Uma imagem com texto, ClipArt

Descrição gerada automaticamente

Relatório

Engenharia de Software

João Esteves 47994

Nádia Mendes 53175

José Morgado 59457

Introdução

Nesta segunda fase do projeto, era-nos pedido que implementássemos duas novas funcionalidades ao programa GanttProject, ao analisarmos este detalhadamente verificamos algumas lacunas na parte da criação de recursos, assim sendo decidimos direcionar o nosso foco a alterar essas mesmas lacunas de entre as quais se destacaram:

O facto de ser possível criar um recurso sem nome atribuído, o que não no pareceu logico, uma vez que um objeto recurso corresponde a um funcionário concreto existente equipa do manager.

Deste modo decidimos que esta seria a nossa primeira funcionalidade, a impedição de se poder criar um recurso sem nome atribuído no programa GanttProject.

Verificamos também que era possível criar dois recursos exatamente iguais, isto é, com o mesmo nome atribuído e a mesma função, o que achamos descabido, uma vez que não nos fazer sentido criar objetos recurso diferentes que no fundo vão corresponder ao mesmo membro da equipa.

Esta foi a nossa escolha para a segunda funcionalidade extra a implementar no projeto GanttProject.

Para realizarmos estas duas novas funcionalidades, começamos por definir o use case diagram dos recursos, de seguida seguimos para a sua implementação usando o programa StarUML.

Achamos por bem também definir um use case diagram diagrama de recursos, uma vez que os recursos estavam implícitos neste. Definimos tambem um use case diagram para o gantt uma vez que as tarefas existentes neste se iam relacionar diretamente quer com as tarefas atribuídas aos recursos contidos na equipa quer ao diagrama de recursos. De seguida seguimos para a sua implementação usando o programa StarUML,

Numa segunda fase criamos documentação que continha as descrições de todos os use case utilizados nos diagramas acima referidos, achamos que teria bastante utilidade para que obtivéssemos uma melhor precessão do comportamento do programa e das novas funcionalidades que pretendíamos implementar. De seguida criamos user stories para cada um dos use case descritos na documentação.

Numa terceira fase seguimos para a implementação no código Java das novas funcionalidades e os subsequentes testes unitários, para que pudéssemos testar as mesmas.

Por fim escolhemos algumas métricas para analisar, nomeadamente Mood metrics, Dependency metrics e Complexity metrics a fim de relacionarmos com alguns code smells detetados na primeira fase deste projeto.

Use Case Diagrams

Diagrama de Recursos:

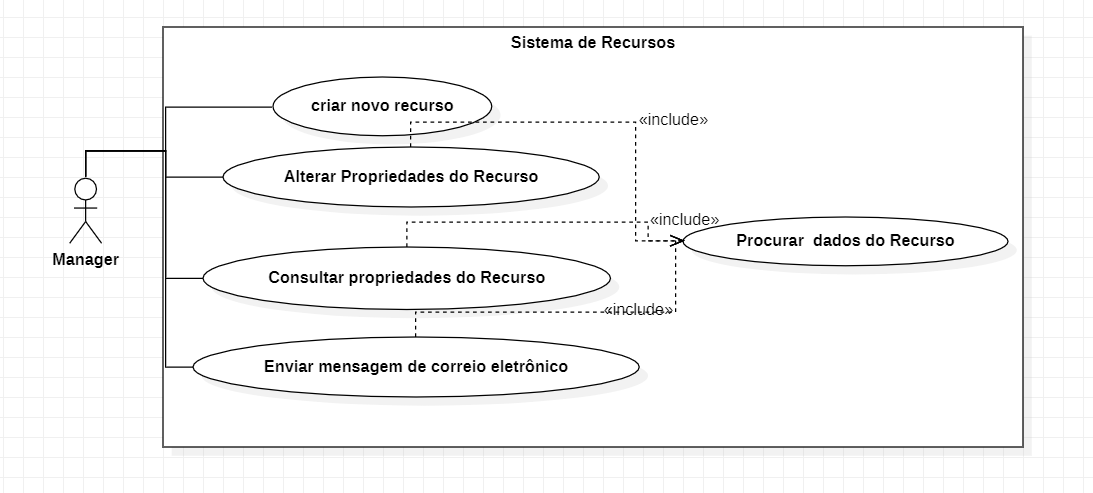
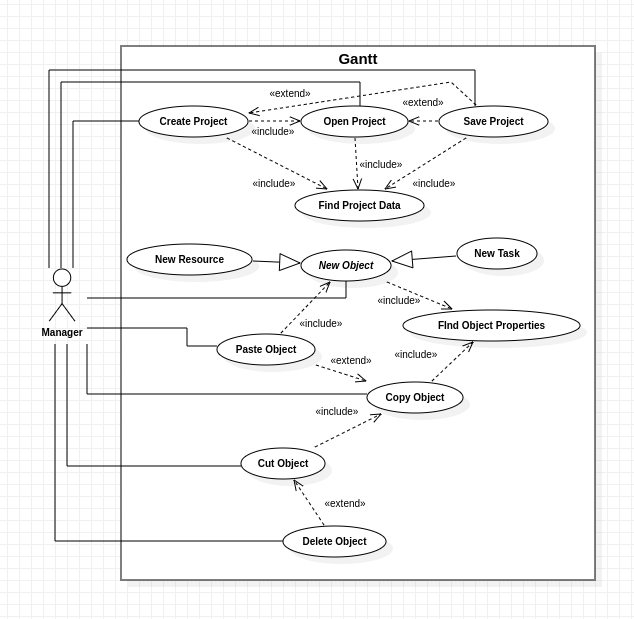
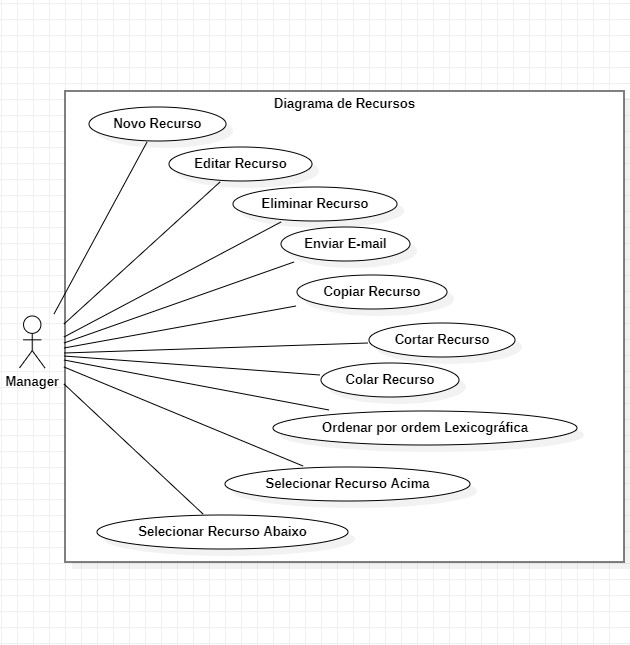


Diagrama do gantt:

Diagrama do Diagrama de recursos:

Estes foram os diagramas que achamos uteis para uma melhor precessão do projeto e da implementação das novas funcionalidades.

Use Case Descriptions

**Use case:** Criar novo recurso

**ID:** 1

**Description:** O manager cria um novo recurso

**Main actor:** Manager

**Secondary actor:** Nenhum

**Pré-conditions:** Nenhuma

**Main flow:**

1. Use case começa quando o manager seleciona a opção novo recurso.
2. O manager introduz no a informação do novo recurso a criar.
   1. O manager introduz no geral os dados relativos ao nome do recurso, telefone, email, a função do novo recurso da equipa seguindo-se os dados relativos à taxa de pagamento do recurso a criar.
   2. O manager introduz os dias de folga, caso já tenha essa informação.
   3. O manager introduz informação nas colunas personalizadas.
   4. O manager pode atribuir tarefas ao novo recurso a criar, caso estas já existam no sistema.
3. O sistema verifica se o manager atribuiu nome ao recurso a criar.
4. Include (procurar dados no sistema dos recursos existentes).

4.1 O sistema verifica se não existe na base de dados nenhum outro recurso com o mesmo nome e role.

1. O sistema valida os dados do recurso a criar introduzidos pelo manager.
2. O sistema cria um novo recurso.

**Post conditions:** Os dados do novo recurso são guardados no sistema.

**Alternative flows:**

Cancelar

Recurso sem nome atribuído

Recurso já existente

**Alternative flow:** Cancelar

**ID:** 1.1

**Brief description:** O manager informa o Sistema que deseja cancelar a criação do novo recurso.

**Primary actors:** Manager

**Secondary actors:** Nenhum

**Preconditions:** Nenhuma

**Alternative flow:**

1. O alternative flow inicia-se após o manager selecionar a opção novo recurso, passo 1.

**PostConditions:** Nenhuma

**Alternative flow:** Recurso sem nome atribuído

**ID:** 5.2

**Brief description:** O Sistema informa o manager que o recurso a criar não possui nenhum nome atribuído.

**Primary actors:** Manager

**Secondary actors:** Nenhum

**Preconditions:** O manager não introduz o nome do recurso a criar.

**Alternative flow:**

1. O alternative flow começa no passo 3.
2. O sistema avisa o manager de que este não introduziu nome ao recurso a criar.

