exceeded for all the computed metrics. In Table 2 we can see that a high value influences negatively almost all the quality attributes discussed. The values of $CCTG$ metric are relatively high considering that there are few components in the solution. The $CCTG$ value for the ninth component is considered to be high yielding a very hard understandability and testability.

6.4 Conclusions

The current chapter approaches the problem of metrics based components assembly evaluation. Metrics are selected and new metrics are proposed in order to quantify these aspects of interactions between components, studying the influence of metrics values on quality attributes. The proposed assessment provides decision support in the selection of optimum solution from a set of configurations which represents possible solutions. The chapter is based on the published papers [SV07b, SV07a].

7 Conclusions and Future Research Directions

The main goal of this thesis, as stated at the beginning of this work, is to investigate how metrics can be applied in the assessment of software systems, in particular in the assessment of Object-Oriented Design and of Component-Based Systems. We also aim to investigate methods and techniques in order to offer a relevant interpretation of the obtained measurement results.

In this context, our work introduced *A quantitative evaluation methodology for object-oriented design*. This methodology, based on static analysis of the source code, is described by a conceptual framework which has four layers of abstraction: *i*) *Object-Oriented Design Meta-Model* – formally defines the domain of the assessment, specifying the elements that are evaluated, their properties and the relationships between them; *ii*) *Formal Definitions of OOD Metrics* – consists of a library of OOD metrics definitions; *iii*) *Specifications of the Assessment Objectives* – specifies in a formal manner the assessment objectives using a metrics based approach; *iv*) *Measurement Results Analysis* – uses the fuzzy analysis method in order to interpret the measurement results obtained in the assessment process.

Another direction that was investigated is in the field of Component Based Systems. The selection of a component (within a set of possible candidates which offer similar functionalities) requires the evaluation of the candidate components using objective methods. The evaluation results help developers in the selection task. The thesis proposes a formal approach concerning component based systems assessment. More precisely, we have defined a general, scalable and integrated model for a quantitative evaluation regarding both individual components and the system obtained by connecting components. The proposed model provides a standard terminology in order to formally define metrics. Fuzzy clustering analysis is used as a robust method for the interpretation of the obtained measurements results.

New software metrics are defined and introduced to predict the quality attributes for the whole system and to best select a solution from a set of found configurations. We have proposed some quality attributes (reusability, functionality, understandability, maintainability, testability) to consider when to analyze the quality of an assembly. New software metrics (CCG - Component Coupling Grade, CCTG - Component Coupling Total Grade) are used to select, among all obtained configurations, the solution that best represents the system requirements.

For each original approach we have suggested possible developments and new research directions in the assessment of software systems.

Bibliography

[Abr93] F.B. Abreu. Metrics for Object Oriented Environment. In *Proceedings of the 3rd International Conference on Software Quality, Tahoe, Nevada, EUA, October 4th - 6th*, pages 67–75, 1993.

[Abr95] F.B. Abreu. The MOOD Metrics Set. In *9th European Conference on Object-Oriented Programming (ECOOP'95) Workshop Metrics*, 1995.

[AC99] C. Alves and J. Castro. Pore: Procurement-oriented requirements engineering method for the component based systems engineering development paradigm. *Int. Conf. Software Eng. CBSE Workshop*, 1999.

[AC01] C. Alves and J. Castro. Cre: A systematic method for cots component selection. *In Brazilian Symposium on Software Engineering, Rio De Janeiro*, 2001.

[AR94] F.B. Abreu and Carapuca Rogerio. Candidate Metrics for Object- Oriented Software within a Taxonomy Framework. In *Journal of systems software 26*, pages 359–368, 1994.

[BA04] M. A. S. Boxall and S. Araban. Interface Metrics for Reusability Analysis of Components. In *Proceeding of Australian Software Engineering Conference ASWEC'2004*, pages 40 – 51, 2004.

[Bar02] A.L. Baroni. *Formal Definition of Object-Oriented Design Metrics*. PhD thesis, Ecole des Mines de Nantes, 2002.

[BBA02] A. Baroni, S. Braz, and F. Abreu. Using OCL to Formalize Object-Oriented Design Metrics Definitions. In *ECOOP'02 Workshop on Quantitative Approaches in OO Software Engineering*, 2002.

[BBeA03a] A. Baroni, S. Braz, and F. Brito e Abreu. A formal library for aiding metrics extraction. *Proceedings of ECOOP Workshop on Object-Oriented Re-Engineering*, 2003.

