Assessing Modeling Languages, metrics and tools

No Author Given

Department of Informatics, University of Minho
Campus de Gualtar, 4710-057 Braga, Portugal

Abstract. Any traditional engineering field has metrics to rigorously assess the quality of their products. From a long time ago, engineers know that the production process is not all; the output must comply with the rules and good-practices, must satisfy the requirements and must be competitive.
Professionals in the new field of software engineering started a few years ago to define metrics to appraise their product: individual programs and software systems.
This concern motivates the need to assess not only the outcome but also the process and tools employed in its development. In this context, assessing the quality of programming languages is a legitimate objective; in a similar way, it makes sense to be concerned will models and modeling approaches, as more and more people starts the software development process by a modeling phase.
In the paper we discuss the quality of modeling languages, introducing and motivating the topic, presenting metrics, and comparing tools.

Keywords: : Modeling Languages, Software/Language Quality, Software/Language Metrics, UML

1 Introduction

A model is a representation of reality aiming for the simplification of some complex object.

Models are built so that we can better understand the system being developed. They help us to visualize a system as it is or as we need it to be. Models allow us to specify the structure and behavior of a system, they provide the guidance lines/blueprints in constructing a system, and finally, models document the decision taken for a given system;

Some models are best described textually, other graphically. All interesting systems exhibit structures that transcend what can be represented in a programming language.

A modeling language is a language whose vocabulary and rules focus on the conceptual and physical representation of a system.

Specifying means building models that are precise, unambiguous and complete. The UML addresses the specification of all the important analysis, design and implementation decision.

That aside, when assessing a modeling language we might infer on its quality.

The software functionality, trustability, usability, efficiency, maintability and portability.

* Importância do uso de métricas num projecto: no que consistem, o que medem, o que ajudam a melhorar, etc. - esta parte liga-se com o que sugiro abaixo - sugiro partir de uma definição geral de métricas;

Nowadays, metrics become increasingly essential for Software Engineering: they are crucial for quality assessment and reengineering processes. In *Forward Engineering* they are used to measure the software quality and estimate cost and effort of software projects [FP98]. In the field of *Software Evolution*, they can be both used to identify stable or unstable parts of a system as to determine where refactoring can be or have been applied [DDN00]. They even can be used for assessing the quality and complexity of software systems in *Software Reengineering* or *Reverse Engineering* [CC90].

When focusing on the field of Object-Oriented (OO) systems, many metrics have been proposed for assessing the design of a software system. However, most of the existing approaches involve the analysis of the source code and cannot be applied in earlier stages of the development process. In fact, it is not always simple to apply the existing metrics in this earlier stages. As the Unified Modeling Language, proposed by Booch, Jacobson and Rumbaugh [BRJ05] has became a standard for expressing OO systems, apply metrics to these models enables an early estimate of development efforts, implementation time, complexity and cost of the system under development.

* Especificar o tipo de métricas sobre o qual nos vamos focar (UML) - o paragrafo acima ja fala 1pc, completar se tiverem+ideias

** estrutura do artigo - proposta a completar

In this paper, we will introduce and discuss the major existing metrics for UML models, and focus on present a set of tools designed for measure UML projects. In what follows, Section 2 is devoted to the metrics assessment process. In Section 3 we describe the principal measurements applicable to the most popular UML diagrams. Then, in Section 4 we present some tools designed for apply metrics to UML models and the results of applying them a real case-study. We conclude in Section 5 with a comparsion between the presented tools (.....).

2 Metrics Assessment

* Explicar aqui que as métricas não podem ser observadas, por si só, fora do contexto. Este problema prende-se essencialmente com o facto deste tipo de medições ser empírica, ou baseada em métodos empíricos.

> Effective management of any process requires quantification, measurement, and modeling. Software metrics provide a quantitative basis

> for the development and validation of models of the software development process. Metrics can be used to improve software productivity and quality. [Mil98]

* Explicar em maior detalhe o que se entendo como validação de métricas (Kaner e Walter Bond).