**PostConditions:** Nenhuma

**Alternative flow:** Recurso já existente

**ID:** 5.3

**Brief description:** O Sistema informa o manager que já existe um recurso, na base de dados igual ao que este esta a tentar criar.

**Primary actors:** Manager

**Secondary actors:** Nenhum

**Preconditions:** O manager introduz os dados de um recurso já existente na base de dados do sistema.

**Alternative flow:**

1. O alternative flow começa no passo 4.1.
2. O sistema avisa o manager de que na base de dados já existe na base este recurso.

**PostConditions:** Nenhuma

**Use case:** Consultar propriedades do recurso

**ID:** 2

**Description:** O manager consulta as propriedades do recurso

**Main actor:** Manager

**Secondary actor:** Nenhum

**Pré-conditions:** Têm de existir pelo menos um recurso na base de dados do sistema.

**Main flow:**

1. O use case começa quando o manager seleciona o recurso a consultar
2. Include (procurar dados do recurso)
3. O manager seleciona a opção propriedades do recurso

3.1 O manager visualiza no geral os dados relativos ao nome do recurso, telefone, email, a função do novo recurso da equipa seguindo-se os dados relativos à taxa de pagamento do recurso a visualizar.

* 1. O manager introduz no geral os dados relativos ao nome do recurso, telefone, email, a função do novo recurso da equipa seguindo-se os dados relativos à taxa de pagamento do recurso a criar.
  2. O manager visualiza os dias de folga, caso já tenha essa informação.
  3. O manager visualiza a informação nas colunas personalizadas.
  4. O manager pode visualizar as tarefas do novo recurso, caso este já tenha tarefas atribuídas no sistema.

**Post conditions:** Nenhuma

**Alternative flows:** Cancelar

**Alternative flow:** Cancelar

**ID:** 2.1

**Brief description:** O manager informa o Sistema que deseja cancelar a visualização do recurso.

**Primary actors:** Manager

**Secondary actors:** Nenhum

**Preconditions:** Nenhuma

**Alternative flow:**

1. O alternative flow inicia-se após o manager selecionar a opção propriedades do recurso no passo 3.

**PostConditions:** Nenhuma

**Use case:** Alterar propriedades do recurso

**ID:** 3

**Description:** O manager altera as propriedades de um recurso selecionado, contido na base de dados do sistema.

**Main actor:** Manager

**Secondary actor:** Nenhuma

**Pré-conditions:** Têm de existir pelo menos um recurso na base de dados do sistema.

**Main flow:**

1. O use case começa quando o manager seleciona o recurso a alterar.
2. Include ( procurar dados do recurso).
3. O manager seleciona a opção propriedades do recurso.
4. O manager pode introduzir alterações no geral, onde se situam os dados relativos ao nome do recurso, telefone, email, a função do novo recurso da equipa seguindo-se os dados relativos à taxa de pagamento do recurso a criar.
5. O manager pode alterar ou adicionar os dados relativos aos dias de folga do recurso em questão.
6. O manager pode alterar ou introduz informação nas colunas personalizadas.
7. O manager pode atribuir tarefas ao recurso a alterar, caso estas já existam no sistema.
8. O sistema atualiza a base de dados, os novos dados do recurso a alterar.

**Post conditions:** Os dados do recurso são alterados na base de dados do sistema.

**Alternative flows:** Cancelar

**Alternative flow:** Cancelar

**ID:** 3.1

**Brief description:** O manager informa o Sistema que deseja cancelar a alteração do recurso.

**Primary actors:** Manager

**Secondary actors:** Nenhum

**Preconditions:** Nenhuma

**Alternative flow:**

1. O alternative flow inicia-se após o manager selecionar a opção propriedades do recurso no passo 3.

**PostConditions:** Nenhuma

**Use case:** Enviar mensagem de correio eletrónico

**ID:** 4

**Description:** O manager envia um email a um dado recurso.

**Main actor:** Manager

**Secondary actor:** Nenhuma

**Pré-conditions:** Têm de existir pelo menos um recurso na base de dados do sistema.

**Main flow:**

1. O use case começa quando o manager seleciona o recurso ao qual quer enviar o email em questão.
2. Include (procurar dados do recurso na base de dados)
3. O manager seleciona a opção enviar mensagem de correio eletrônico.
4. If o recurso tenha associado o email, basta manager redigir o email.
5. Else se o recurso não possuir nenhum email associado a si, o manager necessita introduzir o email do recurso que deseja contactar.

5.1 O manager escreve a mensagem a enviar via email ao recurso selecionado.

6. O manager seleciona a opção enviar

7. O sistema envia o email ao recurso selecionado

**Post conditions:** O email deverá ter sido enviado pelo sistema .

**Alternative flows:** Cancelar

**Alternative flow:** Cancelar

**ID:** 4.1

**Brief description:** O manager informa o Sistema que deseja cancelar a o envio da mensagem de correio eletronico.

**Primary actors:** Manager

**Secondary actors:** Nenhum

**Preconditions:** Nenhuma

**Alternative flow:**

1. O alternative flow inicia-se após o manager selecionar a opção enviar mensagem de correio eletrônico.

**PostConditions:** Nenhuma

|  |
| --- |
| **Use case:** Find Project Data |
| **ID:** 5 |
| **Description:** The manager finds the data belonging to the selected project |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** Project is selected |
| **Main flow:** |
| **1-** The manager introduces the identification of the project to be searched in the system |
| **2-** The manager introduces |
| **3-** The system finds the data belonging to the project |
| **Post conditions:** |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** Create Project |
| **ID:** 6 |
| **Description:** The manager creates a new Project in the file system |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** None |
| **Main flow:** |
| Extension point: Manager said yes to the prompt to save first |
| **1-** The use case starts when the manager selects the option to create a new project |
| **2-** The manager inputs the data for the new project (name, organization, link, description, domain, weekends and public holidays) |
| **3-** Include(FInd Project Data) |
| **4-** The system saves it on the project data |
| **5-** Include(Open Project) |
| **Post conditions:** The Project was created |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** Open Project |
| **ID:** 7 |
| **Description:** The manager opens an existing project from the file system |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** None |
| **Main flow:** |
| Extension point: Manager said yes to the prompt to save first |
| **1-** The use case starts either when the manager selects the option to open an existing project, when they created a new project or after they open ganttproject |
| **2-** If the manager selected the option to open an existing project |
| **2.1-** The manager selects which project to open from the file system |
| **3-** Else if the manager created a new project |
| **3.1-** The system selects the newly created project |
| **4-** Else if gantt has just opened |
| **4.1-** The systems selects the last opened project |
| **5-** Include(Find Project Data) |
| **6-** The system opens the project for view and editing |
| **Post conditions:** Project is open |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** Save Project |
| **ID:** 8 |
| **Description:** The manager saves the open project to a selected file name |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** Project is open |
| **Main flow:** |
| **1-** The use case starts when the Manager selects the option to save the project or when they said yes to the prompt to save first |
| **2-** Include(FInd Project Data) |
| **3-** The manager selects a directory and inputs a file name |
| **4-** The system saves the project to the file name on the chosen directory |
| **Post conditions:** Project is saved |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** New Resource |
| **ID:** 9 |
| **Description:** The manager creates a new resource in the system |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** None |
| **Main flow:** |
| **1-** (o1.) The use case starts when the manager selects the option to create a new resource |
| **2-** (o2.) The system asks the manager to define the properties of then new resource, which include name, phone, email, role, payment information, days off, custom columns and assignments |
| **3-** The manager inserts the properties |
| **4-** Includes(Find Object Properties) |
| **5-** (o5.) The system creates a new resource into the system with the provided properties |
| **Post conditions:** Resource Created |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** New Task |
| **ID:** 10 |
| **Description:** The manager creates a new task in the system |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** None |
| **Main flow:** |
| **1-** (o1.) The use case starts when the manager selects the option to create a new task |
| **2-** Includes(Find Object Properties) |
| **3-** (o3.) The system creates a new task into the system with default properties |
| **Post conditions:** Task Created |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** New Object |
| **ID:** 11 |
| **Description:** The manager creates a new object in the system |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** None |
| **Main flow:** |
| **1-** The case study starts when the manager selects the option to create an object |
| **2-** The system asks the manager to define the object's properties |
| **3-** The manager inserts the properties |
| **4-** Includes(FInd Object Properties) |
| **5-** The system creates a new object into the system with the properties given |
| **Post conditions:** None |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** Find Object Properties |
| **ID:** 12 |
| **Description:** The manager finds the data related to the object |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** None |
| **Main flow:** |
| **1-** The manager introduces the object to be searched in the system |
| **2-** The system find the properties of the object |
| **Post conditions:** |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** Copy Object |
| **ID:** 13 |
| **Description:** The manager copies one of the objects |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** None |
| **Main flow:** |
| **1-** The use case starts when the manager selects the copy option |
| **2-** Includes(Find Object Properties) |
| **3-** The system copies the selected object |
| **Extension point:** Paste |
| **Post conditions:** Object Copied |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** Paste Object |
| **ID:** 14 |
| **Description:** The manager creates a copy of the copied object |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions for segment 1:** Paste |
| **Segment 1 flow:** |
| **1-** The use case starts when the manager chooses the paste option |
| **2-** Include(New Object) |
| **3-** The system creates a copy of the copied object |
| **Post conditions for segment 1:** None |