[BBeA03b] A. Baroni, S. Braz, and F. Brito e Abreu. Using OCL to formalize object-oriented design metrics defnitions. *Proceedings of ECOOP Workshop on Quantative Approaches in Object-Oriented Software Engineering*, 2003.

[BBM96] V. Basili, L. Briand, and W. Melo. A validation of object-oriented design metrics as quality indicators. 20(10):751–761, 1996.

[BDE99] J. Bansiya, C. Davis, and L. Etzkorn. An entropy-based complexity measure for object-oriented designs. *Theory an Practice of Object Systems.*, 5(2):111–118, 1999.

[BDW99] L. Briand, J. Daly, and J. Wust. A Unified Framework for Coupling Measurement in Object-Oriented Systems. *IEEE Transactions on Software Engineering*, 25(1):91–121, 1999.

[Bez81] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.

[BHSS06] Paul Baker, Mark Harman, Kathleen Steinhofel, and Alexandros Skaliotis. Search Based Approaches to Component Selection and Prioritization for the Next Release Problem. In *ICSM '06: Proceedings of the 22nd IEEE International Conference on Software Maintenance*, pages 176–185, Washington, DC, USA, 2006. IEEE Computer Society.

[BK95] J.M. Bieman and B.K. Kang. Cohesion and Reuse in an Object-Oriented System. *ACM Symposium on Software Reusability*, 1995.

[BME07] G. Booch, R. A. Maksimchuk, and Michael W. Engel. *Object-Oriented Analysis and Design with Applications.* Addison-Wesley, 2007.

[Boo94] G. Booch. *Object-Oriented Analysis and Design with Applications.* Benjamin Cummings, Redwood City, 2 edition, 1994.

[BR88] V. Basili and D. Rombach. The TAME project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering*, 14(6), 1988.

[BTV06] M.F. Bertoa, J.M. Troya, and A. Vallecillo. Measuring the Usability of Software Components. Journal of Systems and Software. *IEEE Software*, 79(3):427–439, 2006.

[CK93] S. R. Chidamber and C. F. Kemerer. A Metrics suite for Object Oriented design. *IEEE Transactions on Software Engineering*, 20(6):476 – 493, 1993.

[CK94] S.R. Chidamber and C.F. Kemerer. A Metric Suite for Object- Oriented Design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[CL02] I. Crnkovic and M. Larsson. *Building Reliable Component-Based Software Systems.* Artech House publisher, 2002.

[Crn03] I. Crnkovic. Component-based Software Engineering - New Challenges in Software Development. In *Proceeding of the 25th International Conference on Information Technology Interfaces*, pages 9 – 18, 2003.

[CS01] Philip T Cox and Baoming Song. A Formal Model for Component-Based Software. In *Proceedings of the IEEE 2001 Symposia on Human Centric Computing Languages and Environments (HCC'01)*, pages 304–311, 2001.

[CS03] A. Chatzigeorgiou and G. Stephanides. Entropy as a Measure of Object-Oriented Design Quality. In *1st Balkan Conference on Informatics (BCI'2003), Thessaloniki, Greece, November 21-23*, 2003.

[CSSW04] I. Crnkovic, H. Schmidt, J. A. Stafford, and K. Wallnau. Message from the Chairs. In *Proceedings of The 6^{th} ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction*, pages 1 – 7, 2004.

[CXS04] Alexander Chatzigeorgiou, Spiros Xanthos, and George Stephanides. Evaluating Object-Oriented Designs with Link Analysis. In *Proceedings of the 26th International Conference on Software Engineering*, pages 23–28, 2004.

[CY91] P. Coad and E. Yourdon. *Object-Oriented Design*. Prentice Hall, London, 2 edition, 1991.

[DDN00] S. Demeyer, S. Ducasse, and O. Nierstrasz. Finding refactorings via change metrics. In *In Proceedings of OOPSLA 2000, ACM SIGPLAN Notices*, pages 166 – 178, 2000.

[DeM82] T. DeMarco. *Controlling Software Projects; Management, Measurement and Estimation*. Yourdan Press, New Jersey, 1982.

[Des] Software Design Pattern. `http://en.wikipedia.org/wiki/Software_design_pattern`.

[Dum88] Dan Dumitrescu. *Hierarchical pattern classification*. Fuzzy Sets and Systems 28, 1988.

