



Relatório ES

2022/2023

Grupo

Bruno Melo - 60019

Duarte Cruz - 59765

João Pereira - 60180

Neel Badracim - 60492

Ricardo Bessa - 60485

Índice

Índice	2
Introdução	4
1ª Fase	5
Objetivos	5
Constrangimentos	5
Bruno Melo	6
Design Patterns	6
Pattern 1: Memento Pattern	6
Pattern 2: Decorator Pattern	7
Pattern 3: Factory Method Pattern	8
Code Smells	9
Code Smell 1: Dead Code	9
Code Smell 2: Data Class	10
Code Smell 3: Dead Code	11
Duarte Cruz	12
Design Patterns	12
Pattern 1: Chain of Responsibility Pattern(Behavioral)	12
Pattern 2: Builder Pattern(creational)	13
Pattern 3: Iterator	14
Code Smells	15
Code Smell 1: Comments and Dead Code	15
Code Smell 2: Shotgun surgery	16
Code Smell 3: Dead code and Long parameter list	17
João Pereira	18
Design Patterns	18
Pattern 1: Decorator Pattern	18
Pattern 2: Façade Pattern	18
Pattern 3: Factory Method Pattern	19
Code Smells	20
Code Smell 1: Data Class	20
Code Smell 2: Comments and Dead Code	20
Code Smell 3: Shotgun Surgery	21
Neel Badracim	22
Design Patterns	22
Pattern 1: Proxy Pattern (Structural Design Pattern)	22
Pattern 2: Iterator	22
Pattern 3: Singleton Method (Creational)	23

Code Smells	24
Code Smell 1: Data Class	24
Code Smell 2: Dead Code	24
Code Smell 3: Comments and Speculative Generality	25
Ricardo Bessa	26
Design Patterns	26
Pattern 1: Facade Pattern (Structural Design Pattern)	26
Pattern 2: Singleton Pattern (Creational Design Pattern)	27
Pattern 3: Abstract Factory Pattern (Creational)	28
Code Smells	29
Code Smell 1: Comments	29
Code Smell 2: Dead Code	30
Code Smell 3: Comments e Speculative generality	31
2ª Fase	33
Use Case Diagrams	33
Use case 1 - Project	34
Use case 2 - Edit	36
Use case 3 - Tasks	38
Use case 4 - Resources	40
Use case 5 - Help	42
Métricas	44
Chidamber-Kemerer	45
Complexity Metrics	48
Dependency Metrics	51
Lines of Code Metrics	54
Martin Packaging Metrics	58
Potenciais problemas no código	63
Design Patterns identificadas	64
Novas implementações	65
Funcionalidade 1 - Google Drive	66
Funcionalidade 2 - Google Calendar	67
Vídeo	68
Conclusão	69
Anexos	70
Webgrafia	71
Programas	72

Introdução

A expansão da ferramenta GanttProject com duas novas funcionalidades serviu de mote para a aplicação dos conhecimentos adquiridos ao longo do semestre sobre boas práticas de engenharia de software. Com estas novas funcionalidades, o grupo propôs-se também a aprender a trabalhar com bibliotecas muito utilizadas hoje em dia, através da Google API. Desta forma, pretendemos modernizar a ferramenta apresentada enquanto praticamos a metodologia Scrum no âmbito da aplicação de um Agile mindset.

Repositório da 1ª Fase: https://github.com/MagoPT/ES2223_ganttproject

Repositório da 2ª Fase: <https://github.com/joaopereira12/ganttproject>

1ª Fase

Objetivos

O objetivo desta 1ª parte do projeto é o de identificar os design patterns discutidos em aula. Para isso, nossa identificação incluirá:

- Code snippet
- A localização exata do código
- Uma explicação do motivo pelo qual identificamos isto como uma instanciação de um design pattern.

Constrangimentos

Cada membro da equipa será responsável por identificar três design patterns diferentes. Os design patterns podem se sobrepor de um membro da equipa para outro, desde que correspondam a locais diferentes no código. Cada membro da equipa também deve rever três outros design patterns identificados pelos outros membros da equipa.

Tabela de revisão:

		Patterns			Smells		
		1	2	3	1	2	3
Membro	Bruno	Duarte	João	Neel	Ricardo	Duarte	João
	Duarte	João	Neel	Ricardo	Bruno	João	Neel
	João	Neel	Ricardo	Bruno	Duarte	Neel	Ricardo
	Neel	Ricardo	Bruno	Duarte	João	Ricardo	Bruno
	Ricardo	Bruno	Duarte	João	Neel	Bruno	Duarte

Como ler esta tabela:

- O Design Pattern 1 do Bruno foi revisto pelo Duarte
- O Code Smell 1 do Ricardo foi revisto pelo Neel

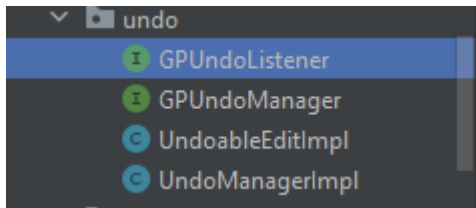
Bruno Melo

Design Patterns

Pattern 1: Memento Pattern

Localização do code pattern:

ganttproject/src/main/java/net/sourceforge/ganttproject/undo:



ganttproject/src/main/java/net/sourceforge/ganttproject/undo/UndoableEditImpl.java:

```
UndoableEditImpl.java ES2223_ganttproject/ganttproject/src/main/java/net/sourceforge/ganttproject/undo
91
92 @Override
93 public void undo() throws CannotUndoException {
94     try {
95         restoreDocument(myDocumentBefore);
96         if (projectDatabaseTxn != null) {
97             try {
98                 projectDatabaseTxn.undo();
99             } catch (ProjectDatabaseException e) {
100                 GPLogger.log(e);
101             }
102         }
103     } catch (DocumentException | IOException e) {
104         undoRedoExceptionHandler(e);
105     }
106 }
107
```

ganttproject/src/main/java/net/sourceforge/ganttproject/GanttProjectBase.java:

```
myUndoManager = new UndoManagerImpl( project: this, parserFactory: null, myDocumentManager, myProjectDatabase) {
    @Override
    protected ParserFactory getParserFactory() { return GanttProjectBase.this.getParserFactory(); }
};
```

Explicação:

Este conjunto de classes e interfaces trabalham em conjunto para dar ao usuário a possibilidade de desfazer edições no Projeto Gantt. Desfazer uma edição é o mesmo que retornar o objeto Gráfico de Gantt a um de seus estados anteriores. Esta é uma característica do Memento Pattern.

