WEB TOOL FOR OBJECT ORIENTED DESIGN METRICS

José R. Hilera, Luis Fernández-Sanz and Marina Cabello
Department of Computer Science, University of Alcalá, Alcalá de Henares, Spain

Keywords: UML, XMI, Software metrics.

Abstract: An open source web application to calculate metrics from UML class diagrams is presented. The system can process any class diagram encoded in XMI format. After processing the XMI document, a complete report can be obtained in two different formats, HTML and spreadsheet file. The application can be accessed freely in a website. Source code is available for downloading.

1 INTRODUCTION

Measurement is an essential element of any discipline which intends to reach the level of engineering. In the case of software engineering, various important software attributes have been determined as key factors for the evaluation of application and systems: size, complexity, the expected frequency of error occurrences or test coverage (Fenton and Pfleeger, 1997). Metrics are a useful resource to help developers and managers to control if software products meet desired quality characteristics.

There are a few CASE tools that calculate some of the most popular OO metrics, with the aim of evaluating the UML diagrams being created with the tool itself (in www.objectsbydesign.com there is a short list of them). There are also commercial applications that offer the possibility of calculating metrics from UML files, although they are not free neither open source, e.g. *SDMetrics* (www.sdmetrics.com). Other authors have developed software for computing metrics from diagrams based on the Web Application Extension (WAE) for UML (Ghosheh and Black, 2009).

This paper presents a Web application developed by the authors which enables automatic measure of OO design metrics from class diagram encoded as standard XMI format. The system generates a complete report with detailed values of the most common metrics. Section 2 presents the metrics implemented in the system. Section 3 introduces the XMI format used to represent UML models. Section 4 describes the main features of the application. Finally, some conclusions as well as future lines of action are presented.

2 OBJECT ORIENTED DESIGN METRICS

Although metrics for different software lifecycle phases have been proposed (analysis, design, testing, maintenance, etc.), design metrics have reached a higher level of maturity and validation due to the interest and effort they have attracted since early 1990s. Obviously, OO design is mainly based on the de-facto standard notation UML: the main OO design model is the class diagram as stated in usage surveys (Dobing and Parsons, 2006). Design metrics have been concentrated on the exploration of basic OO concepts like encapsulation, inheritance, polymorphism and class complexity. As required for optimizing applicability, our web system only considers design metrics applicable to UML class diagrams. A total of 32 different metrics described below were implemented.

2.1 Classical Metrics

A set of 20 classical metrics ranked into four categories of metrics have been considered. The first category includes the well-known CK metrics set (Chidamber and Kemerer, 1994) aimed at defining a measure of the design complexity in relation to their impact on external quality attributes such as maintainability, reusability, etc. These metrics are useful for predicting the frequency of changes through classes during the maintenance phase,

R. Hilera J., Fernández-Sanz L. and Cabello M. (2010).
WEB TOOL FOR OBJECT ORIENTED DESIGN METRICS.
In *Proceedings of the 5th International Conference on Software and Data Technologies*, pages 300-303
DOI: 10.5220/0003011003000303
Copyright © SciTePress

detecting possible design flaws or violations of design philosophy. The metrics implemented in the system are the following:

- WMC: Weighted Methods per Class
- DIT: Depth of Inheritance
- NOC: Number Of Children.

The second group of metrics are the ones proposed by Briand et al. (1997) as measure of coupling between classes:

- IC_Attr: Import Coupling with Class-Attribute interaction between a class and the rest
- EC_Attr: Export Coupling with Class-Attribute interaction between a class and the rest
- IC_Par: Import Coupling with Class-Method interaction between a class and the rest
- EC_Par: Export Coupling with Class-Method interaction between a class and the rest.