[DW98] D. F. D'Souza and A. C. Wills. *Objects, Components, and Frameworks with UML: The Catalysis Approach*. Addison-Wesley, Reading, MA, 1998.

[EGH02] L. H. Etzkorn, S. Gholston, and W. E. Hughes. A semantic entropy metric. *Journal of Software Maintenance: Research and Practice.*, 14(4):293–310, 2002.

[EWEBBF] Mohamed El-Wakil, Ali El-Bastawisi, Mokhtar Boshr, and Ali Fahmy. Software Metrics - A Taxonomy. Technical report.

[FBB+99] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[FBR04] Michael Roy Fox, David C. Brogan, and Jr. Paul F. Reynolds. Approximating component selection. In *WSC '04: Proceedings of the 36th conference on Winter simulation*, pages 429–434. Winter Simulation Conference, 2004.

[Fen94] N. Fenton. Software Measurement: A Necessary Scientific Base. *IEEE Transactions on Softw. Engineering, 20(3)*, 1994.

[Fen95] N.E. Fenton. *Software Metrics: A Rigorous Approach.* International Thomson Computer Press, London, UK, 1995.

[FMD06] A. Fanea, S. Motogna, and L. Diosan. Automata-based Component Composition Analysis. *Studia Universitas Babes-Bolyai, Seria Informatica*, LI(1):13–20, 2006.

[FP97] N. Fenton and S.L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* International Thomson Computer Press, London, UK, second edition, 1997.

[FP02] M. Frentiu and H.F. Pop. A study of dependence of software attributes using data analisys techniques. *Studia Universitas Babes-Bolyai, Seria Informatica*, L(2):53–66, 2002.

[GA04a] Miguel Goulo and O Brito E Abreu. Formalizing Metrics for COTS. In *Proceddings of the ICSE Workshop on Models and Processes for the Evaluation of COTS Components*, pages 37–40, 2004.

[GA04b] Miguel Goulo and O Brito E Abreu. Software Components Evaluation: an Overview. In *In Proceedings of the 5th Conferncia da APSI*, 2004.

[GA05a] M. Goulao and F.B. Abreu. Formal Definition of Metrics upon the CORBA Component Model. In *First International Conference on the Quality of Software Architectures*, 2005.

[GA05b] M. A. Goulo and F. B. Abreu. Composition Assessment Metrics for CBSE. In *Proceedings of The 31st Euromicro Conference, Component-Based Software Engineering Track*, pages 96 – 103, 2005.

[GG03a] N. S. Gill and P. S. Grover. Component-based measurement: few useful guidelines. *SIGSOFT Softw. Eng. Notes*, 28(6):1–4, 2003.

[GG03b] N. S. Gill and P. S. Grover. Reusability Issues in Component-based Development. *SIGSOFT Softw. Eng. Notes*, 28(4):1–4, 2003.

[GG04] N. S. Gill and P. S. Grover. Few important considerations for deriving interface complexity metric for component-based systems. *SIGSOFT Softw. Eng. Notes*, 29(2):1–4, 2004.

[GG07] L. Gesellensetter and S. Glesner. Only the Best Can Make It: Optimal Component Selection . *Electronic Notes in Theoretical Computer Science*, 176(2):105–124, 2007.

[GHJV94] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[Hai] System.Reflection-based ILReader. `http://blogs.msdn.com/b/haibo_luo/archive/2006/11/06/system-reflection-based-ilreader.aspx`.

[HDM03] A. v. d. Hoek, E. Dincel, and N. Medvidovic. Using Service Utilization Metrics to Assess and Improve Product Line Architectures . In *In Proceedings of the 9th IEEE International Software Metrics Symposium Metrics*, pages 0 – 0, 2003.

[HK01] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[HS96] B. Henderson-Sellers. Object-Oriented Metrics-Measures of Complexity. *Prentice Hall, Sydney*, 1996.

[JD98] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[JF88] R.E. Johnson and B. Foote. Designing reusable classes. *Journal of Object-Oriented Programming*, 1(2):22–35, 1988.

[JMF99] A. Jain, M. N. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[Kel89] W. T. Kelvin. *Popular Lectures and Addresses*. 1889.

[Kon95] J. Kontio. OTSO: A Systematic Process for Reusable Software Component Selection. Technical report, Technical report, University of Maryland, 1995.

[KSW95] K. Kim, Y. Shin, and C. Wu. Complexity Measures for Object-Oriented Program Based on the Entropy. In *In Proceedings of the Second Asia Pacific Software Engineering Conference*, 1995.