Review 1

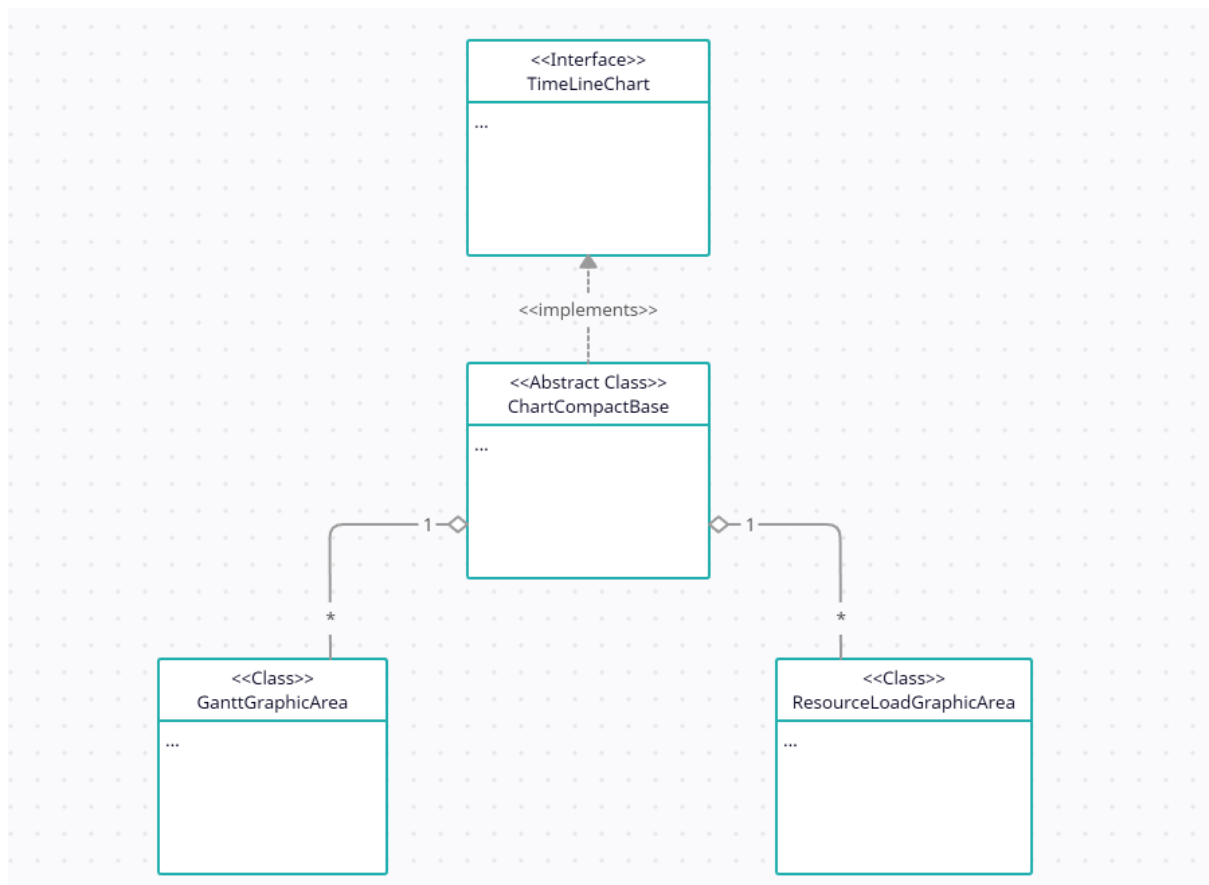
Este *Design Pattern* foi revisto e aprovado por Duarte Cruz.

Pattern 2: Decorator Pattern

Localização do pattern:

ganttproject/src/main/java/net/sourceforge/ganttproject/ChartComponentBase.java
ganttproject/src/main/java/net/sourceforge/ganttproject/chart/TimelineChart.java
ganttproject/src/main/java/net/sourceforge/ganttproject/GanttGraphicArea.java
ganttproject/src/main/java/net/sourceforge/ganttproject/ResourceLoadGraphicArea.java

Explicação:



O Decorator pattern permite que um usuário adicione novas funcionalidades a um objeto existente sem alterar sua estrutura. Esse tipo de padrão de design vem sob o padrão estrutural, pois esse padrão atua como um invólucro para a classe existente.

Esse padrão cria uma classe decoradora que envolve a classe original e fornece funcionalidade adicional mantendo a assinatura dos métodos de classe intacta.