Fenton [Fen99] estimates that companies spend about 4% of the development budget in the establishment of metrics program, therefore, engineers should also certify and garantee that the applyed metrics actualy quantify, measure, and model the atributes of the system.

> This has a likely consequence: if a project or company is managed according to the results of measurements, and those metrics are inadequately validated, insuficiently understood, and not tightly linked to the attributes they are intended to measure, measurement distortions and dysfunctional should be commonplace. [KB04]

The IEEE Standard 1061 [IEE98] lays out a methodology for developing metrics for software quality attributes. The standard defines an attribute as *"a measurable physical or abstract property of an entity."* A quality factor is a type of attribute, *"a management-oriented attribute of software that contributes to its quality."*

A metric is defined as being a **measurement function**, and a **software quality metric** is defined as *"a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality."*

Any software metric must comply with the following criteria: correlation, consistency, tracking, predictability, discriminative power and reliability.

3 Applying Metrics To UML Models

–esta intro ficará para depois de termos as subsec prontas

As métricas para avaliação de software estão bem mais desenvolvidas que as métricas para linguagens de modelação. É então interessante ter em conta o estudo já realizado sobre métricas para avaliação de código, como é o caso das CK Metrics, como ponto de partida para métricas de modelação.

3.1 CK Metrics

One of the most popular suites of OO metrics was proposed by Chidamber and Kemerer [CK94]. They were proposed to capture different aspects of object-oriented designs, including complexity, coupling and cohesion, and were posteriorly adapted for modeling languages as we can see in [MP06].

This suite is composed by six metrics: Weighted Methods Per Class (WMC), Depth of Inheritance Tree (DIT), Number of Children (NOC), Coupling between Object Classes (CBO), Response For a Class (RFC), e Lack of Cohesion in Methods (LCOM). We detail bellow each metric and its features.

Weighted methods per class (WMC): This metric regards to the complexity of a class's methods, being equal to the sum of the complexities of those methods defined. There two kinds of WMC metrics:

- $WMC1_1$ can be obtained from a class diagram of a *UML* model. It is computed by identifying the class and counting the number of methods in that class, which means that, in this case, the WMC metric considers a method as a unity.
- WMC_{cc} is also computed by identifying the class and counting the number of methods, but each method is not a unity, is the result of a McCabe Cyclomatic Complexity of it. Another difference is that this kind of WMC cannot be computed only with information of diagrams class, needing information of other kinds of diagrams, like sequence or activity.

Depth of inheritance tree (DIT): This is metric is equal to the maximum length from the class to the root of the inheritance, which could be defined as the depth of the class. It is computed by taking the union of all the class diagrams in a *UML* model and traversing the inheritance hierarchy of the class.

Number of children (NOC): This metric represents the number of childs and descendants of a certain class. Can be obtained gathering all diagrams class, in a *UML* modulation, and checking all the inheritance relations of the class.

Coupling between object classes: Two classes are related if the method of a class uses a instance variable or method of another class. Counting the number of classes to which the class are related and counting all kind of references of the attributes and parameters of the methods of the class, an estimate of this metric's value can be obtained from the class diagrams. Though, it is possible to calcute a more precise value if the behavioural diagrams are taken into account, since the usage of instance variable and invocation methods are additional information.

Response for a class (RFC) - This metric is the number of methods that can be invoked by an object of a given class. It can be obtained from a class diagram and, also, by behaviour diagrams (e.g. sequece diagrams), which can inform of several methods of other classes that are invoked by each of the class's methods.

Lack of cohesion in methods (LCOM)- Is the metric that measure the number of sets of instance variables accessed by every pair of methods, of a given class, that has a non-empty intersection. With the intention of computing a value for this metric, it is essential the information of the usage instance variables by the methods of a class, i.e., it is needed a sequence diagram (a class diagram does not have information about the usage).

The set of metrics that were here defined can be found and cited in several papers (probably the most famous [MP06]). They are currently the most studied and used to evaluate *UML*models.

3.2 Use Case Metrics

The Use Cases Diagrams are graphical representations of entities which interact with the system - the *actors*, and operations that the system must perform for them. They define a sequence of actions which illustrate a specific way of using the system.