|  |
| --- |
| **Use case:** Cut Object |
| **ID:** 15 |
| **Description:** The manager cuts an object |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions:** None |
| **Main flow:** |
| **1-** The use case starts when the manager chooses the cut option |
| **2-** Include(Copy Object) |
| **Extension point:** Cut |
| **Post conditions:** |
| **Alternative flows:** None |

|  |
| --- |
| **Use case:** Delete Object |
| **ID:** 12 |
| **Description:** The manager deletes an object from the system |
| **Main Actor:** Manager |
| **Secondary Actor:** None |
| **Pre-conditions for segment 1:** Cut and Paste |
| **Segment 1 flow:** |
| **1-** The system deletes the object |
| **Post conditions for segment 1:** Object deleted |
| **Pre-conditions:** None |
| **Main flow:** |
| **1-** The use case starts when the manager selects the delete option |
| **2-** The system deletes the object |
| **Post conditions:** Object deleted |

**Use case:** Eliminar um recurso

**ID:** 16

**Description:** O manager seleciona a opção eliminar recurso

**Main actor:** Manager

**Secondary actor:** Nenhum

**Pré-conditions:** Existir pelo menos um recurso na base de dados do sistema e selecionar esse mesmo recurso.

**Main flow:**

1. Use case começa quando o manager seleciona a opção excluir recurso.
2. Include (procurar dados no sistema dos recursos existentes).

**Post conditions:** Os dados do recurso são eliminados da base de dados do sistema.

**Alternative flows:** Nenhum

**Use case:** Copiar recurso

**ID:** 17

**Description:** O manager seleciona a opção copiar recurso

**Main actor:** Manager

**Secondary actor:** Nenhum

**Pré-conditions:** Existir pelo menos um recurso na base de dados do sistema e selecionar esse mesmo recurso.

**Main flow:**

1. Use case começa quando o manager seleciona a opção copiar recurso.
2. Include (procurar dados no sistema dos recursos existentes).
3. Coloca uma copia dos dados do recurso selecionado numa espécie de pilha.

**Post conditions:** Os dados do recurso são colocados numa espécie de pilha.

**Alternative flows:** Nenhum

**Use case:** Colar recurso

**ID:** 18

**Description:** O manager seleciona a opção colar recurso

**Main actor:** Manager

**Secondary actor:** Nenhum

**Pré-conditions:** Existir pelo menos um recurso na base de dados do sistema, selecionar um recurso.

**Main flow:**

1. Use case começa quando o manager seleciona a opção colar recurso.
2. Include (procurar dados no sistema dos recursos existentes).
3. If se na nossa pilha já existir um recurso, independentemente do recurso selecionado o recurso que será duplicado é o contido na pilha
4. Else caso a nossa pilha esteja vazia, a opção e ignorada.

**Post conditions:** Adiciona uma copia do recurso contido na pilha na base de dados do sistema.

**Alternative flows:** Nenhum

**Use case:** Cortar recurso

**ID:** 19

**Description:** O manager seleciona a opção cortar recurso

**Main actor:** Manager

**Secondary actor:** Nenhum

**Pré-conditions:** Existir pelo menos um recurso na base de dados do sistema e selecionar esse mesmo recurso.

**Main flow:**

1. Use case começa quando o manager seleciona a cortar recurso.
2. Include (procurar dados no sistema dos recursos existentes).
3. O recurso e eliminado da barra de recursos do diagrama de recursos.

**Post conditions:** Os dados do recurso são eliminados da base de dados do sistema.

**Alternative flows:** Nenhum

**Use case:** Para cima

**ID:** 20

**Description:** O manager seleciona a opção para cima ou 

**Main actor:** Manager

**Secondary actor:** Nenhum

**Pré-conditions:** Existir pelo menos dois recursos na base de dados do sistema, selecionar esse mesmo recurso que se pretende deslocar e ainda este recurso a mover para cima tem de ter pelo menos um recurso acima dele.

**Main flow:**

1. Use case começa quando o manager seleciona a para cima.
2. Include (procurar dados no sistema dos recursos existentes).
3. O recurso é movido uma casa para cima na barra de recursos do diagrama de recursos.

**Post conditions:** O recurso é deslocado uma casa para cima, na barra de recursos do diagrama de recursos.

**Alternative flows:** Nenhum

**Use case:** Para baixo

**ID:** 21

**Description:** O manager seleciona a opção para baixo ou 

**Main actor:** Manager

**Secondary actor:** Nenhum

**Pré-conditions:** Existir pelo menos dois recursos na base de dados do sistema, selecionar esse mesmo recurso que se pretende deslocar e ainda este recurso a mover para baixo tem de ter pelo menos um recurso abaixo dele.

**Main flow:**

1. Use case começa quando o manager seleciona a para baixo.
2. Include (procurar dados no sistema dos recursos existentes).
3. O recurso é movido uma casa para baixo na barra de recursos do diagrama de recursos.

**Post conditions:** O recurso é deslocado uma casa para baixo, na barra de recursos do diagrama de recursos.

**Alternative flows:** Nenhum

**Use case:** Ordenar por ordem lexicográfica

**ID:** 22

**Description:** O manager seleciona no diagrama de recursos o nome dos recursos.

**Main actor:** Manager

**Secondary actor:** Nenhum

**Pré-conditions:** Existir pelo menos dois recursos na base de dados do sistema, para que a ordenação possa ser efetuada.

**Main flow:**

1. Use case começa quando o manager seleciona no diagrama de recursos o nome dos recursos.
2. Include (procurar dados no sistema dos recursos existentes).
3. If se selecionar o nome seta para cima a ordenação é feita por ordem alfabética do nome do recurso e da sua função.
4. Else se selecionar o nome seta para baixo a ordenação é feita por ordem alfabética inversa tendo em conta o nome do recurso e da sua função.

**Post conditions:** Ordena os recursos contido na base de dados do sistema.

**Alternative flows:** Nenhum

User Stories

User Stories relativos ao recurso:

User Story 1:

Como manager, pretendo adicionar um recurso, para que possa adicionar um novo recurso a base de dados do sistema.

User Story 2:

Como manager, pretendo consultar as propriedades de um recurso existente na base de dados, para que possa aceder a sua informação.

User Story 3:

Como manager, pretendo alterar as propriedades de um recurso existente, para que possa manter atualizada informação relativa a esse recurso na base de dados do sistema.

User Story 4:

Como manager, pretendo enviar uma mensagem de correio eletrónico a um recurso, para que possa sempre que necessário comunicar diretamente com esse recurso.

User Story 5:

Como manager, pretendo consultar os dados de um projeto, para que possa aceder e editar a sua informação.

User Story 6:

Como manager, pretendo criar projetos, para que os possa adicionar ao sistema.

User Story 7:

Como manager, pretendo abrir projetos, para que os possa visualizar e editar.

User Story 8:

Como manager, pretendo guardar projetos, para que os possa usar recorrentemente.

User Story 9:

Como manager, pretendo criar tarefas, para que os possa adicionar ao sistema.

User Story 10:

Como manager, pretendo consultar as propriedades de um objeto existente na base de dados, para que possa aceder à sua informação.

User Story 11:

Como manager, pretendo copiar objetos existentes, para que possa fazer clones ou simplificar o ato de criar vários objetos parecidos.

User Story 12:

Como manager, pretendo colar objetos copiados, de modo a fazer clones ou simplificar o ato de criar vários objetos parecidos.

User Story 13:

Como manager, pretendo cortar objetos copiados, de modo a simplificar o ato de mover o objeto.

User Story 14:

Como manager, pretendo apagar objetos, para que possa apagar qualquer erro feito durante a criação.

User Story 15:

Como manager, pretendo eliminar um recurso, para que consiga eliminar o recurso da base de dados do sistema que já não faz parte da equipa.

User Story 16:

Como manager, pretendo copiar um recurso, para colocar numa espécie de pilha um objeto recurso igual a um já existente.

User Story 17:

Como manager, pretendo colar um recurso perviamente inserido numa espécie de pilha, para que possa duplicar um recurso já existente.

User Story 18:

Como manager, pretendo cortar do diagrama de recursos um recurso já existente, para que possa retirar da base de dados do sistema, um recurso já sem utilidade.

User Story 19:

Como manager, pretendo deslocar para cima um recurso previamente selecionado, para que possa ordenar livremente os recursos já existentes.