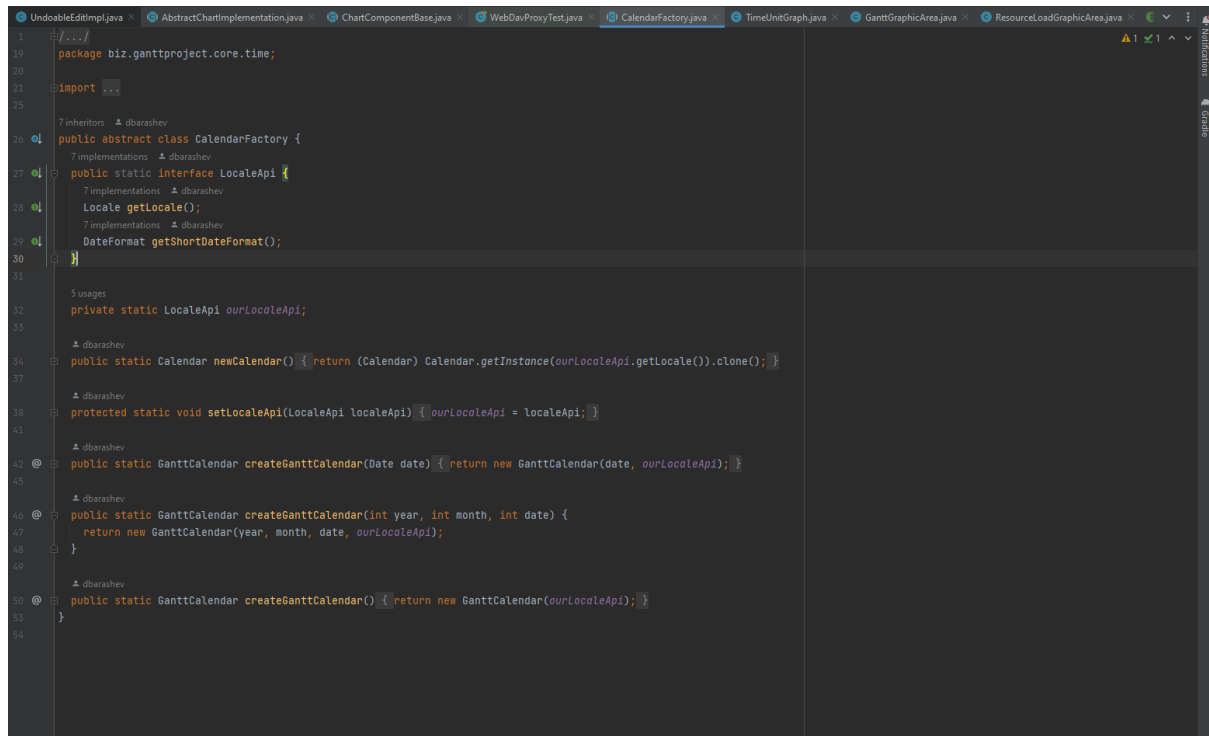
Review 2

Este *Design Pattern* foi revisto e aprovado por João Pereira.

Pattern 3: Factory Method Pattern

Localização do code pattern:

biz.ganttproject.core/src/main/java/biz/ganttproject/core/time/CalendarFactory.java

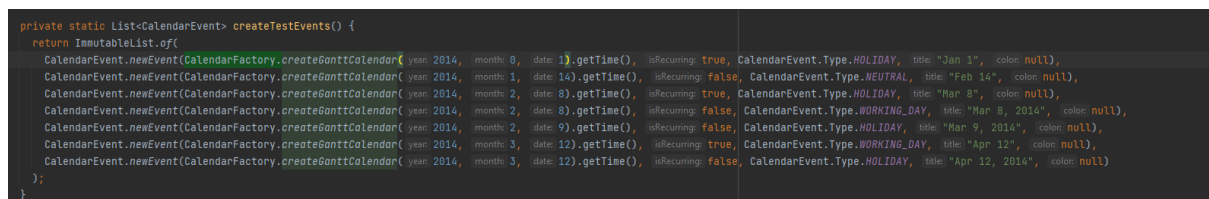


```
1 //...
19 package biz.ganttproject.core.time;
20
21 import ...
22
23 7 inheritors  ▴ dbarashev
24 public abstract class CalendarFactory {
25     7 implementations  ▴ dbarashev
26     public static interface LocaleApi {
27         7 implementations  ▴ dbarashev
28         Locale getLocale();
29         DateFormat getShortDateFormat();
30     }
31
32     5 usages
33     private static LocaleApi ourLocaleApi;
34
35     ▴ dbarashev
36     public static Calendar newCalendar() { return (Calendar) Calendar.getInstance(ourLocaleApi.getLocale()).clone(); }
37
38     ▴ dbarashev
39     protected static void setLocaleApi(LocaleApi localeApi) { ourLocaleApi = localeApi; }
40
41     ▴ dbarashev
42     public static GanttCalendar createGanttCalendar(Date date) { return new GanttCalendar(date, ourLocaleApi); }
43
44     ▴ dbarashev
45     public static GanttCalendar createGanttCalendar(int year, int month, int date) {
46         return new GanttCalendar(year, month, date, ourLocaleApi);
47     }
48
49     ▴ dbarashev
50     public static GanttCalendar createGanttCalendar() { return new GanttCalendar(ourLocaleApi); }
51 }
52
53
54
```

Explicação:

No padrão Factory, criamos objetos sem expor a lógica de criação ao cliente e nos referimos a objetos recém-criados usando uma interface comum.

Isso é exatamente o que a classe CalendarFactory faz. Ele cria objetos de calendário sem expor sua lógica de criação ao usuário.



