

Metrics:
Chidamber & Kemerer
Pedro Grilo
59213
P3

1 Metric Chidamber & Kemerer:

1.1 CBO (Coupling between Object Classes)

Descrição: É quando uma classe está acoplada uma à outra, como por exemplo, uma classe usar métodos ou variáveis de instância definidas pela outra classe.

1.2 DIT (Depth of Inheritance Tree):

Descrição: É quando uma classe depende de muitas classes hierarquicamente, pois possivelmente esta classe irá herdar muitos métodos e variáveis e assim de grosso modo irá ficar confusa.

1.3 LCOM (Lack of Cohesion of Methods):

Descrição: Mede o quanto os métodos de uma classe estão relacionados uns com os outros. Por exemplo uma classe muito coesa, significa que existe muitos acoplamentos entre os métodos da classe.

1.4 NOC (Number of Children):

Descrição: Quantidade de todos os filhos diretos de uma certa classe base.

1.5 RFC (Response for a Class):

Descrição: É um conjunto de métodos de uma certa classe que podem ser chamados/instanciados através de outra classe.

1.6 WMC (Weighted Methods Per Class):

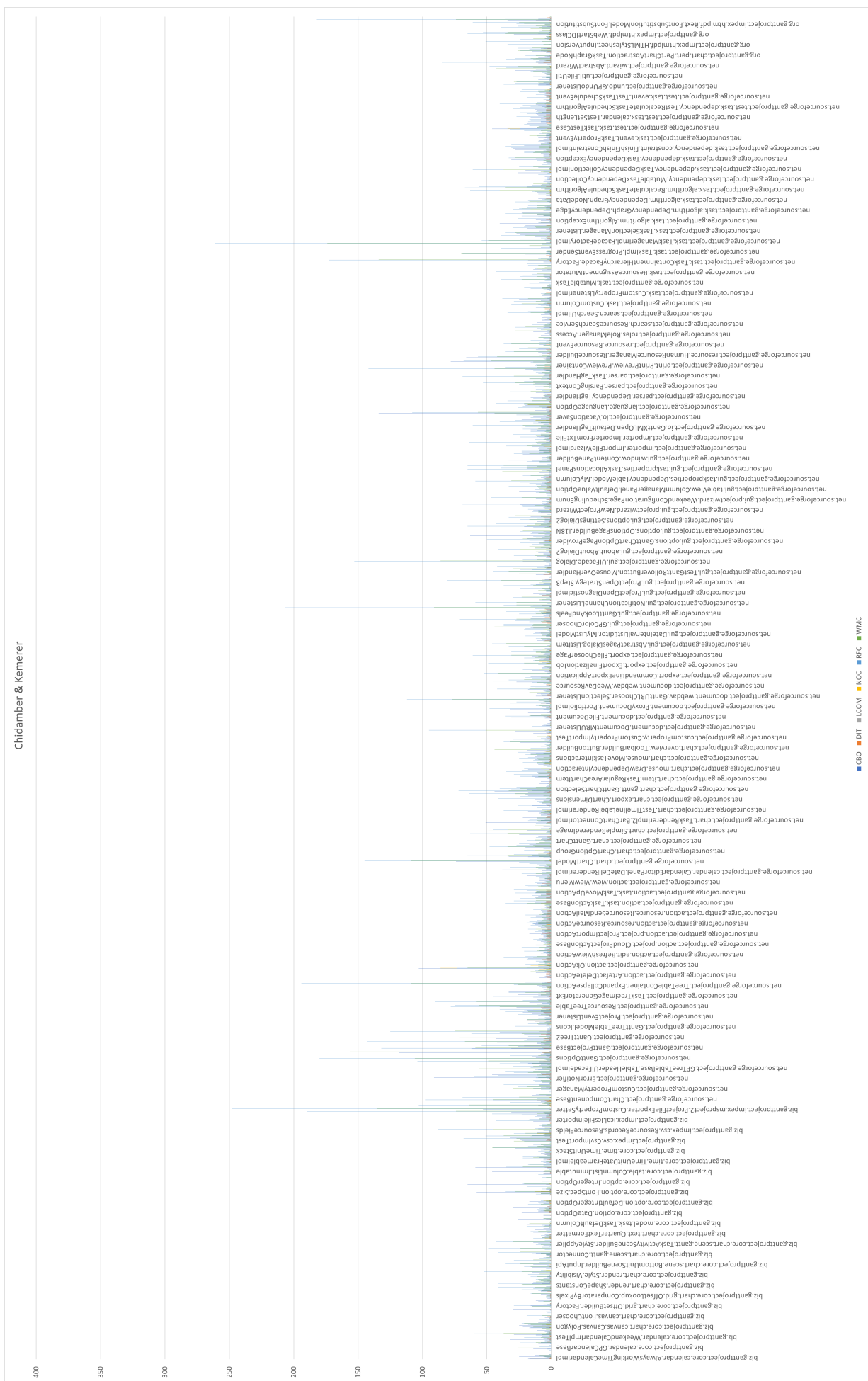
Descrição: O WMC, é um tipo de indicador de quanto tempo e esforço será preciso para alterar os métodos de uma certa classe de modo a implementar alguma funcionalidade. Por exemplo, uma classe que tenha muitos métodos, serão mais impossíveis de editar, sendo assim consecutivamente mais difíceis de reutilizar.

2 Potencial trouble spots in the codebase:

2.1 Análise do gráfico:

Descrição: Após observar o gráfico abaixo e os dados no excel, podemos perceber que existe um ficheiro que tem valores de RFC realmente altos (368), ou seja, para este projeto está praticamente 17 vezes acima da média (21).

Este ficheiro é o seguinte: net.sourceforge.ganttproject.GanttProject



2.2 Relacionar a classe com o RFC:

Esta classe é a classe Main deste projeto (tem também incluído o método Main), ou seja, todas as outras classes precisarão de a chamar para realizar alguma ação no programa, pois esta inclui todas as Roles, todos os Recursos Humanos, todos os Managers, todas as Tasks, etc...

A solução deste problema passaria a haver uma classe topo, tal como era feito na unidade curricular de POO, onde os métodos de logica iriam para essa tal classe e a classe da Main servia só para apresentação de conteúdo. Por exemplo, nesta classe existem uma quantidade de Setters e Getters, tal como é apresentado abaixo, onde simplesmente podiam estar nessa tal classe topo.

```
@Override
public String getDescription() { return prjInfos.getDescription(); }
↳ dbarashev
@Override
public void setDescription(String description) {...}
5 usages ↳ dbarashev
@Override
public String getOrganization() { return prjInfos.getOrganization(); }
1 usage ↳ dbarashev
@Override
public void setOrganization(String organization) {...}
↳ dbarashev
@Override
public String getWebLink() { return prjInfos.getWebLink(); }
↳ dbarashev
@Override
public void setWebLink(String webLink) {...}
↳ dbarashev
@Override
public HumanResourceManager getHumanResourceManager() {...}
↳ dbarashev
@Override
public TaskManager getTaskManager() { return myTaskManager; }
↳ dbarashev
@Override
public RoleManager getRoleManager() {...}
```

Outro problema desta classe é ser uma classe muito extensa - Code Smell (Long Class) - pois tem cerca de 1300 linhas, e caso seja preciso acrescentar uma funcionalidade geral no programa, irá ser complicada.