[Lak96] J. Lakos. *Large-Scale C++ Software Design*. Addison-Wesley, 1996.

[LD02] Michele Lanza and Stphane Ducasse. Beyond Language Independent Object-Oriented Metrics: Model Independent Metrics. In F. Abreu, Mario Piattini, Geert Poels, and Houari A. Sahraoui, editors, *Proceedings of the 6th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, pages 77–84, 2002.

[LH89] K.J. Lieberherr and I.M. Holland. Assuring good style for object-oriented programs. *IEEE Software*, 6:38–48, 1989.

[LH93a] W. Li and S. Henry. Maintenance Metrics for the Object Oriented Paradigm. *IEEE Proc. First International Software Metrics Symp*, pages 52–60, 1993.

[LH93b] W. Li and S. Henry. Object-oriented metrics that predict maintainability. *Journal of Systems and Software*, 23(2):111–122, 1993.

[Lis87] Barbara Liskov. Keynote address - data abstraction and hierarchy. In *OOPSLA '87 Addendum to the proceedings on Object-oriented programming systems, languages and applications (Addendum), NY, USA*, 1987.

[LK94] M. Lorenz and J. Kidd. *Object-Oriented Software Metrics*. Prentice-Hall Object-Oriented Series, Englewood Cliffs, NY, 1994.

[LM06] M. Lanza and R. Marinescu. *Object-Oriented Metrics in Practice*. Springer Verlag, 2006.

[log] Open Source Project: log4net. `http://logging.apache.org/log4net`.

[Lor93] M. Lorenz. *Object-Oriented Software Development: A Practical Guide*. Prentice-Hall, NJ, 1993.

[LR06] M. Lippert and S. Roock. *Refactoring in Large Software Projects*. John Wiley & Sons, 2006.

[LTGP02] A. Lozano-Tello and A. Gomez-Perez. ABAREMO: how to choose the appropriate software component using the analytic hierarchy process. In *The 14th international conference on Software engineering and knowledge engineering*, pages 781–788, ACM , New York, 2002.

[Mar] Robert Martin. Design Principles and Patterns. `http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf`.

[Mar97] R. Marinescu. The Use of Software Metrics in the Design of Object-Oriented Systems. Technical report, Politehnica University Timisoara, 1997.

[Mar02] R. Marinescu. *Measurement and Quality in Object Oriented Design*. PhD thesis, Faculty of Automatics and Computer Science, University of Timisoara, 2002.

[Mar09] Cristina Marinescu. *Towards Understanding and Quality Assessment of Enterprise Software Systems*. PhD thesis, Faculty of Automatics and Computer Science, University of Timisoara, 2009.

[MBA04] A. Marcus, M. Boxall, and S. Araban. Interface Metrics for Reusability Analysis of Components. In *Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04)*, 2004.

[McC76] T.J. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering, 2(4)*, pages 308–320, 1976.

[Mey88] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, Englewood Cliffs, 1988.

[Mic] Microsoft Corporation. Definition of the term component. `http://www.msdn.microsoft.com/repository/OIM/resdkdefinitionofthetermcomponent.asp`.

[Mih03] Petru Florin Mihancea. Optimizarea detectiei automate a carentelor de proiectare n sistemele software orientate pe obiecte. Technical report, Facultatea de Automatica si Calculatoare, Universitatea Tehnica Timisoara, 2003.

[ML02] T. Mens and M. Lanza. A graph-based metamodel for object-oriented software metrics. *Electronic Notes in Theoretical Computer Science*, 72:57–68, 2002.

[MP06] Jacqueline McQuillan and James Power. Towards the re-usability of software metrics definitions at the meta level. In *Proceedings of the ECOOP'2006 Doctoral Symposium*, 2006.

[MP08] Jacqueline A. McQuillan and James F. Power. A Metamodel for the Measurement of Object-Oriented Systems: An Analysis using Alloy. In *IEEE International Conference on Software Testing Verification and Validation*, pages 288–297, 2008.

[MSL06] S. Mazeiar, Li. Shimin, and T. Ladan. A Metric-Based Heuristic Framework to Detect Object-Oriented Design Flaws. In *Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC06)*, 2006.

[NH04] V. L. Narasimhan and B. Hendradjaya. A New Suite of Metrics for the Integration of Software Components. In *The First International Workshop on Object Systems and Software Architectures WOSSA 2004*, 2004.

