Analysis of the Designs of Coupling Measures: A Case Study

Laila Cheikhi
École de Technologie Supérieure
laila.cheikhi.1@ens.etsmtl.ca

Abran Alain
École de Technologie Supérieure
alain.abran@.etsmtl.ca

Miguel Lopez
Cetic, Belgium,
mlo@Cetic.bc

Abstract

Various measures have been proposed in software engineering for evaluating the quality of object-oriented software systems, many of them aimed at measuring the structural properties of the design of the software, such as coupling, cohesion and inheritance. There is diversity among the current proposals for coupling measurements and models, reflecting a lack of consensus on coupling and the need for a reference framework. This paper investigates the design of coupling measures based on Abran and Jacquet's model of the process for designing a measurement method. This analysis is illustrated with a case study using measures of one type of coupling, suggested by Chidamber and Kemerer: Coupling Between Objects (CBO). This case study verifies whether or not this CBO measure includes all the design elements of a measurement method.

1. Introduction

Quantitative measures constitute a significant tool for evaluating the quality of software by making it possible, when these measures are well defined, to evaluate quality a priori (measurement of internal attributes) or a posteriori (measurement of external attributes), from various perspectives (developer, end-user, manager, etc).

With the introduction of object-oriented technology in software engineering, several measures have been proposed by researchers and practitioners to take into account the new characteristics of this technology [CHID94], CHID91, BRIA97, LI93, ABRE95]. Most of the object-oriented (OO) measures have been designed by intuition or in an empirical way, often based on the ease of counting certain entities of OO systems [ABRA02]. These OO measures have often been used with the aim of evaluating, in particular, the quality of the design and of the source code, via the evaluation of design quality criteria such as coupling, cohesion and inheritance. In this article, the focus is on the coupling criteria.

The Design KA of the Software Engineering Body of Knowledge – SWEBOK [ABRA04/1] [ISO 05]) – indicates that "*most measures that have been proposed* [for the design phase] *generally depend on the approach used for producing the design. These measures are classified in two broad categories: function-oriented (structured) design measures* and *object-oriented design measures* [ABRA04b].

These design measures are proposed in order to evaluate, or to try to predict, the internal quality of the software product (visible to the developers) and its external quality (visible to the users). Coupling is one of the important internal attributes of software design which is studied in structured programming, and one which takes on a great deal of importance in the OO paradigm: in the literature, it is suggested that a high quality software design must obey the criterion of low coupling [BRIA97].

Chidamber and Kemerer [CHID91, CHID94] proposed a suite of metrics for OO design, including measures such as: Coupling Between Objects (CBO) and Response for a Class (RFC). Li and Henry [LI93] proposed coupling measures such as Message Passing Coupling (MPC) and Data Abstraction Coupling (DAC). Abreu *et al.* [ABRE95] suggested measuring coupling for the whole system with the Coupling Factor (COF) measure. Briand *et al.* [BRIA97] proposed a series of 18 coupling measures for class coupling, taking into account the various mechanisms offered by the C ++ language.

This diversity in measures corresponds to the variety of objectives of these measures and corresponding designs. Sometimes, the same measure can be applied differently by its users, thereby leading to a diversity of results and of ways to interpret these results. A potential cause of such a difficulty could be ambiguity in the definition of the concept to be measured in terms of the objectives of the measure, the characteristics of the measured attribute and the terminology used to describe the measure.

In this paper, the objective is to study the coupling measure design itself. A process model for the design of a measurement method has been proposed in [JACQ97, ABRA99] and is used in this paper to study whether or not the designs of coupling measures include the various design elements of a measurement method. One of the methods

most often used, and most often quoted by researchers when analyzing software coupling, Chidamber and Kemerer's CBO, has been selected as a case study.

This paper is structured as follows. The terminology selected is presented in section 2. Related work on the design of measurement methods is presented in section 3, and on coupling measures in section 4. Section 5 then analyzes the design of Chidamber and Kemerer's coupling measure (CBO). Section 6 presents a discussion on the problems identified in the variety of coupling measures proposed. Section 7 provides a summary and identifies some suggested next steps.

2. Terminology

In software engineering, and particularly in software measurement, terms with different names, and even different meanings, are sometimes used with the same objective; for example, "measure" and "metrics". For the sake of clarity and consistency, it is important to provide precise definitions. This section presents the terminology adopted for this analysis, for both measurement and OO concepts.

2.1. Measurement-related terms

Some of the terms have been adopted from IEEE 610 [IEEE 90], ISO 9126-1 [ISO 01] and the ISO International Vocabulary of Basic and General Terms in Metrology (VIM-1993 [ISO 93]) - this ISO document represents the international consensus on measurement terminology.