These diagrams are functional specifications, collected at the beginning of a system development process. They are crucial to an early estimate of the system complexity and its development efforts, as we can see by the UC metrics defined in several works like [KB02], [MAC05], [Rib01].

In fact, measuring the number of use cases, actors and communications among them is a good indicator of the system complexity, as well as to quantify the relationship between diagrams (i.e. estimates the number of UC that extends other, the number of UC that includes other, etc).

One remarkable work on this area was performed by Michele Marchesi [Mar98]. Table 1 illustrates the Use Case metrics defined on this work.

Marchesi Metrics

Metric	**Description**
NA	Number of actors of the system.
UC1	Number of Use Cases in the system.
UC2	Number of communications among UC and Actors
UC3	Number of communications among UC and Actores without redundancies
UC4	Global complexity of the system

Table 1. Marchesi Use Case Metrics

The **UC4** metric represents a balance of the global complexity of the system, and its value is obtained through the values of **UC1**, **UC2** and **UC3** metrics.

3.3 Class Diagram Metrics

These diagrams are used to describe the types of objects in a system and the relationships among them. They describe the structure of a system by showing the system classes, their attributes and methods or operations. Their quality have a huge impact on the final quality of the software under development, as they describe the general model of the system information.

Measures like the *Number of Attributes in the Class* (NAC), the *Number of Operations in the Class* (NOC), *Number of Inherited Attributes* (NIA), *Number of descents/ancestors od a Class*(NDC/NAC), or even the *Number of Interfaces Implemented* (NII) are used both for indicate the system complexity and as a index of quality. Many works present several metrics for this diagrams [GPC00], [Eic], [YWG04]. The OOA metrics defined by Marchesi [Mar98] also contemplate Class Diagrams, as we can see in Table 2.

Marchesi Metrics

Metric	**Description**
NC	Number of Classes
CL1	Weighted Number of responsibilities of a Class
CL2	Weighted Number of Dependencies of a Class
CL3	Depth of inheritance tree
CL4	Number of immediate subclasses of a given class
CL5	Number of distinct class

Table 2. Marchesi Class Diagram Metrics

Marchesi Metrics

Metric	**Description**
NP	Number of Packages.
PK1	Number of Classes
PK2	Weighted Number of responsibilities of a Class
PK3	Overall Coupling among Packages

Table 3. Marchesi Package Metrics

In UML, classes can be grouped in Packages to define subsystems or even for implementation purposes. The measurement of a package complexity is useful to forecast the development efforts of it. For that, we can measure properties like the *Number of Classes of a Package*, the *Total Number of Packages in the system*, or the *Number of Interfaces in the Package*. Marchesi [Mar98] suggests several Package Metrics as we can see in Table 3 that allow to estimate this complexity.

3.4 Other Metrics

Statechart diagrams illustrate the behavior of an object. They define different states of an object during its lifetime, which are changed by events. A *state* expresses an action of an object during a certain time, when it don't receive external stimulus nor is there any change in its attributes.

Métricas de Estados

Métrica	**Descrição**
TEffects	The number of transitions with an effect in the state machine.
TGuard	The number of transitions with a guard in the state machine.
TTrigger	The number of triggers of the transitions of the state machine.
States	The number of states in the state machine.

Table 4. Exemplos de Métricas sobre Diagramas de Estados

Measures like the *Number of Entry Actions*, *Number of Exit Actions*, *Number of Transitions*, or even the *Number of Activities* are associated to the complexity and dimension of the problem [GMP02]. In Table 4 we can notice some examples from measurable attributes for this type of diagram.

Activity diagrams describe work flows and are very useful for detail operations of a class (including behaviors expressed by parallel processing). As we can see in Table reft:act several metrics for this diagrams are available.

Métricas de Actividade

Métrica	**Descrição**
Actions	The number of actions of the activity.
ObjectNodes	The number of object nodes of the activity.
Pins	The number of pins on nodes of the activity.
Guards	The number of guards defined on object and control flows of the activity.