User Story 20:

Como manager, pretendo deslocar para baixo um recurso previamente selecionado, para que possa ordenar livremente os recursos já existentes.

User Story 21:

Como manager, pretendo ordenar alfabeticamente o nome dos recursos existentes na base de dados do sistema, para que possa manter os recursos existentes ordenados.

User Story 22:

Como manager, pretendo ordenar alfabeticamente a função desempenhada pelos recursos existentes na base de dados do sistema, para que possa manter os recursos existentes ordenados.

Dependency Metrics

Class Metrics:

Cyclic

Calcula o número de classes ou interfaces das quais cada classe depende direta ou indiretamente e que, por sua vez, dependem direta ou indiretamente dela.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.1 Histograma das dependências cíclicas (Cyclic) contidas no projeto.

Neste projeto podemos ver tal como sugere o histograma acima, que existe cerca de 450 classes com um elevado número de dependências cíclicas cerca de 500 a 512, em contraste temos também 400 classes que contem apenas entre 1 e 5 dependências cíclicas ou até nenhuma dependência cíclica.

A media de cíclica neste projeto é de 270.03 devido a termos classes com elevado número de dependências cíclicas e noutras um número muito reduzido.

Uma imagem com texto

Descrição gerada automaticamente

Fig.2 Classes exemplo com dependências cíclicas.

Essas dependências cíclicas podem gerar códigos difíceis de entender e testar, alem de na maior parte destas classes com um elevado número de dependências cíclicas, podermos verificar code smells como Larges Classes, Divergent Class, Feature Envy, Inapropriate Intimacy e Shotgun Surgery.

Dcy

Calcula o número de classes ou interfaces das quais cada classe depende diretamente.

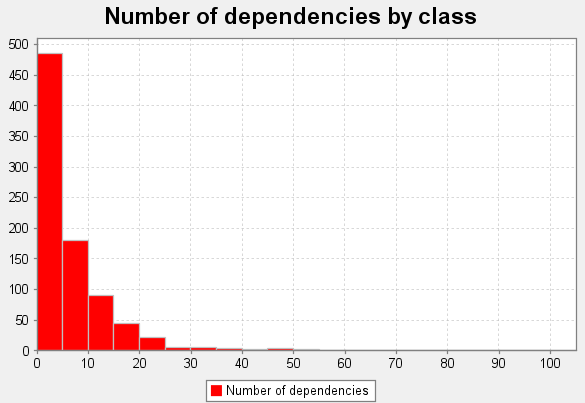


Fig.3 Histograma do número de dependências diretas de cada classe de outras classes ou interfaces (Dcy) contidas no projeto.

Neste histograma podemos verificar tal como sugere a figura acima, que existe um elevado número de classes sem dependências diretas, temos cerca de 175 classes com o número de dependências entre 5 a 10, acima de 10 dependências temos cerca de 80 classes, acima de 20 dependências o numero de classes já se reduz para cerca de 25, e acima de 25 dependências já só temos cerca de 10 classes. Sendo a média das dependências 7.06 uma vez que a maior parte das classes tem um número reduzido de dependências e as que apresentam um número mais elevado são poucas.

Uma imagem com texto

Descrição gerada automaticamente

Fig.4 Classes exemplo com dependências diretas.

Nestas classes com dependências diretas podemos verificar code smells como Larges Classes, Divergent Class, Data Class e Shotgun Surgery.

Dcy\*

Mede o número de classes ou interfaces das quais cada classe depende direta ou indiretamente.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.5 Histograma do número das dependências transitivas diretas ou indiretas (Dcy\*) contidas no projeto.

No histograma acima constatamos que cerca de 260 classes têm entre 0 e 40 dependências transitivas diretas e indiretas, cerca de 10 classes tem entre 41 a 80, e por fim cerca de 575 classes apresentam um numero de dependências entre 850 a 886, sendo este valor bastante elevado. Assim sendo resulta uma média de 587.43 dependências transitivas diretas e indiretas.

Uma imagem com texto

Descrição gerada automaticamente

Fig.6 Classes exemplo com dependências transitivas diretas e indiretas.

Nestas classes com dependências diretas podemos verificar code smells como Larges Classes, Divergent Class, Inappropriate Intimacy e Shotgun Surgery.

Dpt

Calcula o número de classes ou interfaces que dependem diretamente de cada classe.

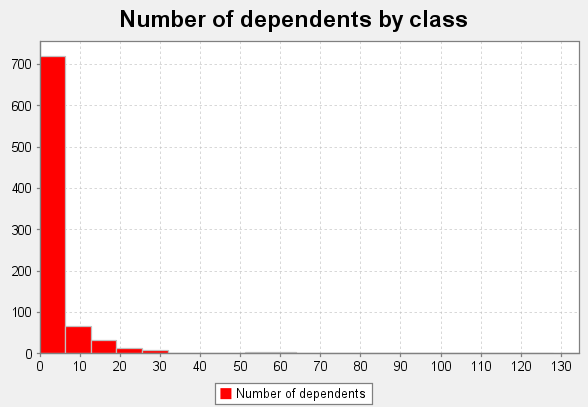


Fig.7 Histograma do número de dependentes por classe (Dpt) contidas no projeto.

No histograma acima podemos verificar que cerca de 730 classes e interfaces possuem uma dependência diretamente de outra classe entre 0 e 5, cerca de 70 classes e interfaces dependem entre 6 a 12 classes, aproximadamente 30 classes e interfaces dependem de cerca de 13 a 18 classes, à volta de 10 classes e interfaces dependem de 19 até 24 classes, e aproximadamente 5 classes e interfaces dependem de cerca de 25 a 32 classes. A média é de 4.23 uma vez que a maior parte das classes e interfaces tem um número reduzido de dependentes por classe.

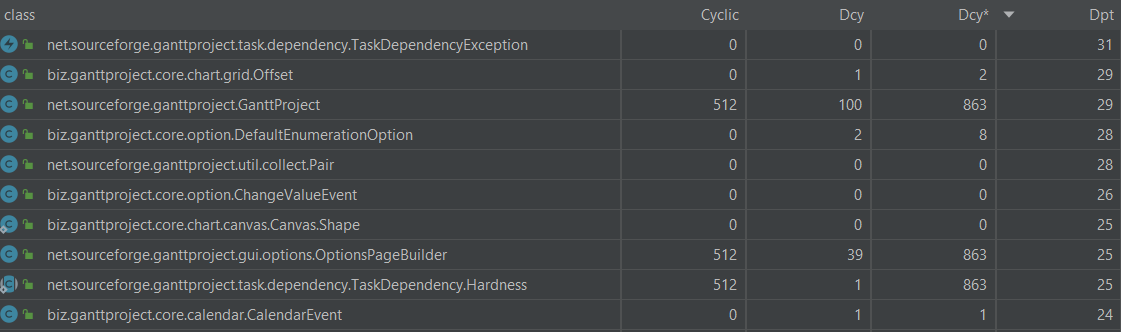


Fig.8 Classes exemplo com número de dependentes por classe.

Nestas classes com dependentes podemos verificar code smells como Refused Request e Inappropriate Intimacy.

Dpt\*

Calcula o número de classes ou interfaces que dependem direta ou indiretamente de cada classe.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.9 Histograma do número de classes ou interfaces que dependem direta ou indiretamente de cada classe (Dpt\*) contidas no projeto.

No histograma acima podemos verificar que cerca de 160 classes e interfaces dependem direta ou indiretamente de 0 a 60 classes, 640 classes e interfaces dependem de cerca de 620 a 657 classes, e cerca de 40 dependem de 660 a 673 classes. Sendo a média de 520.45 uma vez que um elevado número de classes e interfaces depende direta ou indiretamente de outras classes contidas no projeto.

Uma imagem com texto, interior, captura de ecrã

Descrição gerada automaticamente

Fig.10 Classes exemplo com número de classes que dependem direta ou indiretamente de cada classe.

Nestas classes com dependentes podemos verificar code smells como Refused Request e Inappropriate Intimacy.

PDcy

Calcula o número de packages dos quais cada classe depende direta ou indiretamente

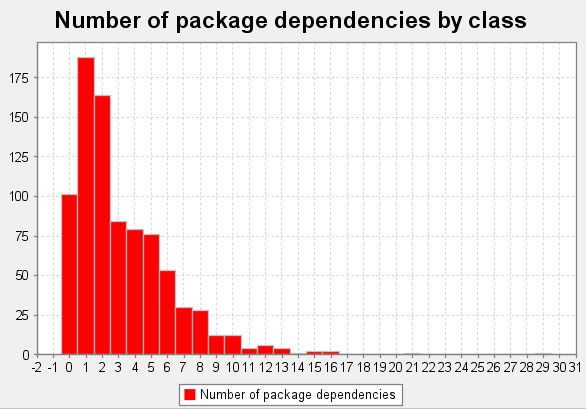


Fig.11 Histograma do número packages dos quais cada classe depende direta ou indiretamente (DPcy) contidas no projeto.