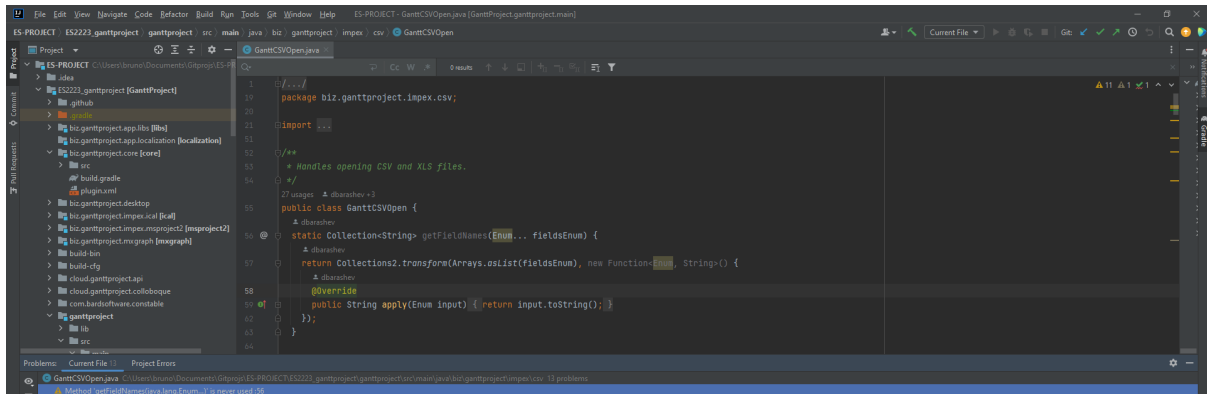
```
private static List<CalendarEvent> createTestEvents() {
    return ImmutableList.of(
        CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014, month: 0, date: 1).getTime(), isRecurring: true, CalendarEvent.Type.HOLIDAY, title: "Jan 1", color: null),
        CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014, month: 1, date: 14).getTime(), isRecurring: false, CalendarEvent.Type.NEUTRAL, title: "Feb 14", color: null),
        CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014, month: 2, date: 8).getTime(), isRecurring: true, CalendarEvent.Type.HOLIDAY, title: "Mar 8", color: null),
        CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014, month: 2, date: 8).getTime(), isRecurring: false, CalendarEvent.Type.WORKING_DAY, title: "Mar 8, 2014", color: null),
        CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014, month: 2, date: 9).getTime(), isRecurring: false, CalendarEvent.Type.HOLIDAY, title: "Mar 9, 2014", color: null),
        CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014, month: 3, date: 12).getTime(), isRecurring: true, CalendarEvent.Type.WORKING_DAY, title: "Apr 12", color: null),
        CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014, month: 3, date: 12).getTime(), isRecurring: false, CalendarEvent.Type.HOLIDAY, title: "Apr 12, 2014", color: null)
    );
}
```

Review 3

Este *Design Pattern* foi revisto e aprovado por Neel Badracim.

Code Smells

Code Smell 1: Dead Code



Localização do code smell a partir da diretoria do repository:

ganttproject/src/main/java/biz/ganttproject/impex/csv/GanttCSVOpen.java, linha 56.

Explicação:

O método `getFieldNames(Enum... fieldsEnum)` nunca é utilizado, sendo conhecido como: dead code.

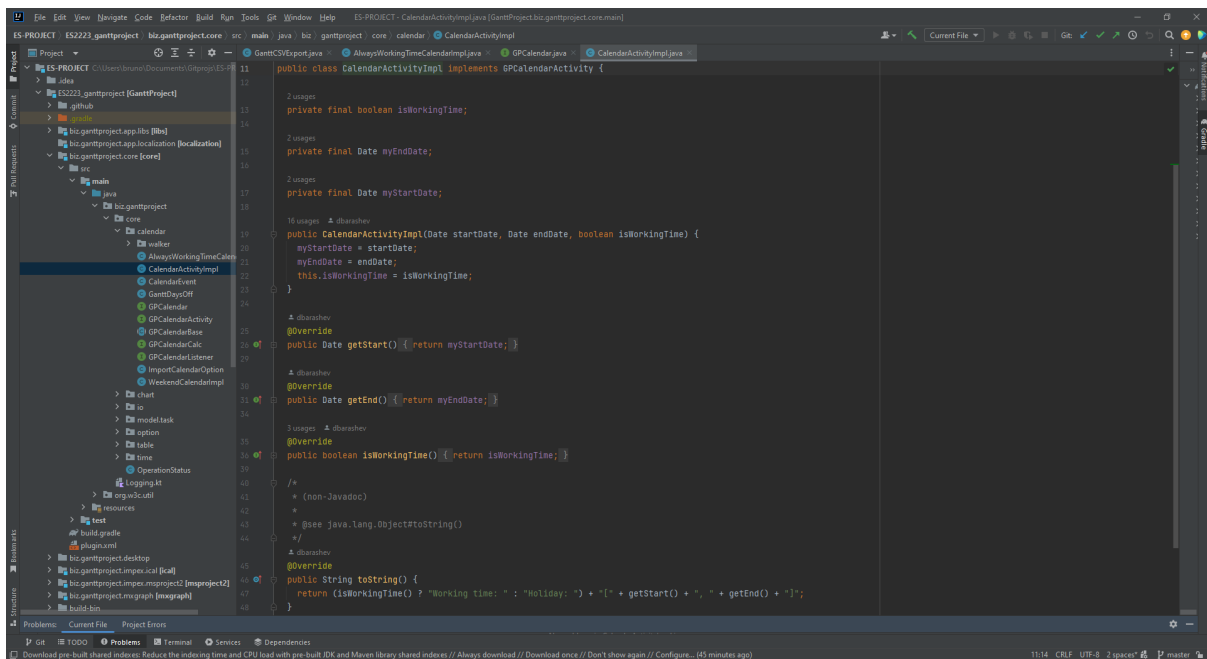
Proposta de correção:

O método deve ser apagado.

Review 1

Este *Code Smell* foi revisto e aprovado por Ricardo Bessa.

Code Smell 2: Data Class



Localização do code smell a partir da diretoria do repository:

biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/CalendarActivityImpl.java

Explanation:

A `CalendarActivityImpl.java` class é muito pequena e é apenas utilizada para guardar e devolver dados, sendo conhecida como o code smell: Data Class.