Attribute: a measurable physical or abstract property of an entity [ISO 01].
Entity: in computer programming, any item that can be named or denoted in a program; for example, a data item, program statement or subprogram [IEEE 90].
Design: the process of defining the architecture, components, interfaces and other characteristics of a system or component, or the result of (that) process [IEEE 90].
Design entity: an element (component) of a design that is structurally and functionally distinct from other elements and that is separately named and referenced [IEEE 90].
Internal attribute: an attribute that can be measured purely in terms of the product itself by the analysis of the static properties of the software [ISO 01].
External attribute: an attribute that can be measured only when the product is executed in a specific environment [ISO 01].
Measurement: a set of operations having the object of determining the value of a quantity [ISO 93].
Measurement method: a logical sequence of operations, described generically, used in the performance of measurement [ISO 93].
Scale: a set of values with defined properties [ISO 01]. Examples of types of scale are: a nominal scale, which corresponds to a set of categories; an ordinal scale, which corresponds to an ordered set of scale points; an interval scale, which corresponds to an ordered scale with equidistant scale points; and a ratio scale, which not only has equidistant scale points, but also possesses an absolute zero. Measuring using nominal or ordinal scales produces qualitative data, and measuring using interval and ratio scales produces quantitative data.

2.2. OO terms

The terminology adopted for the set of OO concepts is the following one:

An *object* is characterized by a state (a set of properties characterizing the object as well as the value of each property) and a behavior (a set of operations of the object).
A *class* is the specification of the object; it is the basic model or prototype from which the objects are created. The classes are defined by the public services offered to the users by the intermediary of the methods and public attributes.
The *methods* implement the operations that can be carried out in a class (the logic of programming).
The attributes represent the properties of a class. They are defined by their names and their types: simple or complex (int i or Class_A i).
The *inheritance* corresponds to the creation of new classes (derived classes), but based on the existing classes (base classes) whether by reuse or by extension of the functionalities of the previous ones. A derived class inherits the structure and the behavior of the basic class and can be used as a basic class for a later derivation, giving rise to a hierarchy of classes called an *inheritance tree*.
The *dependence* between classes corresponds to the relation between the classes. A class (client) can use the functionalities (attributes or methods) offered by another class (server) via the exchange of messages.

3. Measurement process model

3.1. High-level model

The process for designing a measurement method has been discussed in Abran and Jacquet [JACQ97, ABRA99] and is illustrated in Figure 1. Such a model can also be used when verifying the design of existing measures proposed to the industry.

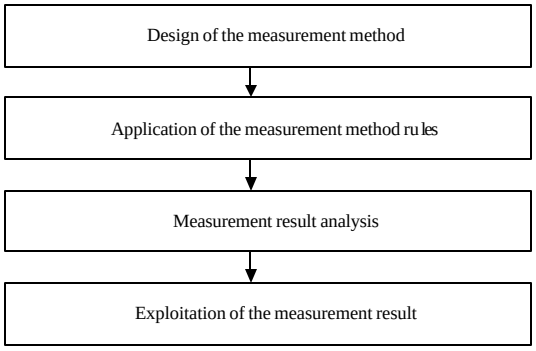


Figure 1: High-level model of a measurement process [JACQ97, ABRA99]

This model identifies four major steps, starting with the design itself of the measurement method and ending with the exploitation of the measurement results:

- Design of the measurement method: this step includes the definition of the objectives, the identification of the attribute to be measured and the definition of the numerical assignment rules.
- Application of the measurement method rules: this activity corresponds to the application of the measurement method to the software or to a part of the software. In this step, the necessary documentation for the application of the measurement method is collected, the software is modeled and the numerical rules are applied.
- Measurement results analysis: this activity corresponds to the study of the results obtained following the application of the measurement method. These results must be documented on the one hand, and, on the other hand, be analyzed by different methods in order to ensure their integrity.
- Exploitation of the measurement result: in this step, the measurement result is exploited in quality models, productivity models or estimation models.

3.2. Design of the measurement method

Since we are interested in the design of measurement methods, we discuss this next in more detail. This design process is composed of four substeps, as presented in Figure 2. The model is illustrated taking as an example a measurement method that has achieved international recognition at the ISO level as ISO 19761 [ISO 03], the COSMIC-FFP measure [ABRA03].

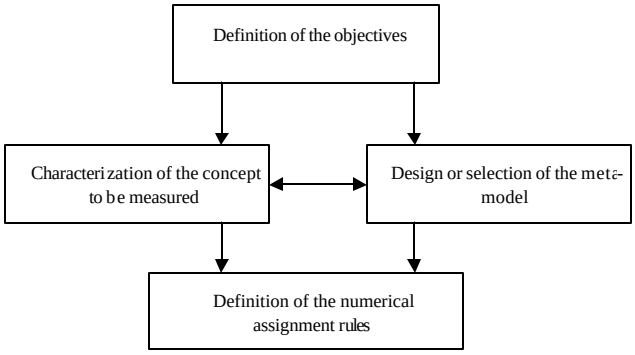


Figure 2: Design of the measurement method [JACQ97, ABRA99]

- Definition of the objectives: during this substep, the '*why*' of measurement must be well defined, and include the point of view and perspectives related to the use of the measurement results. For example, the objective of the COSMIC-FFP measure (ISO 19761 [ISO 93]) is to measure the functional size of the software.