[OC08] Mark O'Keeffe and Mel Cinnide. Search-based refactoring: an empirical study. *Journal of Software Maintenance and Evolution: Research and Practice*, 20(5):345–364, 2008.

[OCBZ09] Steffen Olbrich, Daniela S. Cruzes, Victor Basili, and Nico Zazworka. The Evolution and Impact of Code Smells: A Case Study of Two Open Source Systems. In *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 2009), Lake Buena Vista, Orlando, Florida*, 2009.

[P.F05] P.F. Mihancea and R. Marinescu. Towards the optimization of automatic detection of design flaws in object-oriented software systems. In *In Proc. of the 9th European Conf. on Software Maintenance and Reengineering*, pages 92–101, 2005.

[Pfl98] S.L. Pfleeger. *Software Engineering – Theory and Practice*. Prentice Hall, 1998.

[Rei01] R. Reiing. Towards a model for object-oriented design measurement. *Proceedings of ECOOP Workshop on Quantative Approaches in Object-Oriented Software Engineering*, 2001.

[RH00] L. Rosenberg and L. Hyatt. Software Quality Metrics for Object-Oriented System Environments. Technical report, NASA Goddard Space Flight Center, Greenbelt, Maryland, 2000.

[Rie96] A.J. Riel. *Object-Oriented Design Heuristics*. Addison-Wesley, 1996.

[RL92] C. Rajaraman and M.R. Lyu. Some Coupling Measures for C++ Programs. *Proceedings of TOOLS USA-92, Prentice-Hall, Englewood Cliffs, NJ*, 1992.

[Ros98] Linda H. Rosenberg. Applying and Interpreting Object Oriented Metrics. In *In Software Technology Conference (April 1998)*, 1998.

[SC06] C. Serban and C. Cretu. Impact on Design Quality of Refactorings on Code via Metrics. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 39–44, 2006.

[Sch03] Jean-Guy Schneider. *Component Scripts and Glue: A Conceptual framework for software composition*. PhD thesis, Institute fr Informatik (IAM), Universitt Bern, Berne, Switzerland, 2003.

[Ser06] C. Serban. Coupling measurement for compiled .Net code. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 21–26, 2006.

[Ser09a] C. Serban. A Formal Approach for OOD Metrics Definition. *First International Conference on Modelling and Development of Intelligent Systems, Sibiu, Romania*, pages 262–269, 2009.

[Ser09b] C. Serban. High Coupling Detection Using Fuzzy Clustering Analysis. *Knowledge Engineering: Principles and Techniques (Post-proceedings of KEPT 2009), International Conference, Babes-Bolyai University, Presa Universitara Clujeana*, pages 258 – 264, 2009.

[Ser09c] C. Serban. High Coupling Detection Using Fuzzy Clustering Analysis. *Special Issue of Studia Universitatis Babes-Bolyai Informatica: Proceeding of The International Conference on Knowledge Engineering: Principles and Techniques*, pages 223 – 226, 2009.

[Ser10] C. Serban. A conceptual framework for object-oriented design assessment. In *UKSim 4th European Modelling Symposium on Mathematical Modelling and Computer Simulation, Pisa, 17 - 19 November*, pages 90–95, 2010.

[Ser11] C. Serban. God Class Design Flaw Detection In Object Oriented Design. A Case–Study. *Studia Universitas Babes-Bolyai, Seria Informatica*, LVI(4):33–38, 2011.

[Sha09] Arun Sharma. *Design and Analysis of Metrics for Component-Based Software Systems*. PhD thesis, School of Mathematics and Computer Applications, Thapar University, India, 2009.

[SKG07] Arun Sharma, Rajesh Kumar, and P. S. Grover. A Critical Survey of Reusability Aspects for Component-Based Systems. *World Academy of Science, Engineering and Technology*, 2007.

[SM05] C. Serban and A. Mihis. Software quality assurance. In *Proceedings of the Symposium Zilele Academice Clujene*, pages 207–212, 2005.

[SP08] C. Serban and H.F. Pop. Software Quality Assessment Using a Fuzzy Clustering Approach. *Studia Universitas Babes-Bolyai, Seria Informatica*, LIII(2):27–38, 2008.

[Spa00] M. Sparling. Lessons Learned Through Six Years of Component–Based Development. 43(10):47–53, 2000.