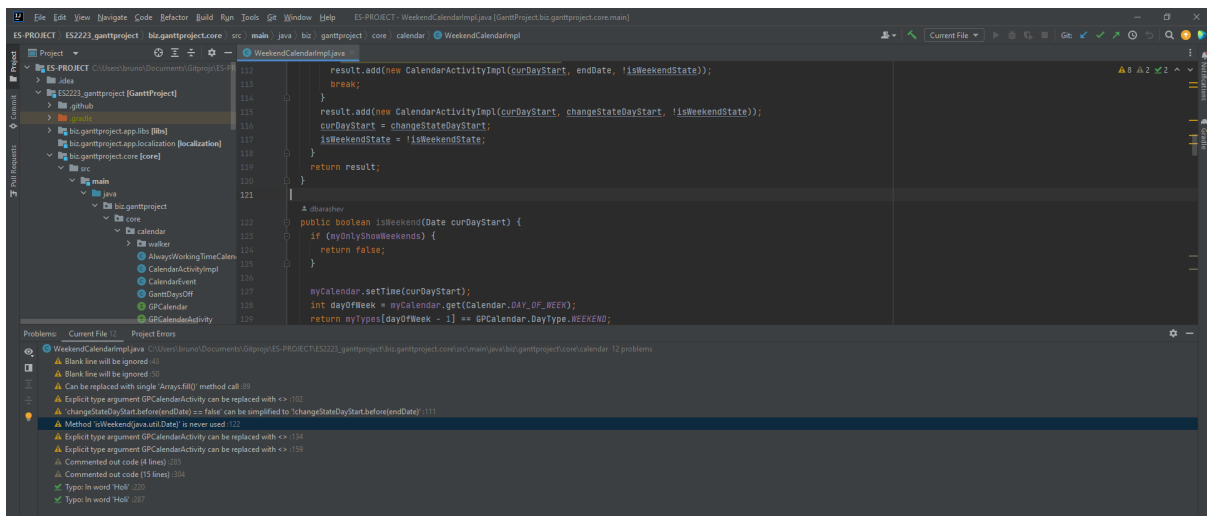
Proposta de correção:

Essa classe pode não ser necessária se sua única funcionalidade for armazenar dados. Como alternativa, algumas funcionalidades podem ser adicionadas a ela para torná-la uma classe necessária.

Review 2

Este *Code Smell* foi revisto e aprovado por Duarte Cruz.

Code Smell 3: Dead Code



Localização do code smell a partir da diretoria do repository:

`biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/WeekendCalendarImpl.java`, linha 122.

Explicação:

O método `isWeekend(Date curDayStart)` nunca é utilizado (dead code).

Proposta de correção:

Este método deveria ser apagado.

Review 3

Este *Code Smell* foi revisto e aprovado por João Pereira.

Duarte Cruz

Design Patterns

Pattern 1: Chain of Responsibility Pattern(Behavioral)

Localização:

ganttproject\src\main\java\net\sourceforge\ganttproject\ganttproject.java, linha 595 -> 604
ganttproject\src\main\java\net\sourceforge\ganttproject\ganttoprtions.java, linha 196 -> 204
java/io/FilterOutputStream.java , linha 196



Explicação:

O design pattern chain of responsibility permite ao objeto cliente enviar um pedido, o primeiro elemento na cadeia vai tentar processá-lo. Se o elemento chamado conseguir resolver o problema então o pedido acaba aqui. Caso o elemento não consiga processar o pedido então chama o próximo elemento da cadeia, este processo repete-se até o pedido ser tratado. Se nenhum elemento na cadeia não consegue resolver o pedido então o pedido não é satisfeito.

Neste exemplo, o código está a tentar sair da aplicação, mas antes de conseguir fazer isto ele têm de salvar os dados da mesma. Para isto ele chama a função `save()`, porém esta função sozinha não consegue fechar o file Writer, então chama a função `close()`. Se algum dos pedidos falhar a aplicação não vai encerrar corretamente.

Review 1

Este *Design Pattern* foi revisto e aprovado por João Pereira.

Pattern 2: Builder Pattern(creational)

Localização:

ganttproject\src\main\java\biz\ganttproject\lib\fx\Components.kt, linha 88

Usado em:

ganttproject\src\main\java\biz\ganttproject\storage\cloud\GPCloudSignupPane.kt, linha 54

VBoxBuilder
-i18n: RootLocalizer -vbox: VBox
+init() +addTitle(i18nKey: String, vararg args: String): Node +addTitleString(title: String): HBox +addTitle(title: LocalizedString): HBox +add(node: Node) +add(node: Node, alignment: Pos?, growth: Priority?): Node +addClasses(vararg classes: String) +addStylesheets(vararg stylesheets: String)

Explicação:

Um Builder é um design pattern criacional que permite ao utilizador construir objetos numa forma mais personalizada. Isto é possível porque este pattern separa o código que normalmente estaria no construtor em outros “sub” métodos.

Neste exemplo temos um VBoxBuilder que permite criar um painel personalizado para o painel de login do GanttProject Cloud.

Review 2

Este *Design Pattern* foi revisto e aprovado por Neel Badracim.

Pattern 3: Iterator

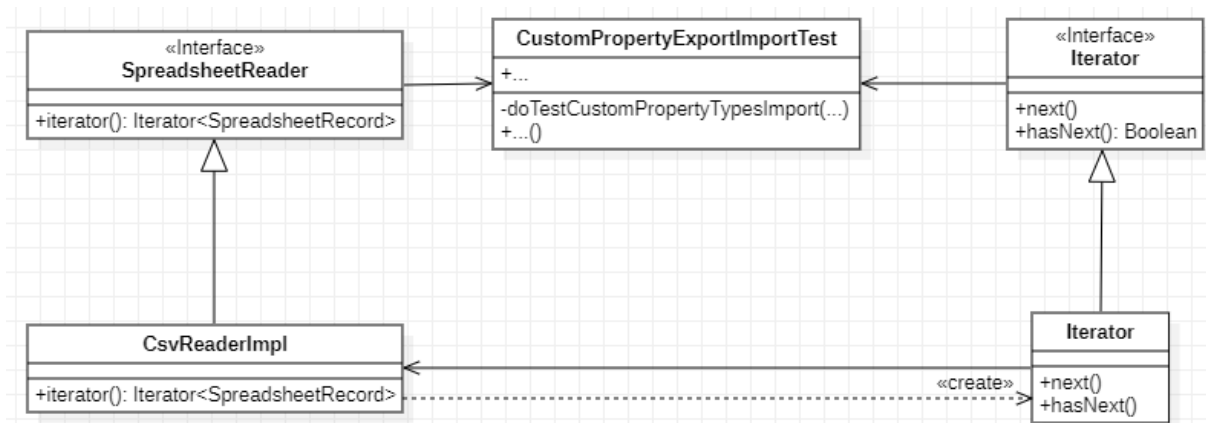
Localização:

ganttproject\src\main\java\biz\ganttproject\impex\csv\CSVImport.kt, linha 55

ganttproject\src\main\java\biz\ganttproject\impex\csv\Spreadsheet.kt

Usado em:

ganttproject-tester\test\biz\ganttproject\impex\csv, linha 107



Explicação:

Um Iterator fornece uma forma de percorrer uma coleção de objetos sem expor a sua implementação.

Neste exemplo nós temos um iterator que permite-nos navegar através de todos os registos de uma Spreadsheet sem expor a sua implementação ao sistema principal.

Review 3

Este *Design Pattern* foi revisto e aprovado por Ricardo Bessa.

Code Smells

Code Smell 1: Comments and Dead Code

```
override fun userInputConsumerChanged(newConsumer: Any?) {  
    // TODO: commit editing  
}
```

Localização do code smell a partir da raiz do repositório:

ganttproject\src\main\java\biz\ganttproject\ganttvview, linha 433.

Explicação:

Este código atualmente não está a fazer nada e têm somente um reminder que algum trabalho será desenvolvido aqui no futuro.

Proposta de refactorização:

Remover esta função

Review 1

Este *Code Smell* foi revisto e aprovado por Bruno Melo.

Code Smell 2: Shotgun surgery

```
private fun createCustomColumn(column: ColumnList.Column): TreeTableColumn<Task, *>? {
    val customProperty = taskManager.customPropertyManager.getCustomPropertyDefinition(column.id) ?: return null
    return when (customProperty.propertyClass) {
        CustomPropertyClass.TEXT -> {
            createTextColumn(customProperty.name,
                { taskTableModel.getValue(it, customProperty)?.toString() },
                { task, value -> undoManager.undoableEdit( localizedName: "Edit properties of task ${task.name}") {
                    taskTableModel.setValue(value, task, customProperty)
                } },
                { runBlocking { newTaskActor.inboxChannel.send(EditingCompleted()) } } })
        }
        CustomPropertyClass.BOOLEAN -> {
            createBooleanColumn<Task>(customProperty.name,
                { taskTableModel.getValue(it, customProperty) as Boolean? },
                { task, value -> undoManager.undoableEdit( localizedName: "Edit properties of task ${task.name}") {
                    taskTableModel.setValue(value, task, customProperty)
                } })
        }
        CustomPropertyClass.INTEGER -> {
            createIntegerColumn(customProperty.name,
                { taskTableModel.getValue(it, customProperty) as Int? },
                { task, value -> undoManager.undoableEdit( localizedName: "Edit properties of task ${task.name}") {
                    taskTableModel.setValue(value, task, customProperty)
                } })
        }
        CustomPropertyClass.DOUBLE -> {
            createDoubleColumn(customProperty.name,
                { taskTableModel.getValue(it, customProperty) as Double? },
                { task, value -> undoManager.undoableEdit( localizedName: "Edit properties of task ${task.name}") {
                    taskTableModel.setValue(value, task, customProperty)
                } })
        }
    }
}
```

Localização do code smell a partir da raiz do repositório:

ganttproject\src\main\java\biz\ganttproject\ganttvview, line number 566.

Explicação:

Esta função têm bastante código repetido, se por exemplo, quisermos mudar a mensagem nós teríamos de mudar individualmente cada linha

Proposta de refactorização:

Criar uma constante para a mensagem e chamar esta constante quando queremos mostrar a mensagem.

Review 2

Este *Code Smell* foi revisto e aprovado por João Pereira.

Code Smell 3: Dead code and Long parameter list

```
public AlgorithmCollection(  
    TaskManagerImpl taskManager,  
    FindPossibleDependeesAlgorithm myFindPossibleDependeesAlgorithm,  
    RecalculateTaskScheduleAlgorithm recalculateTaskScheduleAlgorithm,  
    AdjustTaskBoundsAlgorithm adjustTaskBoundsAlgorithm,  
    RecalculateTaskCompletionPercentageAlgorithm completionPercentageAlgorithm,  
    ChartBoundsAlgorithm projectBoundsAlgorithm, CriticalPathAlgorithm criticalPathAlgorithm,  
    AlgorithmBase scheduler) {  
    myScheduler = scheduler;  
    this.myFindPossibleDependeesAlgorithm = myFindPossibleDependeesAlgorithm;  
    myRecalculateTaskScheduleAlgorithm = recalculateTaskScheduleAlgorithm;  
    myAdjustTaskBoundsAlgorithm = adjustTaskBoundsAlgorithm;  
    myCompletionPercentageAlgorithm = completionPercentageAlgorithm;  
    myProjectBoundsAlgorithm = projectBoundsAlgorithm;  
    myCriticalPathAlgorithm = criticalPathAlgorithm;  
}
```

Localização do code smell a partir da raiz do repositório:

ganttproject\src\main\java\net\sourceforge\ganttproject\task\algorithm

Explicação:

Esta função tem uma variável morta e tem uma lista bastante longa de parâmetros.

Proposta de refactorização:

Remover a variável taskManger da lista de parâmetros e criar um parâmetro de objetos

Review 3

Este *Code Smell* foi revisto e aprovado por Neel Badracim.