- Characterization of the concept to be measured: during this substep, a clear and operational definition of the measured attribute is built. For example, the COSMIC-FFP measurement method is based on the concept of 'types of data movements'.

- Design or selection of a meta-model of the attribute to be measured: the software is different for each user, and how it is seen depends on the user's view point. The meta-model construction process consists in selecting a set of characteristics to represent the software and finding the rules to identify the entities and their relationships. For example: the concepts of boundary, data movements and data groups of the COSMIC-FFP measure and of their relationships, that is the COSMIC-FFP meta-model of a software.

The two substeps described above are, in practice, carried out in parallel because an attribute to be measured cannot be characterized if the software is not modeled, i.e. the meta-model constructed corresponding to the relationships across the characteristics identified for that attribute.

- Definition of the numerical assignment rules: this substep deals with the construction of the rules that specify the assignment of the numerical values to the software through the use of mathematical expressions. For COSMIC-FFP, this includes the selection of a single unit of measurement (the Cfsu, or Cosmic functional size unit) for any data movement type, then the addition of all data movements identified, using the same Cfsu unit.

Another study using this high-level model explored McCabe's Cyclomatic Complexity Number, based on the design of the method rather than on a variety of interpretations of this number [ABRA04a]. This approach made it possible to identify some weaknesses, such as with its numerical assignment rules, that do not take into account the admissible transformations of the different scale types.

4. Coupling Measures – Overview

For the analysis of the design of coupling measures, a review of the literature on the definitions of this criterion is necessary. In this section, we present some definitions of coupling in both the structured paradigm and in the OO paradigm.

Stevens *et al.* defined the coupling concept in the structured paradigm as "*the measure of the strength of association established by a connection from one module to another*" [STEV74]. The IEEE 610.12 standard defines coupling as "*the manner and degree of interdependence between software modules. Types include common-environment coupling, content coupling, control coupling, data coupling, hybrid coupling, and pathological coupling*" [IEEE 90]. These types of coupling are presented in Figure 3.

A model is defined as "(1) *A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from, an assembler, compiler, linkage editor, or executive routine; or (2) A logically separable part of a program*" [IEEE 90].

It should be noted that neither of the two definitions above clearly distinguishes between the concept to be measured and the measurement of this concept. The definition in [IEEE 90] associates the concept of coupling with the concept of interdependence, and specifies the characterization of the coupling concept by enumerating characteristics (common-environment coupling, content coupling, control coupling, data coupling, hybrid coupling and pathological coupling). However, it does not specify how these sub-concepts are connected to each other, e.g. its meta-model of how these sub-concepts are interrelated within the coupling concept.

Chidamber and Kemerer define coupling in the following way: "*An object is coupled to another object if one of them acts on the other, i.e., methods of one use methods or instance variables of another*" [CHID91, CHID94], and proposed two versions of the CBO coupling measure, whereas the first definition excludes coupling due to inheritance and the second definition includes it.

Li and Henry [LI93] view coupling through the passage of messages between the classes and through data abstraction. This latter is measured in terms of ADT (Abstract Data Type) variables. A variable can be an ADT variable if it represents the definition of another class.