The third category includes MOOD metrics (Brito and Melo, 1996). The objective of these metrics is to define a measure of the use of mechanisms of OO design such as inheritance (MIF and AIF metrics), information hiding (MHF and AHF-metric) and polymorphism (the PF metric):

- MHF: Method Hiding Factor
- AHF: Attribute Hiding Factor
- MIF: Method Inheritance Factor
- AIF: Attribute Inheritance Factor
- PF: Polymorphism Factor

Finally, we have implemented the classical metrics proposed by Lorenz and Kidd (1994), both related to size (PIM, NIM, NIV, NCV, and NCM) and to inheritance (NMO, NMI, and NMA):

- PIM: Public Instance Methods
- NIM: Number of Instance Methods
- NIV: Number of Instance Variables
- NCM: Number of Class Methods
- NCV: Number of Class Variables
- NMO: Number of Methods Overridden
- NMI: Number of Methods Inherited
- NMA: Number of Methods Defined

2.2 Classification Metrics

We have also considered other 18 metrics, classified in three different categories, most of them described at (Genero et al. 2005). The first category is related with the size of the elements of a class:

- NumAttr: Number of attributes in a class
- NumOps: Number of operations in a class
- NumPubOps: Number of public operations in a class

Other group of metrics is focused on inheritance:

- NOC: Number Of Children (or direct descendents of a class)
- NumDesc: Number of descendents of a class
- NumAnc: Number of ancestors of a class
- DIT: Depth of Inheritance
- CLD: Class to Leaf Depth
- OpsInh: Number of inherited operations
- AttrInh: Number of inherited attributes

And a set of metrics related with coupling:

- Dep_Out: Number of elements on which a class depends
- Dep_In: Number of elements that depend on a class
- NumAssEl_ssc: Number of associated elements in the same scope (namespace) as a class
- IC_Attr: Import Coupling with Class-Attribute interaction between a class and the rest
- EC_Attr: Export Coupling with Class-Attribute interaction between a class and the rest
- IC_Par: Import Coupling with Class-Method interaction between a class and the rest
- EC_Par: Export Coupling with Class-Method interaction between a class and the rest.

3 XMI FORMAT TO REPRESENT UML CLASS DIAGRAMS

A UML modeling tool is a software application that supports some or all of the notation and semantics associated with the Unified Modeling Language (OMG, 2009). One of the features that should be considered when choosing and UML tool is the possibility of importing and exporting models using the XMI format. XMI (XML Metadata Interchange) is the format for UML model interchange (OMG, 2007).

XMI is a XML dialect that incorporates tags to represent UML diagrams. For class diagrams, listing 1 presents a code extract which corresponds to figure 1 class diagram (exported as a *ClassDiagram.xmi* file). Tags `<packagedElement>` with the attribute `type=Class` are used to represent classes, while `<ownedOperation>`, `<ownedAttribute>` and `<generalization>` tags represent attributes, operations and inherence associations.

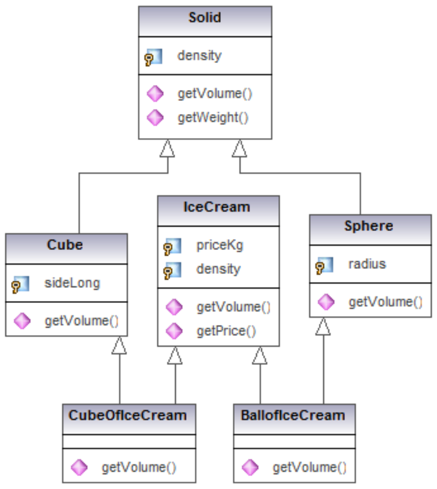


Figure 1: Class diagram example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI
 xmlns:xmi="http://schema.omg.org/spec/
 XMI/2.1"
 xmlns:uml="http://schema.omg.org/spec/
 UML/2.1.2" xmi:version="2.1">
 <uml:Package xmi:id="Root" name="Root">
  <packagedElement xmi:type="uml:Package"
   xmi:id="ComponentView" name="Component
   View" visibility="public"/>
   <packagedElement xmi:type="uml:Class"
    xmi:id="Solid" name="Solid"
    visibility="public">
    <ownedAttribute
     xmi:type="uml:Property"
     xmi:id="density" name="density"
     visibility="protected"/>
    <ownedOperation
     xmi:type="uml:Operation"
     xmi:id="getVolume" name="getVolume"
     visibility="public"/>
    <ownedOperation
     xmi:type="uml:Operation"
     xmi:id="getWeight" name="getWeight"
     visibility="public"/>
   </packagedElement>
   <packagedElement xmi:type="uml:Class"
    xmi:id="Cube" name="Cubo"
    visibility="public">
    <generalization
     xmi:type="uml:Generalization"
     xmi:id="Gen1" general="Solid"/>
    <ownedAttribute
     xmi:type="uml:Property" xmi:id=""
     name="sideLong"
     visibility="protected"/>
    <ownedOperation
     xmi:type="uml:Operation"
     xmi:id="getVolume" name="getVolume"
     visibility="public"/>
   </packagedElement>
...