No histograma acima podemos verificar que cerca de 100 classes não dependem direta ou indiretamente de packages, cerca de 182 classes dependem de 1 package, aproximadamente 160 classes dependem de 2 packages, por volta de 80 classes dependem de 3 packages, outras 76 classes dependem de 5 packages, este número reduz se drasticamente quando a dependência das classes, por packages sobe para 9 até 16. Assim verificamos uma média de dependência direta ou indireta de cada classe por packages de aproximadamente 3.23.

Uma imagem com texto, interior

Descrição gerada automaticamente

Fig.12 Classes exemplo relativas ao número packages das quais cada classe depende direta ou indiretamente.

PDpt

Calcula o número de packages que dependem direta ou indiretamente de cada classe.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.13 Histograma do número packages que depende direta ou indiretamente de cada classe (DPpt) contidas no projeto.

No histograma apresentado acima podemos verificar que aproximadamente 80 packages não dependem direta ou indiretamente de nenhuma classe, 450 packages dependem de uma classe, cerca de 125 packages contidos neste projeto dependem direta ou indiretamente de duas classes, havendo um decréscimo do número de packages dependentes direta ou indiretamente de um maior número de classes.

Assim se verifica uma média de 2.23 ou seja cada package contido neste projeto depende direta ou indiretamente em média de cerca de 2 classes.

Uma imagem com texto, computador, eletrónica

Descrição gerada automaticamente

Fig.14 Classes exemplo com elevado número packages que depende direta ou indiretamente de cada uma delas (DPpt).

Interface Metrics:

Cyclic

Calcula o número de classes ou interfaces das quais cada interface depende direta ou indiretamente e que, por sua vez, dependem direta ou indiretamente dela.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.15 Histograma do número classes ou interfaces das quais cada interface depende direta ou indiretamente (Cyclic) contidas no projeto.

Analisando o histograma verificamos que cerca de 112 interfaces contidas no projeto dependem direta ou indiretamente de 0 a 25 classes ou interfaces e estas por sua vez dependem direta ou indiretamente dessas mesmas interfaces, em contraste vemos que aproximadamente 65 interfaces dependem de 488 a 512 classes ou interfaces e estas por sua vez dessas mesmas interfaces. Gerando uma média de 188.88 ou seja cada interface em média contida neste projeto depende direta ou indiretamente de cerca de 189 classes ou interfaces e estas dessas mesmas interfaces.

Uma imagem com texto

Descrição gerada automaticamente

Fig.16 Exemplo de algumas interfaces contidas no projeto com elevado número de dependências diretas ou indiretas de outras classes ou interfaces.

Essas dependências cíclicas podem gerar códigos difíceis de entender e testar, alem de podermos observar em muitas das interfaces com elevado número de dependências cíclicas presentes neste projeto code smells como Speculative Generality, Refused Request.

Dcy

Calcula o número de classes ou interfaces das quais cada interface depende diretamente.

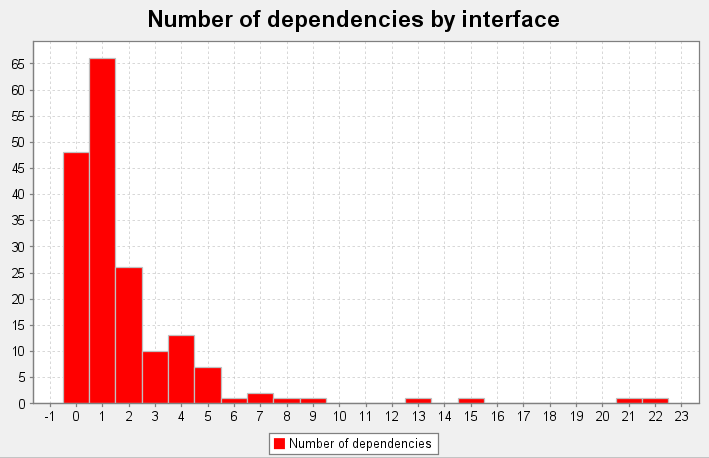


Fig.17 Histograma das dependências diretas das interfaces contidas no projeto para com outras classes ou interfaces (Dcy).

Ao observarmos o histograma acima verificamos que aproximadamente 47 interfaces contidas neste projeto não dependem diretamente de nenhuma classe ou interface, cerca de 66 interfaces dependem diretamente de 1 classe ou interface, perto de 26 interfaces dependem diretamente de 2 classes ou interfaces, e um número reduzido de interfaces depende diretamente de 6 ou mais classes ou interfaces. Sendo a média de dependência direta de cada interface por classes ou outras interfaces de 1.92, ou seja, cada interface contida no projeto depende diretamente de aproximadamente de 2 classes ou interfaces, sendo esta média um valor bastante razoável dada a dimensão do projeto em questão.

Uma imagem com texto

Descrição gerada automaticamente

Fig.18 Exemplo de algumas interfaces contidas no projeto que apresentam valores muito discrepantes da média de dependências diretas de outras classes ou interfaces.

Dcy\*

Mede o número de classes ou interfaces das quais cada interface depende direta ou indiretamente.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.19 Histograma das dependências diretas ou indiretas das interfaces contidas no projeto para com outras classes ou interfaces (Dcy\*).

Neste histograma podemos verificar uma discrepância elevada nas duas barras apresentadas, aproximadamente 110 interfaces dependem direta ou indiretamente de entre 0 a 47 classes ou interfaces, e cerca de 68 interfaces dependem direta ou indiretamente de entre 825 a 875 classes ou interfaces. Assim sendo a média de dependências diretas ou indiretas de cada interface contida neste projeto é de 334.82, sendo este valor bastante elevado o que nos sugere que apesar da média de dependências diretas apresentar um valor bastante aceitável existe um elevado número de dependências indiretas, o que fundamenta a existência dos code smells encontrados em algumas das interfaces, sendo estes Speculative Generality, Refused Request.

Uma imagem com texto

Descrição gerada automaticamente

Fig.20 Interfaces contidas no projeto com um elevado número de dependências diretas ou indiretas de outras classes ou interfaces.

Dpt

Calcula o número de classes ou interfaces que dependem diretamente de cada interface.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.21 Histograma das interfaces das quais dependem diretamente classes ou outras interfaces (Dpt).

Ao analisarmos o histograma verificamos que cerca de 128 interfaces não apresentam dependências diretas de outras classes ou interfaces ou no máximo possuem até 10 classes ou interfaces a dependerem diretamente das mesmas. Aproximadamente 22 interfaces tendem a ter entre 13 a 23 classes ou interfaces a dependerem diretamente delas, verificando-se um decréscimo acentuado do numero de interfaces das quais dependem muitas classes ou interfaces, ainda assim temos uma interfaces das quais dependem diretamente aproximadamente 224 interfaces ou classes. Resultando destes dados uma média de 14.54, ou seja, em média de cada interface dependem diretamente cerca de 15 classes ou interfaces.

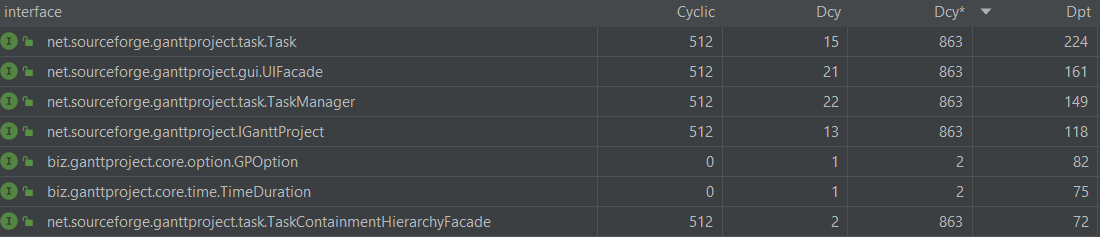


Fig.22 Exemplo das interfaces das quais dependem diretamente um maior numero de classes ou outras interfaces (Dpt).

Dpt\*

Calcula o número de classes ou interfaces que dependem direta ou indiretamente de cada interface.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.23 Histograma das interfaces das quais dependem diretamente ou indiretamente classes ou outras interfaces (Dpt\*).

No histograma acima verificamos que cerca de 10 interfaces contem entre 0 a 40 classes ou interfaces a dependerem diretamente ou indiretamente das mesmas, em oposição temos cerca de 150 interfaces das quais dependem direta ou indiretamente entre 610 a 660 interfaces ou classes, observamos ainda que cerca de 5 interfaces possuem ainda um valor exorbitante de cerca de 680 a 725 classes ou interfaces a dependerem direta ou indiretamente das mesmas. Assim sendo verifica-se uma média de 608.61, ou seja, cada interface possui em média cerca de 609 interfaces ou classes da dependerem dela.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.24 Exemplo das interfaces das quais dependem diretamente ou indiretamente um maior número de classes ou outras interfaces (Dpt\*).