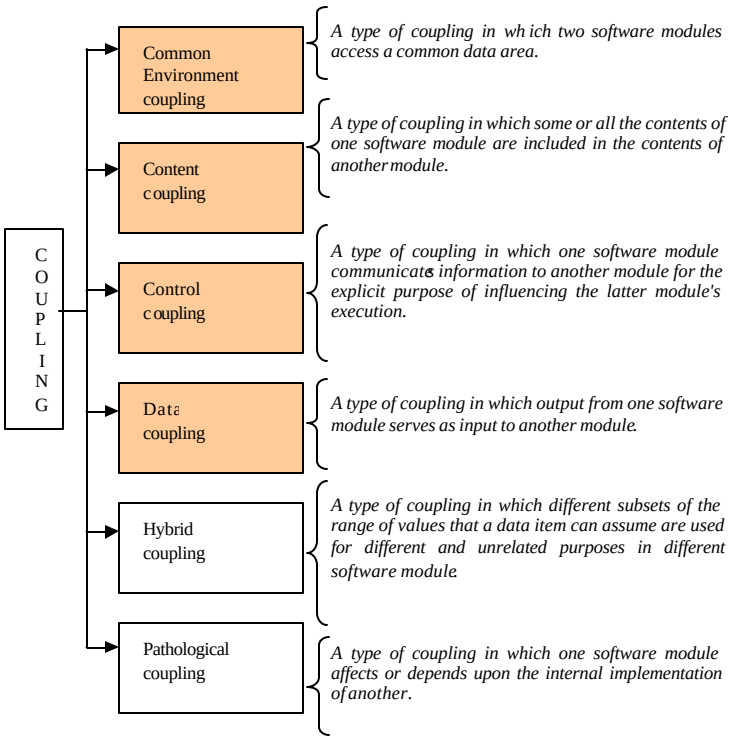


Figure 3: Types of coupling in IEEE 610.12

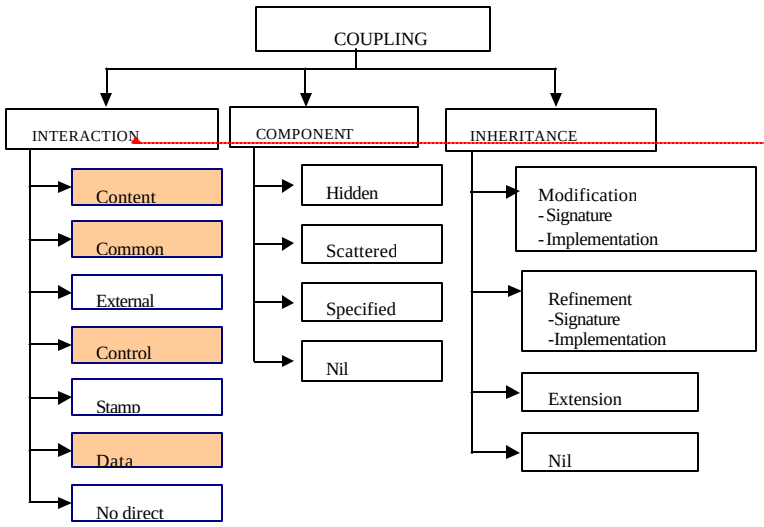
Using the Stevens *et al.* definition in [STEV74] as a basis, Eder *et al.* [EDER94] proposed a classification of coupling in the OO paradigm, and identified three types of relationships (interaction, component and inheritance) from which three kinds were identified and defined as follows:

- Interaction coupling: the methods are coupled by interaction if a method invokes the others and/or via sharing of data. It involves classes and methods.
- Component coupling: the object class C' can be a component of object class C if and only if C' appears in C. It concerns only object classes.
- Inheritance coupling: two classes are coupled by inheritance if one class is a direct or indirect subclass of the other. It concerns only object classes.

Each of these types of coupling category includes many other subtypes, as presented in Figure 4.

Hitz and Montazeri [HITZ95] distinguish between Class Level Coupling (CLC) and Object Level Coupling (OLC). The first represents *"the coupling resulting from state dependencies between classes during the development life cycle, where the state of a class refers to the class definition and the program code of its methods."* The second represents *"the coupling resulting from state dependencies between objects during run-time of a system."* Then, the authors identify a set of factors affecting the strength of coupling for each type.

Briand *et al.* [BRIA97] define coupling as *"the degree of interdependence among the components of a software system."* They have proposed a series of measures to quantify the level of class coupling between classes in OO software systems developed with the C++ programming language. These measures are intended to capture the various types of interaction between the classes and are based on the following three descriptions: (1) the type of relationship that can exist between classes (inheritance, friendship or other); (2) the type of interaction that can exist between classes or their elements (Class-Attribute, Class-Method or Method-Method); and (3) the locus of impact (import or export).



Formatted: Font: 7 pt

Figure 4: Taxonomy of coupling characteristics of OO systems in Eder *et al.* [EDER94])

Briand *et al.* [BRIA99] also proposed their own terminology and their own formalism for the expression of coupling measures. Their objective was to propose a unified "framework" to support the definition of coupling measures. With this intention, they identified ambiguities in the definitions of existing coupling measures, and then proposed a rewriting of the formulas of these measures according to their formalism. On the one hand, the authors identified descriptive criteria for coupling in their study of coupling measures, but, on the other, they did not provide a general framework under which all the coupling measures suggested by the researchers could be gathered. Only one set of criteria to support the definition of new coupling measures in the OO approach was provided: (1) type of connection; (2) locus of impact; (3) granularity of the measure; (4) stability of the server; (4) the direct or indirect coupling; and (5) inheritance.

An inventory of 16 definitions of coupling is presented in Appendix 1. From this non-exhaustive literature review about coupling measures, it can be observed that authors have defined the concept of coupling in their own ways. Each has proposed a different model, a different classification and different measures based on his point of view on coupling and with different mechanisms of identification of coupling between the classes (and their elements: attributes and methods) of an OO system.

On the one hand, there is consensus on the undesirable effects of a high degree of coupling on the structure and the complexity of the system [BRIA97, HITZ95, BRIA99]. On the other, the diversity of the proposed coupling measures shows a lack of consensus on the definition of coupling and of its measurement, making the use of such measures challenging for both prospective users and researchers alike: each definition suggested lead to a different design for a measure.

