# Métricas Chidamber and Kemerer

## Glossário

* **Coupling Between Objects (CBO)** - calcula o número de classes ou interfaces acopladas com a classe selecionada. Uma classe diz-se “acoplada” com outra se estabelece uma relação de dependência com a mesma (ignoram-se relações de dependência devido à hereditariedade)
* **Depth of Inheritance Tree (DIT)** - calcula a profundidade da árvore de hereditariedade para cada classe, isto é, a profundidade corresponde ao número de passos de hereditariedade entre a classe e java.lang.Object
* **Lack of Cohesion Of Methods (LCOM)** - calcula o grau de coesão de uma classe. Dois métodos de uma classe estão relacionados se eles partilham o uso de uma variável ou um chama o outro. A métrica é então o número de componentes do gráfico da relação. Valor 1 indica uma classe fortemente coesa (tornando difícil a separação em classes mais pequenas). Valores mais altos indicam geralmente que a classe devia ser separada noutras mais específicas.
* **Number Of Children (NOF)** - calcula o número de subclasses diretas de cada classe do projeto
* **Response For Class (RFC)** - dá-nos o total do número de construtores e métodos de uma classe mais o número de construtores e métodos que essa mesma classe pode chamar. Classes com um RFC elevado têm grande complexidade e podem ser menos estáveis.
* **Weighted Method Complexity (WMC)** - calcula o total da complexidade ciclomática (mede o número de caminhos de execução distintos ao longo de cada método - soma de if’s, while’s, for’s, etc) dos métodos de cada classe

# Métricas Chidamber and Kemerer - Valores mais relevantes

## Coupling Between Objects (CBD)

| **Classe** | **CBD** |
| --- | --- |
| net.sourceforge.ganttproject.language.GanttLanguage | 138 |
| net.sourceforge.ganttproject.GanttProject | 122 |
| net.sourceforge.ganttproject.action.GPAction | 103 |

## 

## Depth of Inheritance Tree (DIT)

| **Classe** | **DIT** |
| --- | --- |
| net.sourceforge.ganttproject.ResourceTreeTable | 8 |
| net.sourceforge.ganttproject.GanttTreeTable | 8 |
| net.sourceforge.ganttproject.GanttProject | 7 |

## Lack of Cohesion Of Methods (LCOM)

| **Classe** | **LCOM** |
| --- | --- |
| net.sourceforge.ganttproject.export.ConsoleUIFacade | 33 |
| net.sourceforge.ganttproject.GanttProjectBase | 31 |
| net.sourceforge.ganttproject.gui.UIUtil | 22 |

## Number Of Children (NOF)

| **Classe** | **NOF** |
| --- | --- |
| net.sourceforge.ganttproject.action.GPAction | 86 |
| net.sourceforge.ganttproject.test.task.TaskTestCase | 32 |
| net.sourceforge.ganttproject.parser.AbstractTagHandler | 19 |

## Response For Class (RFC)

| **Classe** | **RFC** |
| --- | --- |
| net.sourceforge.ganttproject.GanttProject | 382 |
| net.sourceforge.ganttproject.task.TaskManagerImpl | 260 |
| biz.ganttproject.impex.msproject2.ProjectFileImporter | 248 |

## Weighted Method Complexity (WMC)

| **Classe** | **WMC** |
| --- | --- |
| net.sourceforge.ganttproject.task.TaskManagerImpl | 173 |
| net.sourceforge.ganttproject.GanttProject | 172 |
| org.ganttproject.chart.pert.ActivityOnNodePertChart | 142 |

## 

## 

# 

# Comentários (Métricas Chidamber and Kemerer)

Através da análise dos gráficos e tabelas, podemos concluir que há diversas classes com uma complexidade extremamente elevada, o que origina dificuldades na escrita de testes unitários e pode tornar a classe pouco estável. Exemplo disto é por exemplo a classe GanttProject (uma das classes centrais do programa), que é extremamente extensa e, como podemos ver pelas tabelas acima, tem um CBD, um RFC e um WMC muito elevados (o que nos leva a concluir o que expliquei acima).

Também com base nas tabelas acima, podemos concluir que classes como GanttProjectBase poderiam ser subdivididas em classes mais pequenas de forma a tornar o código mais fácil de testar, já que, embora não seja uma das classes mais extensas, os métodos que lhe pertencem são pouco coesos e era possível melhorar o código nesta parte. Durante o desenvolvimento das funcionalidades, reparámos que esta classe tinha em comentário uma espécie de “separador” que indicava que os métodos a partir daquele ponto estavam mais relacionados com a interface UIFacade, o que reforça a possibilidade de haver essa separação em classes mais concretas.

Com base nos valores da Depth of Inheritance Tree, podemos reforçar novamente o facto de o código ser demasiado complexo, o que se revela diversas vezes problemático. É certo que uma classe com um elevado DIT, não implica code smells, mas pode tornar difícil prever o comportamento da classe devido aos métodos herdados. Este foi certamente um dos problemas com que mais nos deparámos ao desenvolver as funcionalidades porque tivémos de visitar diversas classes para entender como funcionaria certo método que estávamos a tentar chamar, tornando muito mais lenta a implementação de novos acrescentos ao programa.

# Distribuições das Métricas Chidamber and Kemerer