PDcy

Calcula o número de packages dos quais cada interface depende direta ou indiretamente

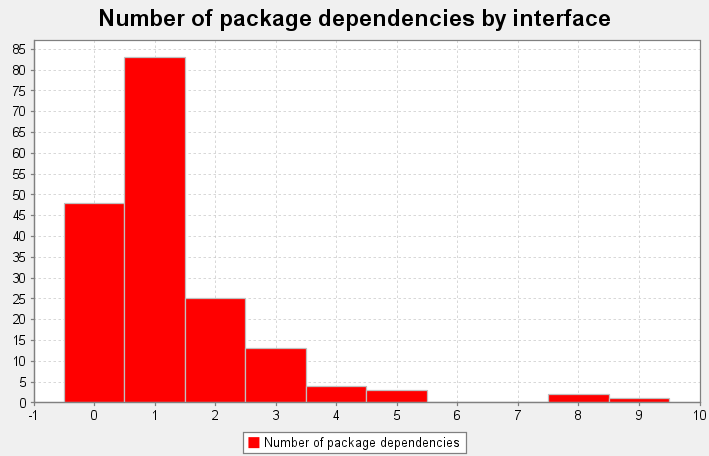


Fig.25 Histograma do número packages das quais cada interface depende direta ou indiretamente (DPpt).

Ao analisarmos o histograma acima verificamos que cerca de 47 interfaces não dependem direta ou indiretamente de nenhum package. Aproximadamente 83 interfaces dependem direta ou indiretamente de um package, havendo um decréscimo acentuado do número de interfaces que dependem de 2 ou mais packages tal como ilustrado acima. Assim sendo verificamos uma média de 1.27, ou seja, em média cada interface apenas depende direta ou indiretamente de um package. Este valor e bastante satisfatório indicando uma boa separação e organização das interfaces pelos packages contidas no projeto.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.26 Interfaces que apresentam maior número de dependências diretas ou indireta de packages.

PDpt

Calcula o número de packages que dependem direta ou indiretamente de cada interface.

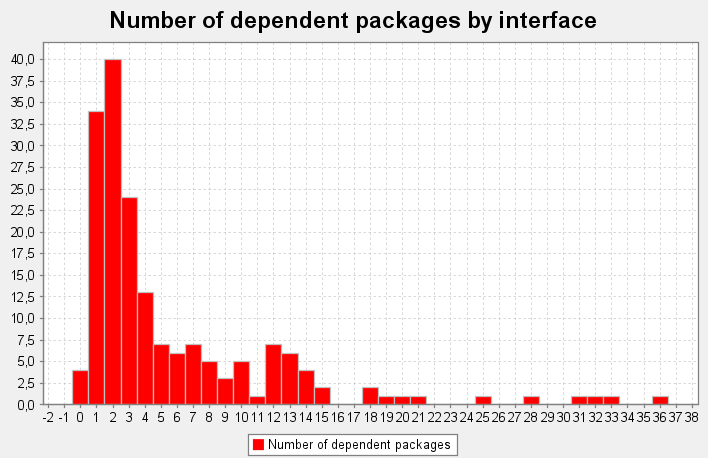


Fig.25 Histograma do número de interfaces das quais dependem direta ou indiretamente packages contidos no projeto (DPpt).

Examinando o histograma acima verificamos que cerca de 3 interfaces não contem qualquer package a depender destas direta ou indiretamente, a volta de 33 interfaces possuem 1 package a depender delas, havendo uma subida do número de interfaces para 40 das quais dependem direta ou indiretamente 2 package, havendo de seguida um decréscimo que se vai verificando no número de interfaces das quais dependem um maior número de packages. Sendo a média total de 5.67, ou seja, em média cada interface possui cerca de 6 packages a dependerem dela direta ou indiretamente.

Uma imagem com texto, computador

Descrição gerada automaticamente

Fig.28 Interfaces que apresentam maior número de packages a dependerem direta ou indiretamente das mesmas.

Packages Metrics:

Cyclic

Calcula o número de packages dos quais cada package depende direta ou indiretamente e que, por sua vez, dependem direta ou indiretamente deles.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.29 Número de packages dos quais cada package depende direta ou indiretamente, e estes por sua vez dependem também eles direta ou indiretamente deles.

No histograma acima verificamos que cerca de 42 packages ou não possuem outros packages a depender direta ou indiretamente deles, e vice versa ou têm apenas cerca de 2 packages a dependerem direta e indiretamente deles e vice versa. Em contraste possuímos cerca de 50 packages dos quais dependem direta ou indiretamente entre 46 a 49 packages e vice versa. De acordo com os dados presentes no histograma obtemos uma média de 26.74, ou seja, em média cada package tem cerca de 27 packages a dependerem diretamente ou indiretamente dele e vice versa, o que nos permite concluir uma má estruturação do projeto, assim sendo este trona-se difícil de entender e testar.

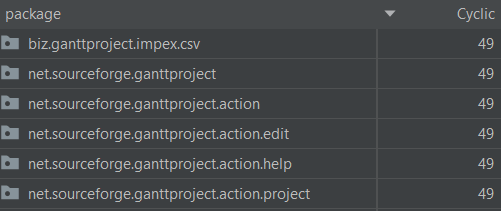


Fig.30 Packages com maior número de packages a dependerem direta ou indiretamente deles vice versa.

PDcy

Calcula o número de packages dos quais cada package depende diretamente

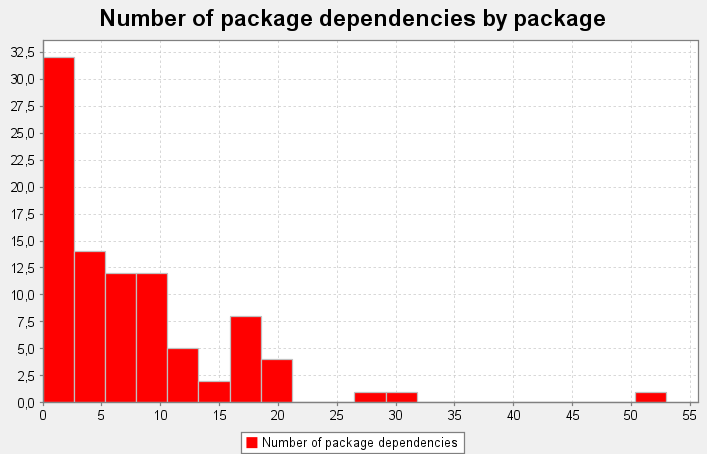


Fig.31 Número de packages dos quais cada package depende diretamente.

Neste histograma podemos verificar cerca de 32 packages possuem entre 0 a 3 packages a dependerem diretamente deles, aproximadamente 14 packages possuem 3 packages a 6 a dependerem diretamente deles, á volta de 11 packages têm entre 6 a 11packages a dependerem diretamente deles, de seguida verifica-se um decréscimo acentuado do número de packages a terem mais de 12 packages a dependerem diretamente deles. Assim verificamos uma média de 7.45, ou seja, de cada package em média dependem diretamente outos 7 packages, o que mais uma vez nos indica uma má estruturação do projeto e pouca independência entre as suas componentes, tornando-se assim o código difícil de entender e restruturar.

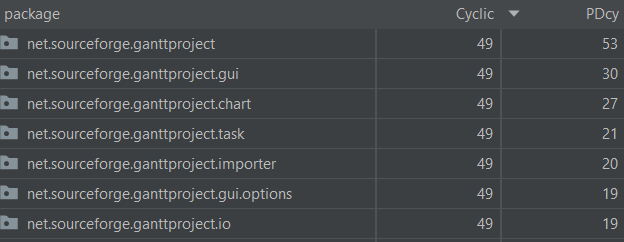


Fig.32 Packages com maior número de packages a depende diretamente deles.

PDpt

Calcula o número de packages que contêm dependências diretas a outros packages.

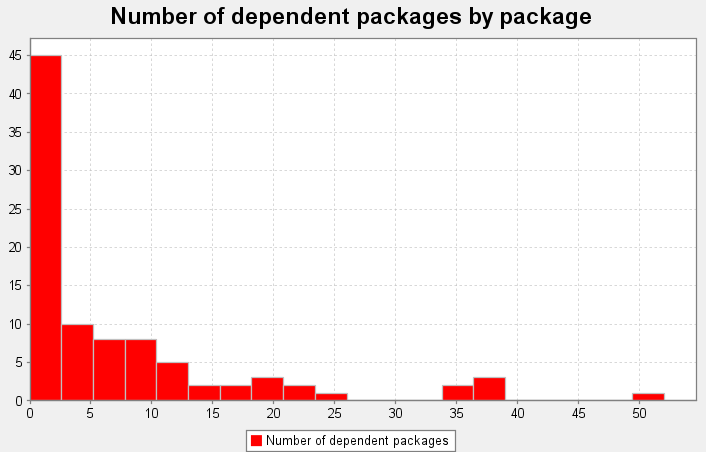


Fig.33 Número de packages que contêm dependências diretas a outros packages.