5. Case study: Analysis of the design of the CBO coupling measure

The CBO measure was proposed by Chidamber and Kemerer [CHID94] to measure the coupling between classes in an OO system. In this section, we analyze the design of this measure to verify whether or not it clearly refers to all the necessary design concepts identified in Abran and Jacquet [JACQ97,ABRA99].

5.1. Definition of the objective

The CBO [CHID94] is based on the following definition: "*An object is coupled to another if one of them acts on the other, i.e., methods of one use methods or instance variables of another. Since objects of the same class have the same properties, two classes are coupled when methods declared in one class use methods or instance variables defined by the other class.*"

Coupling is thus defined as a relation (an object acts on another object; a method of a class uses a method or a variable of another class). This 'acts on' relation is defined next by another type of relation 'a method uses another method or variable'. Moreover, for a class in an OO system, the CBO measure is defined in the following way: "*CBO for a class is a count of the number of other classes to which it is coupled.*"

It can thus be inferred that the objective of the measurement of the coupling of OO design systems is to quantify the interaction of one class with the other classes of the software system.

5.2. Meta-model and attribute

Software is not hardware, and so it is necessary to model it in order to understand it. Modeling software consists in figuring out the set of entity types that describe the software, and the rules that allow the identification of these entity types and their relationships [JACQ97, ABRA99]. Consider that an OO software system is defined as consisting of a set of interacting classes. A class is composed of methods and attributes, and the interaction between the classes is carried out by means of the exchange of information between the methods and the attributes of these classes, i.e. a class needs the functionalities (methods and attributes) offered by the other classes and vice versa. Since the CBO measure uses OO concepts such as class, methods and attributes, CBO is based on the interaction between the classes and their elements. A meta-model of this measure can be figured out in the following way:

- *Entity types*: correspond to class, method and attribute. In this case, to measure the coupling of a class, the entity type used in the numerical assignment rule is then the class of an OO system.
- *Class1*: the class which provides its services to the other classes is called the server class.
- *Class2*: the class which needs the functionalities and the services offered by the other classes is called the client class.
- *Use relation between Class1 and Class2*: the relationship between the two classes is taken into account, and in the two directions:
    - from Class1 toward Class2 , or
    - from Class2 toward Class1.
- *Types of interaction* :
    - Methods of class1 use the methods of class2, or
    - Methods of class1 use the instance variables of class2.

It is important to specify that this measure is applicable only for the statically typed languages. Indeed, within the OO paradigm, any type is a class. However, some languages do not declare the type of the variables. They are called dynamically typed languages because the type of the variables is defined in the execution. For these languages, the inference of the type (determination of the type) of the variables is uncertain, based only on the static analysis of the code (the measurement instruments of the coupling are generally based on a static analysis of the code, i.e. without execution).

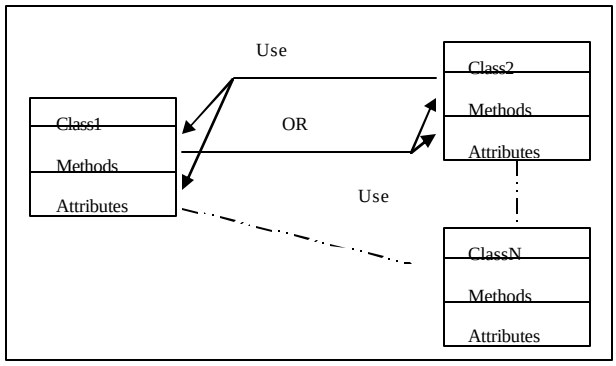


Figure 5: Meta-model of CBO.

Therefore, the meta-model presented here is valid only for statically typed languages, such as Java, Ada and C++. This meta-model (Figure 5), for the Chidamber and Kemerer CBO coupling measure, utilizes various types of entities. Thus, the attribute to be measured is a coupling with respect to the OO system class.

5.3. Numerical assignment rules

Even though the authors did not express the CBO measure in terms of numerical assignment rules, we are proposing the following corresponding ones, based on the definitions quoted in the previous section and on the meta-model identified.

$$CBO(C_i) = \sum_{j=1}^{NCS} [C_j \neq C_i \mid (m_i \in \{M_i\} \; use \; ((m_j \in \{M_j\}) \vee (a_j \in \{A_j\}))) \vee (m_j \in \{M_j\} use \; ((m_i \in \{M_i\}) \vee (a_i \in \{A\})))]$$

NCS: the number of classes in the system.

{CS}: the set of all classes in the system.

C_i : the class to be measured. $C_i \in \{CS\}$.

C_j : $C_j \in \{CS\}$, j varies from 1 to NCS.

$C_j \neq C_i$: this condition excludes the interaction within the class measured. The class for which the measure is calculated is not counted among the classes intervening in the coupling.

$\{M_i\}$: the set of all class C_i methods, and $\{A_i\}$ the set of all class C_i attributes.