Table 5. Exemplos de Métricas sobre Diagramas de Actividade

Besides these metrics, it is possible to measure attributes like the *Number of Activity Groups/Zones*, the *Number of object flows* or even the *Number of Exceptions* of each diagram.

4 Tools for UML Metrics Specification

Nowadays, the most popular UML tools for software application development are Visual Paradigm for UML[1] and Poseidon for UML[2]. They enable a visual environment to model software, which reduces the complexity of software design. However, they not support metrics specification - it is required to use other tools, designed for this task. In the next subsections, we will introduce the leading systems on the quantitative analysis of UML models structural properties, and put them to test for explore their features with a real case-study.

One of the tools that we are going to address is SDMetrics[3], a design measurement tool for UML models. Although its core is open source and available under the GNU Affero General Public License, SDMetrics GUI it is not freely distributed. In order to explore all the system features, SDMetrics staff gently provided us an Academic License which we sincerely would like to thank.

The core functionalities of SDMetrics include:

- the configurable XMI parser for XMI1.0/1.1/1.2/2.0/2.1 input files;
- the metrics engine to calculate the user-defined design metrics;
- the rule engine to check the user-defined design rules.

The other tool we put to test is the Sparx System Enterprise Architect[4], a team-based modeling environment. It embraces the full product development

[1] Available at `http://www.visual-paradigm.com/product/vpuml/`
[2] Available at `http://www.gentleware.com/products.html`
[3] SDMetrics can be found at `http://www.sdmetrics.com/`
[4] `http://www.sparxsystems.com.au`

lifecycle, supporting both software design, requirements management, and metrics calculation for Use Case Diagrams. Thus, it allow to estimate the complexity of the project in a earlier stage, as well as the complexity associated with each actor of the system.

The case-study used to explore this tools is the modelation made for a project of a subject during our graduation. The subject name was Software Systems Development and it's project was *GCS - Gere Com Saber*, in english, Manages With Knowledge.
In the project, GCS is a company which doesn't provide services, but on the other hand, to meet it's clients needs, it has a wide range of suppliers, that GCS subcontracts, who are responsible for the service execution. Multiple suppliers can supply the same service and each service can be delivered in different ways. Each service can then be composed of multiple activities. As an example, there could be a service called *Shirts til 10Kg* and inside this service there could be activities such as *wash, iron, sewing buttons, etc.*
Each activitie of a given service as a stipulated price, and can be hired by a client.
The goal of the project was to model and implement a management software, with the same name as the company, that would help the completion of all the tasks inherent to the company.

As an example of the UML diagrams used on the test, we will display one image of the general Usecase diagram and an excerpt of the class diagram.