Ao analisarmos o histograma verificamos que cerca de 45 packages não contêm dependências diretas ou apenas contêm dependências diretas a aproximadamente 2 packages, de seguida verifica-se um decréscimo abrupto, cerca de 10 packages possuem dependências diretas a cerca de outros 4 a 5 packages, verificando-se ainda que cerca de 3 packages têm dependências diretas a outros 39 a 52 packages. Deste dados resulta uma média de 7.45, ou seja, cada package possui em média a depender diretamente dele outros 7 packages.

Verificamos que a média de PDpt número de packages que contêm dependências diretas a outros packages é a mesma que a média de PDcy calcula o número de packages dos quais cada package depende diretamente, o que faz todo o sentido e o que mais uma vez nos indica que existe uma má estruturação do projeto e pouca independência entre as suas componentes, tornando-se assim o código difícil de entender e restruturar.

Uma imagem com mesa

Descrição gerada automaticamente

Fig.34 Packages com maior número de dependência direta a outros packages.

PDpt\*

Calcula o número de packages que contêm dependências diretas ou indiretas a outros packages.

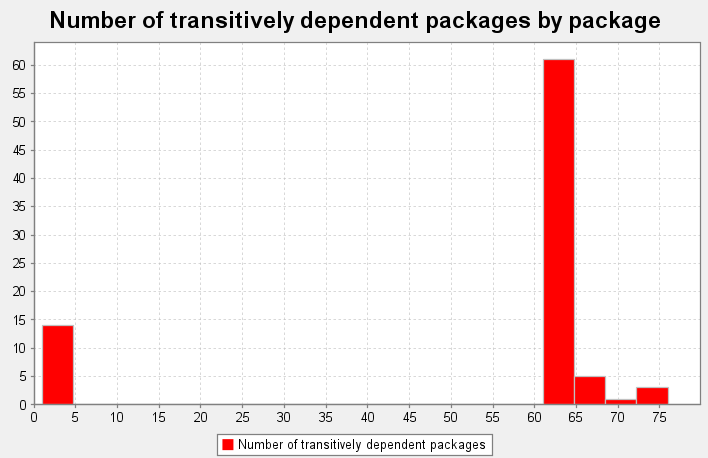


Fig.35 Número de packages que contêm dependências diretas ou indiretas a outros packages.

Neste histograma podemos verificar que aproximadamente 14 packages dependem direta ou indiretamente de 1 a 4 packages, havendo um crescimento abrupto de cerca de 62 packages a dependerem direta ou indiretamente de entre 61 a 64 packages, e visível também que a volta de 7 packages dependem direta ou indiretamente de entre 72 a 76 packages. Verificando-se assim uma média de 52.88, assim sendo em média cada package depende direta ou indiretamente de aproximadamente 53 packages, havendo assim um crescimento abrupto relativamente a média das dependências diretas.

Deste modo a média referida sugere mais uma vez haver de facto uma falta de estruturação do projeto e independência para com os componentes do mesmo, havendo uma lacuna na factorização do código.

Uma imagem com texto, interior

Descrição gerada automaticamente

Fig.36 Packages com maior número de dependências diretas ou indiretas a outros packages.

Chidamber and Kemerer Metrics

WMC Weighted Methods Per Class

WMC é simplesmente o número de métodos por classe. Quanto mais baixo for, mais provável é de termos uma classe que pode ser reutilizada e menor o tempo e recursos usados na sua manutenção e desenvolvimento.

DIT Depth of Inheritance Tree

DIT é o número máximo de classes herdadas da classe até à root. Quanto mais funda uma classe estiver na hierarquia, mais variáveis e métodos é provável que tenha herdado, tornando-a mais complexa. Árvores de grande profundidade têm portanto maior complexidade estrutural, mas por outro lado, também promovem a reutilização de classes.

O DIT recomendado é normalmente <= 5.

NOC Number of Children

NOC é o número de subclasses imediatas que são derivadas de uma classe. O NOC e o DIT estão fortemente interligados, sendo um DIT maior normalmente melhor que o NOC, ou seja, altura é melhor para a complexidade do que largura nas árvores de herança.

Um NOC alto pode indicar:

* Reutilização alta de uma classe base.
* Classe base pode necessitar de mais testes (pois tem um maior número de dependências)
* Uma má abstração da classe parente
* Um mau uso de subclasses. Neste caso é possível ser necessário agrupar e introduzir um novo nível de herança a estas.

Por outro lado, um NOC maior normalmente resulta em menos falhas, possivelmente por ter alta reutilização de poucas classes.

Uma classe com NOC e WMC altos indica uma complexidade elevada no topo da árvore hierárquica, com a classe a influenciar um número alto de classes filhas.

CBO Coupling between Object Classes

CBO é o número de classes às quais uma classe está associada, ou seja, o número de classes das quais uma usa métodos ou variáveis de instância definidas pela outra.

São contados apenas os métodos chamados e referências de variáveis usadas.

Um CBO alto não é útil, quanto mais independente uma classe é, mais fácil é de ser reutilizada. Igualmente não ajuda com a manutenção ou desenvolvimento, já que qualquer mudança na classe pode levar a um maior número de classes afetadas.

O CBO recomendado é <=14.

RFC and RFC’ Response for a Class

Um CBO alto não é útil, quanto mais independente uma classe é, mais fácil é de ser reutilizada. Igualmente não ajuda com a manutenção ou desenvolvimento, já que qualquer mudança na classe pode levar a um maior número de classes afetadas.

MOOD Metrics

**Attribute Hiding Factor (AHF)**

O fator de ocultação dos Atributos mede como as variáveis são encapsulados em uma classe. A visibilidade é contada em relação a outras classes. AHF representa a média de ocultação entre todas as classes do sistema. Um Atributo privado está totalmente oculto.

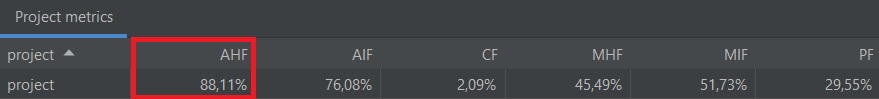
Este fator pode ser calculado pela seguinte fórmula:

Uma imagem com texto

Descrição gerada automaticamente**Ah(Ci) =** Atributos ocultos na classe Ci

**Ad(Ci)** = Av(Ci) + Ah(Ci): Atributos definidos em Ci

**Av(Ci):** Atributos visíveis na classe Ci

**TC:** Número total de Classes.

Uma imagem com mesa

Descrição gerada automaticamente

Geralmente, um valor alto de AHF é aconselhado, na medida em que os atributos de uma classe devem ser ocultos de outras classes, de modo que 100% é o valor ideal do AHF.

Em análise ao nosso programa, como temos um valor de 88,1%, podemos concluir que a maior parte dos nossos atributos são ocultos para outras classes, sendo utilizados apenas nas classes onde foram criados, e com isto temos uma menor complexidade do nosso programa.

**Attribute Inheritance Factor (AIF)**

Em Herança, o filho ou a subclasse herda as propriedades (Atributos) da classe mãe ou da superclasse. A extensão em que esses atributos são herdados é definida pelo Fator de Herança do Atributo (AIF).

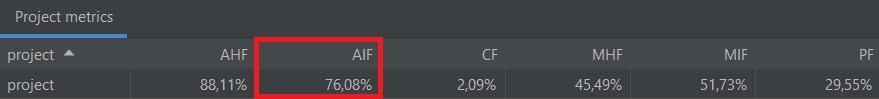
Este fator pode ser calculado pela seguinte fórmula:

Uma imagem com texto

Descrição gerada automaticamente**Ah(Ci) =** Atributos herdados

**Aa(Ci)** = Ad(Ci) + Ah(Ci): Atributos definidos em Ci

**Ad(Ci):** Atributos definidos na classe Ci

**TC:** Número total de Classes.

Uma imagem com mesa

Descrição gerada automaticamente

Geralmente, a faixa de AIF está entre 0% e 48%. De acordo com o nosso programa onde temos uma percentagem de 76,08%, podemos concluir que as classes filho herdam um grande número de atributos de sua classe mãe.

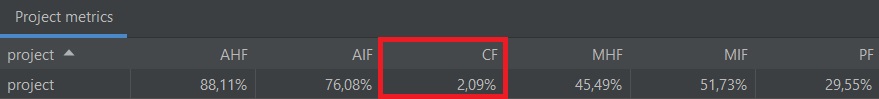
**Coupling Factor (CF)**

O Fator de Acoplamento (CF) é a razão entre o acoplamento real entre diferentes classes e o acoplamento máximo possível que pode acontecer no sistema. Se uma classe pode acessar aos atributos da segunda classe, então diz-se que a primeira classe é acoplada à segunda classe.

Este fator pode ser calculado pela seguinte fórmula:

Uma imagem com texto, quadro branco

Descrição gerada automaticamente**is\_client**(Cc,Cs) =| 1 **if** (Ci⇒Cj)^(Ci≠Cj) , **else** 0

**TC:** Número total de Classes.

Uma imagem com mesa

Descrição gerada automaticamente

O alto valor de CF mostra que as classes do sistema são mais interconectadas e interdependentes, no leva ao problema de que às vezes é muito difícil alterar ou reparar o sistema em caso de qualquer bug ou problema, porque a funcionalidade em que o bug está, poderia ser implementada por mais de duas classes e temos de fazer alterações em todas as classes relacionadas.