João Pereira

Design Patterns

Pattern 1: Decorator Pattern

Localização:

```
\ganttproject\src\main\java\net\sourceforge\ganttproject\export\Exporter  
\ganttproject\src\main\java\net\sourceforge\ganttproject\export\ExporterBase  
\ganttproject\src\main\java\net\sourceforge\ganttproject\export\ExporterToCSV  
\ganttproject\src\main\java\net\sourceforge\ganttproject\export\ExporterToImage
```

Explicação:

O decorator pattern permite adicionar dinamicamente comportamentos a um objeto, usando interfaces e herança. Neste caso, o decorador é a classe abstrata *ExporterBase* que implementa a interface *Exporter*, e as classes *ExporterToCSV* e *ExporterToImage* são expandidas a partir desta.

Review 1

Este *Design Pattern* foi revisto e aprovado por Neel Badracim.

Pattern 2: Façade Pattern

Localização:

```
\ganttproject\src\main\java\net\sourceforge\ganttproject\task\TaskContainmentHierarchyFacade  
de  
\ganttproject\src\main\java\net\sourceforge\ganttproject\task\TaskTreeFacade.kt\FacadeImpl
```

Explicação:

Este *façade pattern* tem a interface *TaskContainmentHierarchyFacade* que é usada para “esconder” a complexidade, sendo implementada pelo *FacadeImpl*.

Review 2

Este *Design Pattern* foi revisto e aprovado por Ricardo Bessa.

Pattern 3: Factory Method Pattern

Localização:

`\ganttproject\src\main\java\net\sourceforge\ganttproject\action\task\OutdentTargetFunctionFactory`

`\ganttproject-tester\test\net\sourceforge\ganttproject\action\task\TaskMoveEnabledPredicateTest`

Explicação:

O *factory method pattern* é usado para “esconder” a criação de objetos. Neste caso, o *OutdentTargetFunctionFactory* é responsável por criar um objeto que é usado no *TaskMoveEnabledPredicateTest*.

Review 3

Este *Design Pattern* foi revisto e aprovado por Bruno Melo.

Code Smells

Code Smell 1: Data Class

Localização:

\ganttproject\src\main\java\net\sourceforge\ganttproject\resource\ResourceEvent

Explicação:

É uma *Data Class* porque apenas tem métodos do tipo `get()` e não tem funcionalidades.

Proposta de refatoração:

A solução poderá ser implementar os métodos usados pela classe *HumanResourceManager* para interagir com a classe *HumanResource*.

Review 1

Este *Code Smell* foi revisto e aprovado por Duarte Cruz.

Code Smell 2: Comments and Dead Code

```
@Override
public void exportPreferences(IEclipsePreferences node, IPreferenceFilter[] filters, OutputStream output)
    throws CoreException {
    // TODO Auto-generated method stub
}
```

Localização:

\ganttproject\src\main\java\net\sourceforge\ganttproject\PreferenceServiceImpl

Explicação:

Este método é *dead code* porque não é usado no projeto e, para além disso, tem um comentário para lembrar trabalho inacabado.

Proposta de refatoração:

Como o código não está a ser utilizado, uma solução possível será removê-lo.

Review 2

Este *Code Smell* foi revisto e aprovado por Neel Badracim.

Code Smell 3: Shotgun Surgery

```
case TaskDependencyConstraint.Collision.START_EARLIER_VARIATION:
    if (0 == (calendar.getDayMask(acceptableStart) & DayMask.WORKING)) {
        acceptableStart = calendar.findClosest(acceptableStart, myDstNode.myTask.getDuration().getTimeUnit(),
            GPCalendarCalc.MoveDirection.BACKWARD, GPCalendar.DayType.WORKING);
    }
    myStartRange = Range.upTo(acceptableStart, BoundType.CLOSED);
    break;
case TaskDependencyConstraint.Collision.START_LATER_VARIATION:
    if (0 == (calendar.getDayMask(acceptableStart) & DayMask.WORKING)) {
        acceptableStart = calendar.findClosest(acceptableStart, myDstNode.myTask.getDuration().getTimeUnit(),
            GPCalendarCalc.MoveDirection.FORWARD, GPCalendar.DayType.WORKING);
    }
    myStartRange = Range.downTo(acceptableStart, BoundType.CLOSED);
    break;
```

Localização:

\\ganttpproject\\src\\main\\java\\net\\sourceforge\\ganttpproject\\task\\algorithm\\DependencyGraph\\ExplicitDependencyImpl

Explicação:

Este método contém código repetido e se quisermos mudar algo no código temos de repetir a mesma mudança em todo o lado onde está a ser utilizado.

Proposta de refatoração:

A solução será criar um método contendo o código repetido, assim se quisermos alterar o método basta alterar um único método e não vários.

Review 3

Este *Code Smell* foi revisto e aprovado por Ricardo Bessa.

Neel Badracim

Design Patterns

Pattern 1: Proxy Pattern (Structural Design Pattern)

```
/**
 * Document which proxies all read methods and forbids all write methods.
 *
 * @author dbarashev (Dmitry Barashev)
 */
6 usages  dbarashev +1
public class ReadOnlyProxyDocument implements Document {
```

Localização do design pattern desde da raiz do repositório:

\\ganttp\\project\\src\\main\\java\\net\\sourceforge\\ganttp\\project\\document\\ReadOnlyProxyDocument.java

Explicação:

Podemos encontrar a Proxy Pattern na classe ReadOnlyProxyDocument . Esta classe age com uma versão simplificada da classe ProxyDocument, uma vez que atua como a classe “original” em alguns métodos mas está restrita a outros. Neste caso ReadOnlyProxyDocument apenas efetua métodos de leitura e dispensa os de escrita, tal como foi indicado pelo programador.

Review 1

Este *Design Pattern* foi revisto e aprovado por Ricardo Bessa.