$\{M_j\}$: the set of all class C_j methods, and $\{A_j\}$ the set of all class C_j attributes.

Example of the calculation of the CBO measure:

```
class A {
B b;
void mA () { b.mB();}
}

class B{
int c;
void mB() {c= mC();}
}

class C {
……...
void mC() {…}
void mC1(B.c) {…}
}
```

Figure 6: Examples of use relations

- CBO(A) = 1. Method mA() of class A uses method mB() of class B.
- CBO(B) = 2. Method mB() of class B uses method mC () of class C, and method mB() is used by method mA() of class A.
- CBO(C) = 1. Method mC() of class C is used by method mB() of class B. Instance variable c of class B is used as a parameter of method mC1() of class C.

In summary, if the coupling is defined as a relation, then the suggested CBO measure is the result of adding the classes whose methods use either the methods or the instance variables of the class for which we want to measure the coupling, or whose methods or instance variables are used by the measured class. The addition of these entities (classes) with this property (above) is thus regarded as a substitute for the measurement of the entity relation 'coupling', which includes an 'action' of an entity on another entity.

One of the fundamental concepts of measurement is the unit of measurement. For a class in OO design, the purpose of the CBO measure is to count the number of classes to which the measured class is coupled. From the definition of

this measure, it appears clearly that it corresponds to the class, since one counts the number of classes to which a class is coupled. By contrast, the numerical assignment rule of this measure is valid from the numerical point of view; it is the addition of only one type of unit, which is the class.

Therefore, the Chidamber and Kemerer CBO measure makes it possible to measure the coupling in terms of the design of the OO design, in particular in terms of the class. However, the authors did not provide any direct information on the attribute to be measured or on the meta-model in their definition.

6. Discussion on coupling

The section providing the overview of previous work on coupling gave a diversified set of definitions of coupling measures and of coupling classifications (in IEEE 610.12 in the context of the structured and OO paradigms; see also the Appendix). This illustrates that there is not yet a consensus on the concept of coupling, either on its definition or on the criteria for determining it. In other words, there is no reference framework on coupling (meta-model and attributes). Consequently, it is important to raise the following questions: Why did the researchers not arrive at a consensus? and, What would be the significance of such a consensus?

In fact, the formation of a base of knowledge of practices, rules and coupling measurements, and obtaining a consensus on this important criterion of the design of a system is an essential step. As is the case in any mature field, it is necessary to standardize the knowledge base to provide researchers with clear definitions, understandable practices and an adequate and unambiguous use of the rules at the time of the design of new coupling measurements.

Example of a coupling model:

Two classifications or models of coupling types are presented in section 4 as examples of a first attempt to build a high-level model for the coupling. The first corresponds to IEEE 610.12 [IEEE 90] and the second is proposed by Eder *et al.* [EDER94].

Comparison of these two models shows that they differ in the way in which coupling is studied. One of the reasons for this difference is the various objectives of each model. Indeed, IEEE 610.12 examines coupling in a general way, while Eder *et al.* are interested specifically in the coupling of OO systems. The second reason is that modules are used by some [IEEE 90], while classes and methods are used by others [EDER94]. However, it is possible to map one of these models to the other, since in Eder *et al.* a module corresponds to a method.

By contrast, it can be observed in Figures 3 and 4 that there are common characteristics in software coupling: *content coupling, common coupling, control coupling and data coupling* (Figure 7).

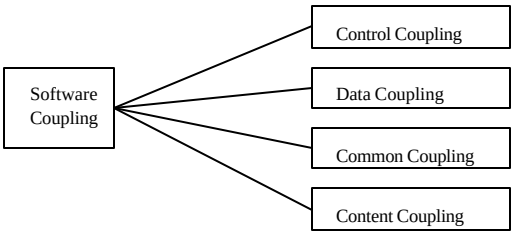


Figure 7: Examples of some common characteristics in models of coupling

This model of common characteristics can constitute a starting point for establishing a single framework for coupling measurements. However, for these two examples of coupling models, a suitable context and basic study of the various types of coupling is important in order to identify other coupling types which have characteristics in common.

To succeed in establishing a high-level model of coupling in a disciplined way, it is necessary:
(1) To agree on the definition of coupling;
(2) To conduct an in-depth study of all the coupling models suggested by researchers, independently of the approach used, in the software engineering field. For this, it is necessary to study each model, its categories, its subcategories

and their definitions and try to find the points of convergence between the various models in order to facilitate the development of the high-level model;
(3) To establish the high-level model that will form the general framework for coupling;
(4) Once this has been accomplished and consensus has been achieved among the various researchers on the set of qualitative and quantitative criteria, the fourth step can begin. This step involves studying the designs of the coupling measurements that exist in the literature and integrate each one into its corresponding category in the model;
(5) Finally, to use this framework as a basis for the definition of new measures by researchers and for the choice of measures by users.