Em análise ao nosso programa o valor de CF é de apenas 2.09% , o que nos permite concluir que irá facilitar-nos em caso de bugs ou de novas implementações.

**Method Hiding Factor (MHF)**

O fator de ocultação dos Métodos mede como estes são encapsulados em uma classe. A visibilidade é contada em relação a outras classes. MHF representa a média de ocultação entre todas as classes do sistema. Um Método privado está totalmente oculto.

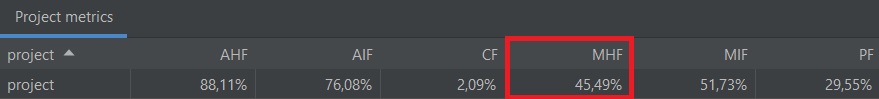
Este fator pode ser calculado pela seguinte fórmula:

**Mh(Ci) =** Métodos ocultos na classe Ci

**Uma imagem com texto

Descrição gerada automaticamenteMd(Ci)** = Mv(Ci) + Mh(Ci): Métodos definidos em Ci

**Mv(Ci):** Métodos visíveis na classe Ci

**TC:** Número total de Classes.Uma imagem com mesa

Descrição gerada automaticamente

Um MHF baixo indica uma implementação insuficientemente abstrata. Uma grande proporção de métodos está desprotegida e a probabilidade de erros é alta. Um MHF alto indica muito pouca funcionalidade. Pode também indicar que o desenho ou modelo inclui uma alta proporção de métodos especializados que não estão disponíveis para reutilização. Um valor aceitável de MHF é de 8% a 25%.

Em concordância com o nosso programa temos um valor de 45,49% de MHF, em que nos permite utilizar e reutilizar uma boa quantidade de métodos, mas que a implementação também é suficientemente abstrata, no que resulta uma boa funcionalidade do programa.

**Method Inheritance Factor (MIF)**

Em Herança, o filho ou a subclasse herda as propriedades (Métodos) da classe mãe ou da superclasse. A extensão em que esses métodos são herdados é definida pelo Fator de Herança do Método (AIF).

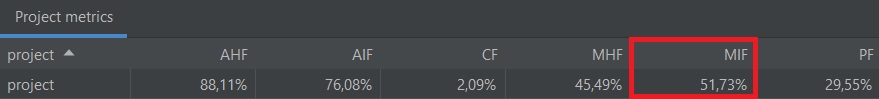
Este fator pode ser calculado pela seguinte fórmula:

**Uma imagem com texto

Descrição gerada automaticamenteMi:** Métodos herdados

**Ma(Ci)** = Md(Ci) + Mi(Ci): Métodos definidos em Ci

**Md(Ci):** Métodos definidos na classe Ci

**TC:** Número total de Classes.

À primeira vista, podemos ser tentados a pensar que a herança deve ser usada extensivamente. No entanto, a composição de várias relações de herança constrói um grafo acíclico direcionado (árvore de hierarquia de herança), cuja profundidade e largura fazem com que a compreensibilidade e a testabilidade desapareçam rapidamente. Geralmente, a faixa de MIF está entre 20% a 80%**Uma imagem com mesa

Descrição gerada automaticamente**.

De acordo com o nossos valores, temos um MIF de 51,73%, o que nos dá uma boa compreensibilidade e testabilidade do nosso programa.

**Polymorphism Factor (PF)**

No polimorfismo, a classe filho pode implementar o método de uma maneira diferente. O mesmo método pode ser implementado de maneiras diferentes na classe filho e na classe mãe. Ele é definido pela razão entre um número real de substituições do método e o número máximo de substituições totais do método.

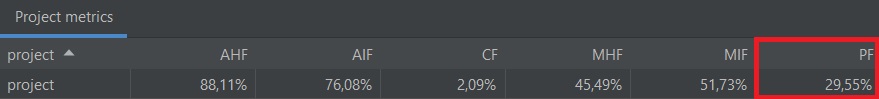
Este fator pode ser calculado pela seguinte fórmula:

**Mo(Ci):** Métodos substituídos na classe Ci

**Uma imagem com texto

Descrição gerada automaticamenteMn(Ci):** Novos Métodos em Ci

**DC(Ci):** Número de descendências da classe Ci (classes derivadas)

**TC:** Número total de Classes.

Uma imagem com mesa

Descrição gerada automaticamente

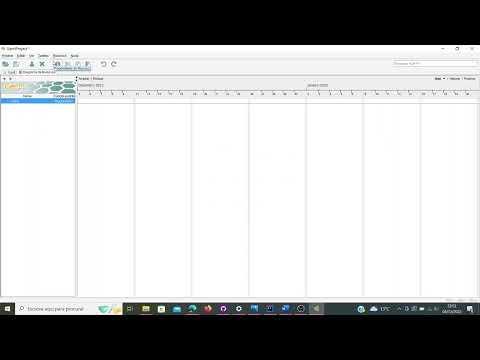
O polimorfismo surge da herança e tem seus prós e contras. Podemos intuitivamente esperar que o polimorfismo (Overrides) possa ser usado em uma extensão razoável para manter o código claro, mas esse código excessivamente polimórfico pode ser muito complexo para entender (porque vários métodos alternativos podem ser executados para uma instrução de chamada). A PF deve estar em uma faixa razoável com um limite inferior e um limite superior.

Ao analisar o PF nosso programa obtivemos um valor de 29,55%, onde podemos concluir que temos um sistema suficientemente claro e limpo com uma complexidade razoável, que nos permite uma melhor compreensão do mesmo.

Referências

1. [Service-Oriented and Cloud Computing: 7th IFIP WG 2.14 European Conference ... - Google Livros](https://books.google.pt/books?id=Rb5sDwAAQBAJ&pg=PA60&lpg=PA60&dq=dependency+metrics+Cyclic++dcy+dcy*++dpt+dpt*&source=bl&ots=zwbznsQpj0&sig=ACfU3U3DkKMgyGI8YyjmhFG1OvzoKjrYwQ&hl=pt-PT&sa=X&ved=2ahUKEwj3tpatrdf7AhUuTKQEHV90BpEQ6AF6BAgIEAM#v=onepage&q=dependency%20metrics%20Cyclic%20%20dcy%20dcy*%20%20dpt%20dpt*&f=false)
2. <https://www.aivosto.com/project/help/pm-oo-ck.html>

Link do video de apresentação das novas funcionalidades implementadas



https://youtu.be/5GlAPrGndAM

Link do repositório no GitHub

[Rick127/ganttproject at BRANCH\_2\_8\_9 (github.com)](https://github.com/Rick127/ganttproject/tree/BRANCH_2_8_9)

Divisão do trabalho

Use case diagrams:

Diagrama de recursos

Realizado por: Nádia Mendes 53175

Revisto por: João Esteves 47994

Diagrama do diagrama de recursos

Realizado por: João Esteves 47994

Revisto por:

Diagrama do gantt

Realizado por: José Morgado 59457

Revisto por:

Use case descriptions:

Use case description dos recursos

Realizado por: Nádia Mendes 53175

Revisto por: João Esteves 47994

Use case description do diagrama de recursos

Realizado por: João Esteves 47994

Revisto por:

Use case description do gantt

Realizado por: José Morgado 59457

Revisto por:

User Stories:

User stories dos recursos

Realizado por: Nádia Mendes 53175

Revisto por: João Esteves 47994

User stories do diagrama de recursos

Realizado por: João Esteves 47994

Revisto por:

User stories dos gantt

Realizado por: José Morgado 59457

Revisto por:

Programação funcionalidades extra:

Realizado por: Nádia Mendes 53175

Testes unitários das funcionalidades extra:

Realizado por: Nádia Mendes 53175

Métricas:

Dependency Metrics

Realizado por: Nádia Mendes 53175

Revisto por: João Esteves 47994

MOOD metrics

Realizado por: João Esteves 47994

Revisto por:

Complexity metrics

Realizado por: José Morgado 59457

Revisto por:

Video:

Realizado por: Nádia Mendes 53175

Relatório final:

Realizado por: Nádia Mendes 53175

Conclusão:

Com este trabalho concluímos que a documentação é uma parte fundamental para a elaboração de um projeto e para a sua reestruturação assim como para a implementação de novas funcionalidades num projeto já existente, especialmente em caso como este em que o projeto no qual tivemos de fazer alterações possuía uma dimensão consideravelmente grande.

Este projeto permitiu-nos também por em pratica os nossos conhecimentos teóricos sobre os code smells e juntamente com as métricas referidas acima, estes permitiram-nos analisar concretamente o impacto que estes podem ter sobre um projeto, o quanto o podem tornar difícil de reestruturar, adicionar novas funcionalidades, compreender e testar.