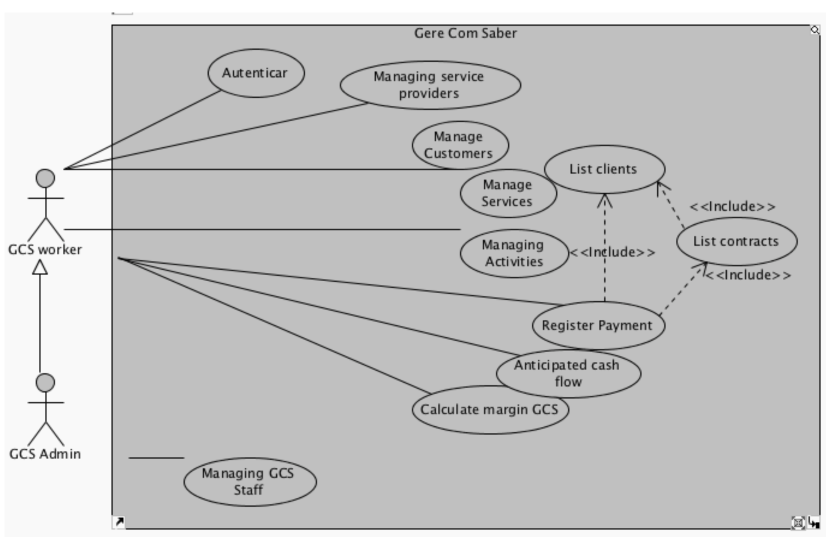


Fig. 1. The general Usecase of the modelation

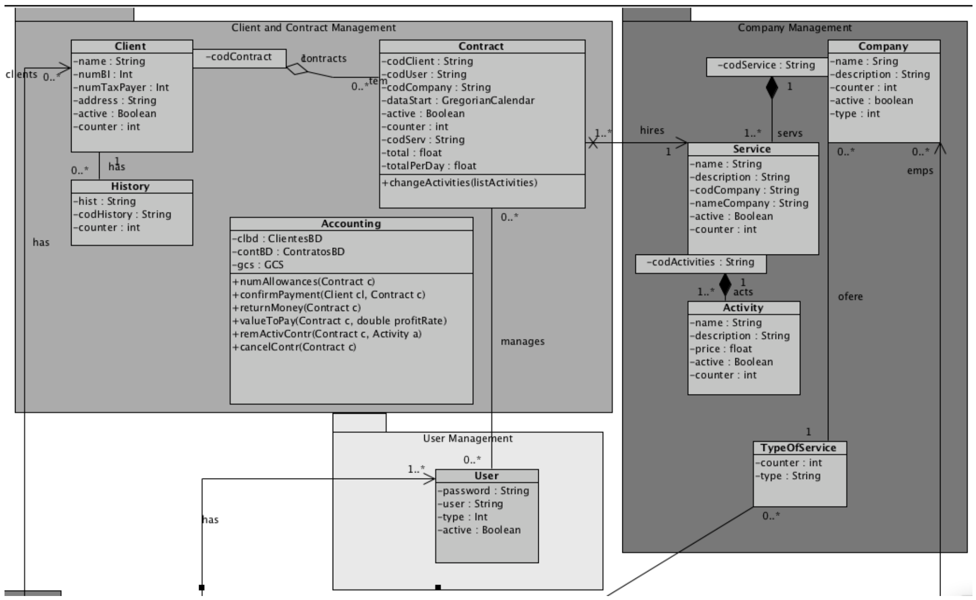


Fig. 2. Excerpt of the Class diagram

4.1 SDMetrics

SDMetrics is a very complete design measurement tool, analysing a wide range of UML diagrams, including Class, Usecase, Activity and Statemachine diagrams. For each type of diagram, this tool generates several metrics.
For example, **NumAttr** metric, one of the metrics that has already been adressed in this paper, is calculated from Class diagrams. Other one is **ExtPts**, wich is calculated from Usecase diagrams, and gives us the number of extension points of a given use case.

SDMetrics is written in java, and provides us a graphical user interface. The type of source files it receives to analyse are *XMI*[5] files, most modeling tools support project exportation in XMI.
This tool allows us to access the results from different views. We will approach the ones that seem the most important:

- **Metric Data Tables** provides a table that matches each UML model element analysed (table line) to it's value for each metric (table column);
- **Histograms** provides a graphical distribution for each design metric;
- **Design Comparison** provides us a mean to compare the structural properties of two *UML* designs. It is very useful to compare the same design in

[5] *XMI* (**X**ML **M**etadata **I**nterchange) is an *OMG* (Object Management Group) standard to generate XML-based representations of UML and other OO data models.

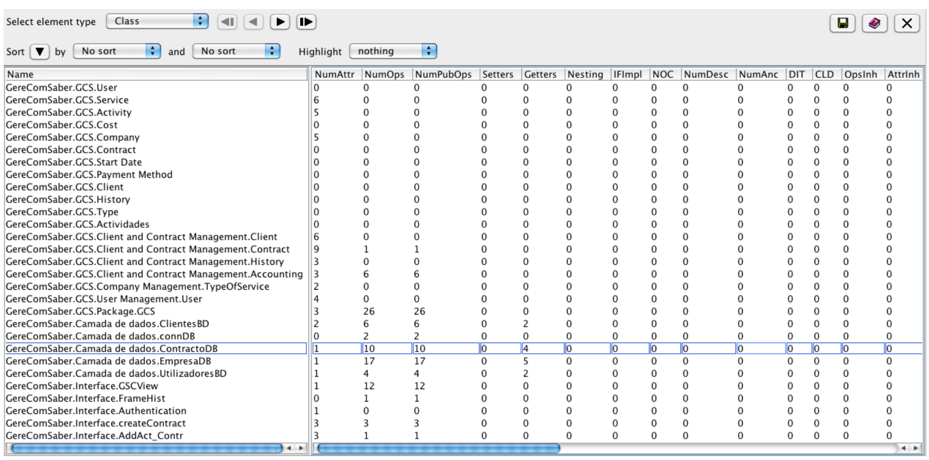
different iterations of the development, or to compare an alternative design to the current one.