7. Summary and next steps

In the past, the majority of coupling measures proposed in the literature by researchers were conceived from the point of view of the user when ameasurement method was designed, without being based on a reference framework. The definitions of some of these measurements contain ambiguities, as is the case with the McCabe Cyclomatic Complexity Number [ABRA04a]. Others are not well defined; consequently, the user can interpret the same measurement in several ways. Abran and Jacquet studied a number of works which treat the validation of metrics. They found various approaches to this subject. To show this diversity, they identified these approaches according to their criteria and techniques of validation, and they checked which steps of the measurement process (1 to 4) they satisfy [JACQ97, ABRA99].

In this article, the measurement process model, in particular the four substeps for the design of a measurement method, was introduced. A review of the literature on the proposed coupling measurements and models or classifications of coupling were also carried out. It was noted that there is a diversity of coupling measures and classification methods, but no framework exists to consolidatethe suggested coupling measures.

A case study was conducted with an aim of studying the design of the CBO, the coupling measure suggested by Chidamber and Kemerer [CHID94]. The design process of Abran and Jacquet's measurement method [JACQ97, ABRA99] was used as an analysis tool. From this analysis, it was noted that the CBO measure does not cover the design of the measurement method directly. Then, the importance of a standardized framework for the coupling based on coupling definitions and classifications in [IEEE 90] and [EDER 94] was discussed. With the knowledge that coupling is treated differently by different authors (modules/classes and methods), some common points were noted.

The definition of such a framework remains important in order to facilitate the definition of new measures, in particular in order to eliminate the lack of understanding and the inconsistencies that exist in the designs of measures.

To achieve this, it is necessary:

- To study in detail the existing coupling models and classifications, in order to establish a high-level model that will consolidateall the characteristics of coupling;
- To gain consensus on this coupling model;
- To integratethe coupling measures into this high-level model while validating their designs, in order to arrive at standardized measurements.

A framework or a standardized structure would not only be of value with respect to the concept of coupling, but would also be valid for the other structural design properties, such as cohesion, for example.

Finally, we suggest that the design process used for the Abran and Jacquet measurement method be used to analyze other measures not only for the attribute of coupling, but for any software measure. The advantage of such an analysis will be to point up some weaknesses of the methods used up to now in the design of these measures, to identify their deficiencies and based on insights gained, propose improvements.

References

[ABRA04a] Abran, A.; Lopez, M.; Habra, N. , *An Analysis of the Mc Cabe Cyclomatic Complexity Number*, 14th International Workshop on Software Measurement - IWSM 2004, Magdeburg, Germany, Shaker-Verlag, 2004, pp. 391-405.

[ABRA04b] Abran A., Moore J.W., Bourque P., Dupuis R. & Tripp L., *Guide to the Software Engineering Body of Knowledge – 2004 Version*, IEEE Computer Society, Los Alamos, 2005, URL: http://www.swebok.org

[ABRA03] Abran, A. ; Desharnais, J.-M. Oligny, S., St-Pierre D. and Symons, C., *COSMIC Implementation Guide to ISO 19761 : COSMIC-FFP*, École de technologie supérieure, Université du Québec, Montréal, 2003.

[ABRA02] Abran, A.; Sellami, A., *Initial Modeling of the Measurement Concepts in the ISO Vocabulary of Terms in Metrology*, 12th International Workshop on Software Measurement - IWSM 2002 , Magdeburg, Germany, Shaker-Verlag , 2002 , pp. 12 .

[ABRA99] Abran, A., Jacquet, J.P., "*A Structured Analysis of the New ISO Standard on Functional Size Measurement-Definition of Concepts*", Fourth IEEE International Symposium and Forum on Software Engineering Standards, 1999, pp. 230-241.

[ABRE95] Abreu, F.B.; Goulao, M.; and Esteves, R., *Toward Design Quality Evaluation of Object-Oriented Software Systems*, 5th International Conference on Software Quality, Austin, Texas, USA, 1995.

[BRIA99] Briand, L.C.; Daly, J.; Wuest, J., *A Unified Framework for Coupling Measurement in Object-Oriented Systems*. IEEE Transactions on Software Engineering, 1999.

[BRIA97] Briand, L. C.; Devanbu, P.; Melo, W., *An Investigation into Coupling Measures for C++*. IEEE, International Conference on Software Engineering 1997.

[CHID94] Chidamber, S.; Kemerer, C., *A Metrics Suite for Object- Oriented Design*. IEEE Transaction on Software Engineering, Vol. 20, No. 6, pp. 476-493, June 1994.

[CHID91] Chidamber, S.; and C. Kemerer, K., *Towards a Metrics Suite for Object-Oriented Design*. In Proceedings Object-Oriented Programming, Systems, Languages, and Applications, Phoenix, AZ, pp. 197-211, November 1991.

[EDER 94] Eder, J.; Kappel, G. and Schrefl, M., *Coupling and Cohesion in Object- Oriented Systems*, Technical Report, Univ. of Klagenfurt, 1994.