  </packagedElement>
 </uml:Package>
</xmi:XMI>
```

Listing 1: Extract of the XMI file representing the class diagram in figure 1.

4 A WEB APPLICATION TO CALCULATE METRICS FOR UML CLASS DIAGRAMS

The developed system (figure 2) is basically an open source web application in C#. It includes two classes and an *aspx* web server page (figure 3). There is a class that manages the XMI files representing a UML class diagram; the other class implements the calculation of the metrics explained in section 2.

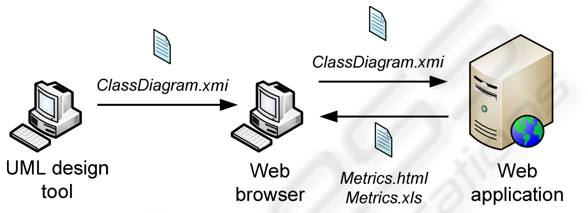


Figure 2: System architecture.

The first step when using the system to obtain a metrics report is the selection and uploading of the XMI file containing the UML class diagram to be measured (figure 4). After processing the XMI document, a complete report in two different formats, HTML (figure 5) and XLS (figure 6) is available so the user can observe and store the values of metrics.

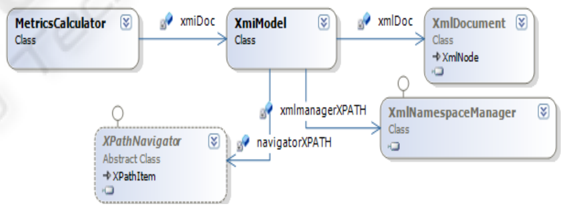
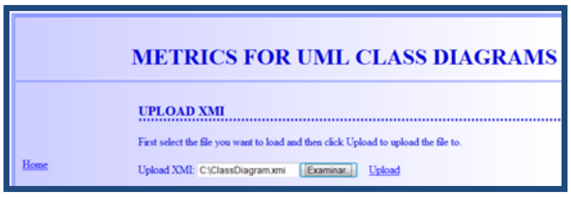


Figure 3: Basic application design.

The two principal classes of the application are:

- *XmiModel*: This class contains all the functionality for processing XMI files. It uses the standard classes *XmlDocument* and *XPathNavigator* from the .NET Framework libraries *System.Xml* and *System.Xml.XPath*.
- *MetricsCalculator*: This class includes the functionality related with the evaluation of the object oriented metrics. It contains an *XmiModel* object necessary to explore the XMI fie with the class diagram to be evaluated.



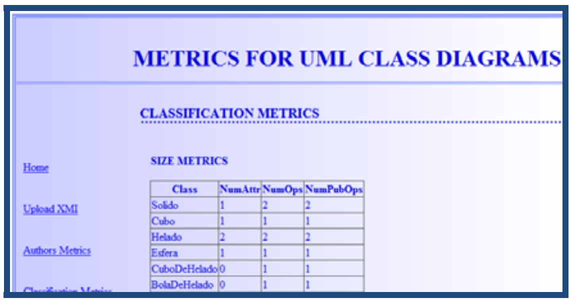
METRICS FOR UML CLASS DIAGRAMS

UPLOAD XMI

First select the file you want to load and then click Upload to upload the file to.

Upload XMI: C:\ClassDiagram.xmi

Figure 4: Part of the Web interface to upload the UML class diagram, encoded in XMI, to be evaluated.



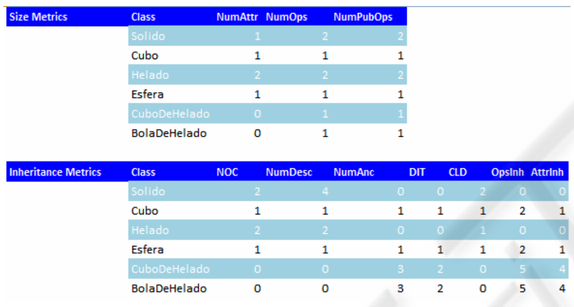
METRICS FOR UML CLASS DIAGRAMS

CLASSIFICATION METRICS

SIZE METRICS

Class	NumAttr	NumOps	NumPubOps
Solido	1	2	2
Cubo	1	1	1
Helado	2	2	2
Esfera	1	1	1
CuboDeHelado	0	1	1
BolaDeHelado	0	1	1

Figure 5: Extract of the metrics report in HTML format.



Size Metrics	Class	NumAttr	NumOps	NumPubOps
	Solido	1	2	2
	Cubo	1	1	1
	Helado	2	2	2
	Esfera	1	1	1
	CuboDeHelado	0	1	1
	BolaDeHelado	0	1	1

Inheritance Metrics	Class	NOC	NumDesc	NumAnc	DIT	CLD	OpsInh	AttrInh
	Solido	2	4	0	0	2	0	0
	Cubo	1	1	1	1	1	2	1
	Helado	2	2	0	0	1	0	0
	Esfera	1	1	1	1	1	2	1
	CuboDeHelado	0	0	3	2	0	5	4