- **Rule Checker** design rules and heuristics detect potencial problems in the UML design such as:
  - incomplete design such as unnamed classes, states without transitions;
  - violation of naming conventions for classes, attributes, operations, packages;
  - etc.
- **Catalog** this view provides us with the definitions of the metrics, design rules, and relation matrices for the current data set, and provides literature references and a glossary for them.

Not a view, but one of the most advanced features in this software is the possibility of defining Custom Design Metrics and Rules. The new metrics are defined in a XML file, with a very particular format, the *SDMetricsML* (SDMetrics Markup Language).
The SDMetrics tool doesn't provide a direct notion of good/bad quality of the design model. Despite that, on the SDMetrics website we can find several indications of how to interpret each output.

Results Next we will show some printscreen's taken from SDMetrics, which will illustrate some of the outputs of this software, for our case-study.

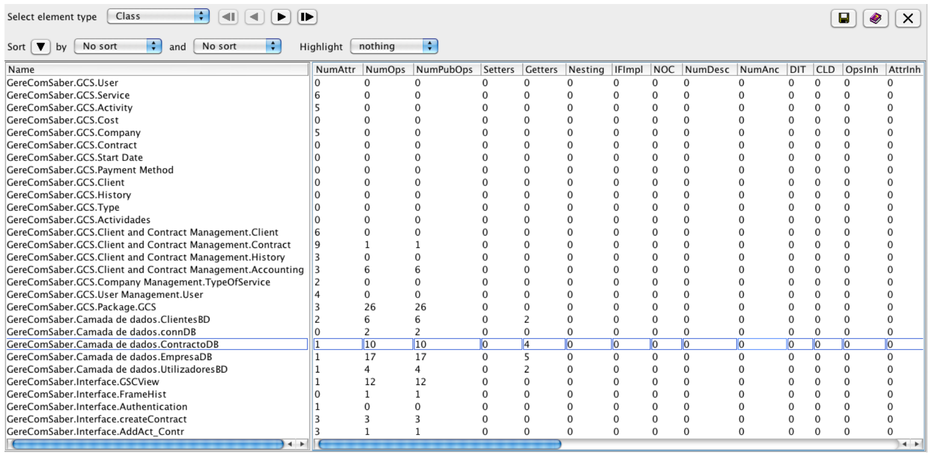
Metrics Table



Name	NumAttr	NumOps	NumPubOps	Setters	Getters	Nesting	IFImpl	NOC	NumDesc	NumAnc	DIT	CLD	OpsInh	AttrInh
GereComSaber.GCS.User	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Service	6	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Activity	5	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Cost	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Company	5	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Contract	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Start Date	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Payment Method	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.History	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Type	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Actividades	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client and Contract Management.Client	6	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client and Contract Management.Contract	9	1	1	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client and Contract Management.History	3	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client and Contract Management.Accounting	3	6	6	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Company Management.TypeOfService	2	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.User Management.User	4	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Package.GCS	3	26	26	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.ClientesBD	2	6	6	0	2	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.connDB	0	2	2	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.ContractoDB	1	10	10	0	4	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.EmpresaDB	1	17	17	0	5	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.UtilizadoresBD	1	4	4	0	2	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.GSCView	1	12	12	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.FrameHist	0	1	1	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.Authentication	1	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.createContract	3	3	3	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.AddAct_Contr	3	1	1	0	0	0	0	0	0	0	0	0	0	0