[SRM+10] Amjan Shaik, R. K. Reddy, B. Manda, C. Prakashini, and K. Deepthi. An Empirical Validation of Object Oriented Design Metrics in Object Oriented Systems. *Journal of Emerging Trends in Engineering and Applied Sciences*, 2(1):216–224, 2010.

[SV07a] C. Serban and A. Vescan. Metrics-based selection of a component assembly. *Special Issue of Studia Universitatis Babes-Bolyai Informatica: Proceeding of The International Conference on Knowledge Engineering: Principles and Techniques*, pages 324 – 331, 2007.

[SV07b] C. Serban and A. Vescan. Metrics for Component-Based System Development. *Creative Mathematics and Informatics*, pages 143 – 150, 2007.

[SVP08] C. Serban, A. Vescan, and H.F. Pop. Component Selection based on Fuzzy Clustering Analysis. *Creative Mathematics and Informatics*, 17(3):505 – 510, 2008.

[SVP09] C. Serban, A. Vescan, and H.F. Pop. A new component selection algorithm based on metrics and fuzzy clustering analysis. *In Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems*, pages 621–628, 2009.

[SVP10a] C. Serban, A. Vescan, and H. F. Pop. A conceptual framework for component-based system metrics definition. In *9th RoEduNet International Conference, June 2010, Sibiu, Romania*, pages 73–78, 2010.

[SVP10b] C. Serban, A. Vescan, and H. F. Pop. A Formal Model for Component-Based System Assessment. In *Second International Conference on Computational Intelligence, Modelling and Simulation, Bali, 28 - 30 September 2010*, pages 261–266, 2010.

[SW49] C.E. Shannon and W Weaver. *The Mathematical Theory of Communication*. Urbana, IL, University:University of Illinois Press, 1949.

[Szy98] C. Szyperski. *Component Software, Beyond Object-Oriented Programming*. ACM Press, Addison-Wesley, 1998.

[TS92] D.P. Tegarden and S.D. Sheetz. Object-oriented system complexity: an integrated model of structure and perceptions. *In OOPSLA92 Workshop on Metrics for Object-Oriented Software Development (Washington DC)*, 1992.

[TSM92] D. Tegarden, S. Sheetz, and D. Monarchi. Effectiveness of Traditional Software Metrics for Object-Oriented Systems. In *25th Hawaii International Confernce on System Sciences*, pages 359–368, 1992.

[Ves08a] A. Vescan. An evolutionary multiobjective approach for the Component Selection Problem. In *Proc. of the First IEEE International Conference on the Applications of Digital Information and Web Technologies, 4 - 6 August, Ostrava, Czech Republic*, pages 252–257, 2008.

[Ves08b] A. Vescan. *Construction Approaches for Component-based Systems*. PhD thesis, Department of Computer Science, Babes-Bolyai University, 2008.

[Ves09] A. Vescan. A Metrics-based Evolutionary Approach for the Component Selection Problem. *in Proceedings of the 11th International Conference on Computer Modelling and Simulation*, 2009.

[VM06] A. Vescan and S. Motogna. Syntactic automata-based component composition. In *The 32nd EUROMICRO Software Engineering and Advanced Applications (SEAA), Proceeding of the Work in Progress session*, pages 13–14, 2006.

[VP08] A. Vescan and H. F. Pop. The Component Selection Problem as a Constraint Optimization Problem. *in Software Engineering Techniques in Progress*, pages 203–211, 2008.

[WBD98] J. Wust, L. Briand, and J. Daly. A Unified Framework for Cohesion Measurement in Object-Oriented Systems, journal = Empirical Software Engineering: An International Journal. 3(2):65–117, 1998.

[Wet04] Richard Wettel. Automated Detection of Code Duplication Clusters. Technical report, Faculty of Automatics and Computer Science, Politehnica University of Timisoara, 2004.

[WM96] Arthur H. Watson and Thomas J. McCabe. Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric. In *National Institute of Standards and Technology NIST Special Publication*, pages 500–235, 1996.

[WYF03] H. Washizaki, H. Yamamoto, and Y. Fukazawa. A Metrics Suite for Measuring Reusability of Software Components. In *In Proceedings of 9th IEEE International Software Metrics Symposium METRICS 2003*, pages 211 – 223, 2003.

[XHC+00] T. Xie, H. Huang, X. Chen, H. Mei, and F. Yang. *Object Oriented Software Quality Evaluation Technology*. Department of Computer Science and Technology, Peking University, 2000.

[Zad65] L.A. Zadeh. *Fuzzy sets*. Information and Control 8, 1965.