	BolaDeHelado	0	0	3	2	0	5	4

Figure 6: Extract of the report in spreadsheet format.

5 CONCLUSIONS

Many software metrics to measure OO designs in general, and class diagrams in particular are available. Studies have checked their usefulness for developers in order to control important features such as inheritance and coupling as well as also those related with size, complexity, encapsulation and polymorphism.

In this paper we have presented a web application to evaluate 32 metrics for UML class diagrams. The application can analyze any class diagram in XMI format so any diagram created using almost any of the main current CASE tools could be used. XMI is a very important standard because allows UML models interchange between CASE tools in form of plain text files.

Our application now is being improved, adding more metrics as well as additional functionality to visualize graphically UML class diagrams (using SVG format supported by browsers) to help developers to check diagrams before staring analysis. This software has been developed as open source and it is available for downloading from [to be included in the final paper version].

REFERENCES

Briand L., Devanbu W., Melo W., 1997. An investigation into coupling measures for C++, *19th International Conference on Software Engineering*, pp. 412-421.

Brito e Abreu F., Melo W., 1996. Evaluating the Impact of Object-Oriented Design on Software Quality, *3rd International Metric Symposium*, pp. 90-99.

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

Dobing, B. and Parsons, J., 2006. How UML is used, *Communications of the ACM*, vol. 49, no. 5, pp. 109-113.

Fento, N. E., Pfleeger, S. L., 1997. *Software metrics: a rigorous and practical approach*, PWS.

Genero M., Piattini, M., Calero, C., 2005. A Survey of Metrics for UML Class Diagrams, *Journal of Object Technology*, vol. 4, no. 9, Nov.-Dec. 2005, pp. 59-92. www.jot.fm/issues/issue_2005_11/article1/.

Ghosheh, E., Black, S., 2009. WapMetrics: A tool for computing UML design metrics for Web applications, *7th ACS/IEEE International Conference on Computer Systems and Applications*, pp.682-689.

Lorenz M., Kidd J., 1994. *Object-Oriented Software Metrics: A Practical Guide*, Prentice Hall, Englewood Cliffs, New Jersey.

OMG, 2007. *XML Metadata Interchange (XMI).* Object Management Group, 2007. www.omg.org/spec/XMI/.

OMG, 2009. *Unified Modeling Language (UML).* Object Management Group, 2009. www.omg.org/spec/UML/.