Fig. 3. Metrics Table for class diagrams

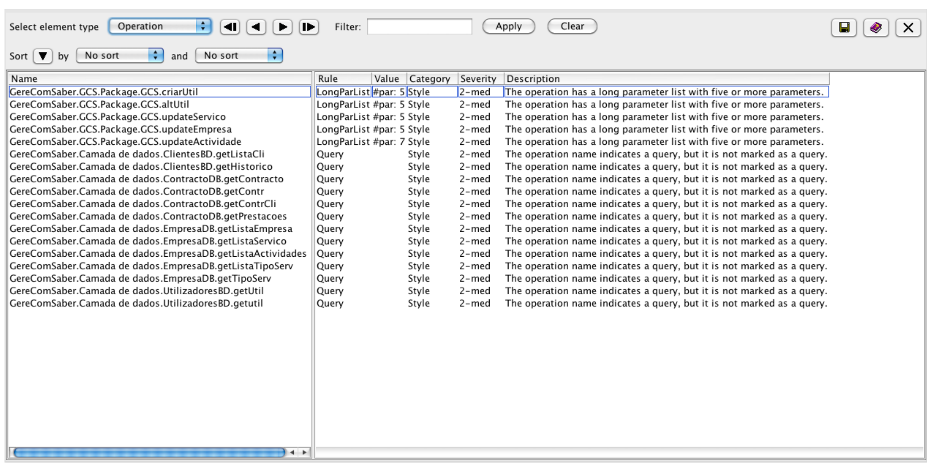
Histogram



Name	NumAttr	NumOps	NumPubOps	Setters	Getters	Nesting	IFImpl	NOC	NumDesc	NumAnc	DIT	CLD	OpsInh	AttrInh
GereComSaber.GCS.User	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Service	6	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Activity	5	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Cost	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Company	5	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Contract	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Start Date	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Payment Method	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.History	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Type	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Actividades	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client and Contract Management.Client	6	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client and Contract Management.Contract	9	1	1	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client and Contract Management.History	3	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Client and Contract Management.Accounting	3	6	6	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Company Management.TypeOfService	2	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.User Management.User	4	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.GCS.Package.GCS	3	26	26	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.ClientesBD	2	6	6	0	2	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.connDB	0	2	2	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.ContractoDB	1	10	10	0	4	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.EmpresaDB	1	17	17	0	5	0	0	0	0	0	0	0	0	0
GereComSaber.Camada de dados.UtilizadoresBD	1	4	4	0	2	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.GSCView	1	12	12	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.FrameHist	0	1	1	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.Authentication	1	0	0	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.createContract	3	3	3	0	0	0	0	0	0	0	0	0	0	0
GereComSaber.Interface.AddAct_Contr	3	1	1	0	0	0	0	0	0	0	0	0	0	0

Fig. 4. Histogram for class diagrams evaluating the metric NumAttr

Rule Checker



Name	Rule	Value	Category	Severity	Description
GereComSaber.GCS.Package.GCS.criarUtil	LongParList	#par: 5	Style	2-med	The operation has a long parameter list with five or more parameters.
GereComSaber.GCS.Package.GCS.altUtil	LongParList	#par: 5	Style	2-med	The operation has a long parameter list with five or more parameters.
GereComSaber.GCS.Package.GCS.updateServico	LongParList	#par: 5	Style	2-med	The operation has a long parameter list with five or more parameters.
GereComSaber.GCS.Package.GCS.updateEmpresa	LongParList	#par: 5	Style	2-med	The operation has a long parameter list with five or more parameters.
GereComSaber.GCS.Package.GCS.updateActividade	LongParList	#par: 7	Style	2-med	The operation has a long parameter list with five or more parameters.
GereComSaber.Camada de dados.ClientesBD.getListaCli	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.ClientesBD.getHistorico	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.ContractoDB.getContracto	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.ContractoDB.getContr	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.ContractoDB.getContrCli	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.ContractoDB.getPrestacoes	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.EmpresaDB.getListaEmpresa	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.EmpresaDB.getListaServico	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.EmpresaDB.getListaActividades	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.EmpresaDB.getListaTipoServ	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.EmpresaDB.getTipoServ	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.UtilizadoresBD.getUtil	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.
GereComSaber.Camada de dados.UtilizadoresBD.getutil	Query		Style	2-med	The operation name indicates a query, but it is not marked as a query.