[HITZ95] Hitz, M. and Montazeri, B. *Measuring Coupling and Cohesion in Object-Oriented Systems* Int'l Symp. Applied Corporate Computing, Monterrey, Mexico, Oct. 1995.

[IEEE 90] IEEE, IEEE Std. 610.12, *Standard Glossary of Software Engineering Terminology*, IEEE-Computer Society, Los Alamitos, 1990.

[ISO 05] ISO/IEC, *ISO/IEC TR 19759:2005 Software Engineering -- Guide to the Software Engineering Body of Knowledge – SWEBOK, International Organization for Standardization - ISO, Geneva, 2005.*

[ISO 03] ISO/IEC, *ISO/IEC 19761:2003 Software Engineering - COSMIC-FFP - A functional size measurement method*, International Organization for Standardization - ISO, Geneva, 2003.

[ISO 01] ISO/IEC, *ISO/IEC 9126-1: Software Engineering – Product quality – Part 1: Quality Model*, International Organization for Standardization - ISO, Geneva, 2001.

[ISO 98] ISO/IEC, *ISO 14143-1:1998 — Information technology - Software measurement - Functional size measurement -- Part 1: Definition of concepts*, Geneva, International Organization for Standardization - ISO, Geneva, 1998.

[ISO 93] ISO, *International Vocabulary of Basic and General Terms in Metrology*, Second edition, International Organization for Standardization – ISO, Geneva, 1993.

[JACQ97] Jacquet, J.-P.; Abran, A., *From Software Metrics to Software Measurement Methods: A Process Model*, Third International Symposium and Forum on Software Engineering Standards (ISESS'97), Walnut Creek, CA, IEEE Computer Society, 1997, pp. 128-135 .

[LI 93] Li, W.; Henry S., *Maintenance Metrics for the Object-Oriented Paradigm* . First International Software Metrics Symposium, 1993, pp. 52–60. Baltimore, Maryland.

[STEV74] Stevens, W. P.; Myers, G.; Constantine, L., *Structured Design*, IBM Systems Journal, Vol. 13, No. 2, 1974.

Appendix : Examples of coupling measures.

Measures	Definitions
Chidamber et Kemerer [CHID91, CHID94]	
CBO	(Coupling Between Object): CBO is related to the notion that an object is coupled to another if one of them acts on the other, i.e., methods of one use methods or instance variables of another. Since objects of the same class have the same properties, two classes are coupled when methods declared in one class use methods or instance variables defined by the other class. CBO for a class is a count of the number of other classes to which it is coupled.
CBO'	The same definition, except that CBO for a class is a count of the number of non-inheritance related couples with other classes.
RFC	(Response Set for Class): RFC = | RS | where RS is the response set for the class. The response set for the class can be expressed as: $RS = \{ M_i \} \cup_{alli} \{ R_i \}$ where M_i = all methods in the class and R_i = set of methods called by M_i .
Li et Henry [LI93]	
MPC	(Message Passing Coupling): the number of send statements defined in a class.
DAC	(Data Abstract Coupling): number of ADTs defined in a class. A variable declared within a class X may have a type of ADT which is another class definition. The number of attributes in a class that have a type another class.
Briand et al. [BRIA 97]	
IFCAIC ACAIC OCAIC FCAEC DCAEC OCAEC IFCMIC ACMIC OCMIC FCMEC DCMEC OCMEC OMMIC IFMMIC AMMIC OMMEC FMMEC DMMEC	These coupling measures are counts of interactions between classes. The measures distinguish the relationship between classes (friendship, inheritance, none), different types of interactions, and the locus of impact of the interaction. The acronyms for the measures indicates what interactions are counted: • The first two letters indicate the relationship (A: coupling to ancestor classes, D: Descendents, F: Friend classes, IF: Inverse Friends (classes that declare a given class c as their friend), O: Others, i.e., none of the other relationships). • The next two letters indicate the type of interaction: ➢ CA: There is a Class-Attribute interaction between classes c and d, if c has an attribute of type d. ➢ CM: There is a Class-Method interaction between classes c and d, if class c has a method with a parameter of type class d. ➢ MM: There is a Method-Method interaction between classes c and d, if c invokes a method of d, or if a method of class *d* is passed as parameter (function pointer) to a method of class *c*. • The last two letters indicate the locus of impact: ➢ IC: Import coupling, the measure counts for a class c all interactions where c is using another class. ➢ EC: Export coupling: count interactions where class d is the used class.
Briand et al. [BRIA99]	
DAC' *RFC*? ?	Derived from DAC, but it count the number of different classes that are used as types for attributes in a class. The same as RFC, except that the response set of a class consists of the set M of methods of the class, and the set of methods directly or indirectly invoked by methods in M.
Abreu et al. [ABRE95]	
COF	COupling Factor : the actual number of couplings not imputable to inheritance / the maximum possible number of non-inheritance couplings in a system.