Fig. 5. Rule checking for Operations

4.2 Sparx Systems Enterprise Architect

Sparx Systems Enterprise Architect is another tool that provides modeling of UML diagrams. It supports mind map diagrams and project management, to provide full traceability from requirements specification to deployment end implementation. This tool also provides also some metrics evaluation to compute the complexity of a project only based on Use Case diagrams.

To perform this evaluation, the user needs to provide a level of implementation complexity to each interaction with authors. This task can be done when defining the Use Case descriptions or when performing the metrics evaluation.

To evaluate the metrics, Enterprise Architect have a wizard, that gives also other options to do the complexity analysis. This options manipulates the Technical complexity and also Environment complexity and be used to adapt the evaluation and perform a better result on estimation.

Also can be filtered the Use Cases used in evaluation. To filter can be used manual selection or regular expressions over use case name. This kind of filtering ables the project to be distributed and evaluated individually.

The final result presented is the maximum number of working hours needed to perform the implementation.

Results

4.3 IBM Rational

–Permite modelação + aplicação de métricas

Results

5 Conclusion

Comparação entre as diversas ferramentas.

References

Grady Booch, James Rumbaugh, and Ivar Jacobson. *Unified Modeling Language User Guide, The (2nd Edition)*. Addison-Wesley Professional, 2 edition, May 2005.

E.J. Chikofsky and II Cross, J.H. Reverse engineering and design recovery: a taxonomy. *Software, IEEE*, 7(1):13 –17, jan 1990.

S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.*, 20:476–493, June 1994.

Serge Demeyer, Stéphane Ducasse, and Oscar Nierstrasz. Finding refactorings via change metrics. In *In Proceedings of OOPSLA '2000 (International Conference on Object-Oriented Programming Systems, Languages and Applications)*, pages 166–177. ACM Press, 2000.

Holger Eichelberger. On class diagrams, crossings and metrics.

Norman E. Fenton. Software metrics: Successes, failures and new directions. 47(2-3), July 1999. pages 149-157.

N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* International Thomson Computer Press, London, UK, 1998.

Marcela Genero, David Miranda, and Mario Piattini. Empirical validation of metrics for uml statechart diagrams, 2002.

Marcela Genero, Mario Piattini, and Coral Calero. Early measures for uml class diagrams. *L'OBJET*, 6(4), 2000.

IEEE. Ieee standard for a software quality metrics methodology. December 1998.

Hyoseob Kim and Cornelia Boldyreff. Developing software metrics applicable to uml models. In *6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE*, 2002.

Cem Kaner and Walter P. Bond. Software engineering metrics: What do they measure and how do we know? (num.14-16), September 2004.

Parastoo Mohagheghi, Bente Anda, and Reidar Conradi. Effort estimation of use cases for incremental large-scale software development. In *In: Proc of the 27th Int'l Conf. on Software Engineering. St Louis*, pages 303–311, 2005.

M. Marchesi. Ooa metrics for the unified modeling language. In *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering (CSMR'98)*, CSMR '98, pages 67–, Washington, DC, USA, 1998. IEEE Computer Society.

Everald E. Mills. Software metrics. December 1998.

Jacqueline A. Mcquillan and James F. Power. Some observations on the application of software metrics to uml models. Technical report, Department of Computer Science National University of Ireland, Maynooth, 2006.

Kirsten Ribu. Estimating object-oriented software projects with use cases. Technical report, University of Oslo, Dept, 2001.

Tong Yi, Fangjun Wu, and Chengzhi Gan. A comparison of metrics for uml class diagrams. *ACM SIGSOFT Software Engineering Notes*, 29:5, 2004.