Pattern 2: Iterator

```
private DefaultMutableTreeTableNode buildTree() {

    DefaultMutableTreeTableNode root = new DefaultMutableTreeTableNode();
    List<HumanResource> listResources = myResourceManager.getResources();
    Iterator<HumanResource> itRes = listResources.iterator();

    while (itRes.hasNext()) {
        HumanResource hr = itRes.next();
        ResourceNode rnRes = new ResourceNode(hr); // the first for the resource
        root.add(rnRes);
    }
    return root;
}
```

Localização do design pattern desde da raiz do repositório:

- ganttp\\project\\src\\main\\java\\net\\sourceforge\\ganttp\\project\\ResourceTreeTableModel.java


Explicação:

Neste caso, temos um Iterador que nos permite navegar por todos os elementos do tipo `HumanResource` sem que nos seja exposto a sua alta complexidade. O iterador oferece-nos métodos simples e assim ajuda-nos a manter o nosso código simples e seguro.

Review 2

Este *Design Pattern* foi revisto e aprovado por Bruno Melo.

Pattern 3: Singleton Method (Creational)



The first screenshot shows the class declaration: `protected static GanttLookAndFeels singleton;`. The second screenshot shows the constructor: `protected GanttLookAndFeels() { infoByClass = new HashMap<String, GanttLookAndFeelInfo>(); infoByName = new HashMap<String, GanttLookAndFeelInfo>(); LookAndFeelInfo[] lookAndFeels = UIManager.getInstalledLookAndFeels(); for (int i = 0; i < lookAndFeels.length; i++) { GanttLookAndFeelInfo info = new GanttLookAndFeelInfo(lookAndFeels[i]); addLookAndFeel(info); } }`. The third screenshot shows the static method: `public static GanttLookAndFeels getGanttLookAndFeels() { if (singleton == null) { singleton = new GanttLookAndFeels(); } return singleton; }`.

Localização do design pattern desde da raiz do repositório:

- `ganttproject\src\main\java\net\sourceforge\ganttproject\gui\GanttLookAndFeels.java`

Explicação:

Esta classe apresenta um construtor e um instância estática “protected”, como podemos ver acima, e por isso tem um Singleton design pattern. Esta instância apenas pode ser acedida através do método `GanttLookAndFeels()`. Este método certifica que apenas existe uma instância desta classe, pois apenas cria uma nova se o valor corrente for nulo ou devolve o valor previamente criado.

Review 3

Este *Design Pattern* foi revisto e aprovado por Duarte Cruz.

Code Smells

Code Smell 1: Data Class

Localização do design pattern desde da raiz do repositório:

- ganttproject\src\main\java\net\sourceforge\ganttpproject\client\RssUpdate.java.

Explicação:

Esta é uma Data Class uma vez que apenas apresenta métodos do tipo “getter”, o que significa que vai ser apenas guardada para guardar informação e não tem nenhuma funcionalidade real. De acordo com a aula teórica 8, isto indica que poderá não ser uma classe essencial ou uma abstração boa.

Alteração proposta:

Para alterar esta situação, uma possível solução seria implementar métodos que utilizassem os dados de forma adequada, adicionando algumas funções relevantes e não métodos “getters” e “setters”.

Review 1

Este *Code Smell* foi revisto e aprovado por João Pereira.

Code Smell 2: Dead Code

Localização do design pattern desde da raiz do repositório:

- ganttproject\src\main\java\net\sourceforge\ganttpproject\chart\ChartModelImpl.java
Line 53.

Explicação:

A variável privada não é usada em lado nenhum no código, ou seja, a sua existência é desnecessária para a classe.

Alteração proposta:

A variável deveria ser apagada uma vez que não utilizada. Removê-la ajudaria a simplificar o código.

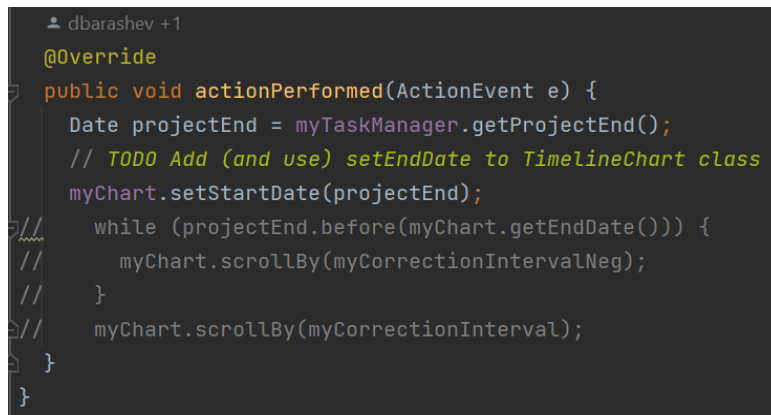
```
private Set<Task> myHiddenTasks;
```

⚠ Private field 'myHiddenTasks' is never used :53

Review 2

Este *Code Smell* foi revisto e aprovado por Ricardo Bessa.

Code Smell 3: Comments and Speculative Generality



```

@Override
public void actionPerformed(ActionEvent e) {
    Date projectEnd = myTaskManager.getProjectEnd();
    // TODO Add (and use) setEndDate to TimelineChart class
    myChart.setStartDate(projectEnd);
    while (projectEnd.before(myChart.getEndDate())) {
        myChart.scrollBy(myCorrectionIntervalNeg);
    }
    myChart.scrollBy(myCorrectionInterval);
}

```

Localização do design pattern desde da raiz do repositório:

- ganttproject\src\main\java\net\sourceforge\ganttp\action\scroll\ScrollToEndAction.java
Line 44-54.

Explicação:

Os comentários do tipo TODO feitos têm como objetivo lembrar futuramente o programador. Isto indica que o código está inacabado. Ao vermos novamente o método, para além disso, podemos ver que o programador já pensava numa futura implementação que foi deixada a meio e que não é necessária neste momento.

Refactoring proposal:

The best solution would be to choose simpler code rather than features that are not needed at this time. This means that the code should be deleted.

A melhor solução seria simplificar o código apagando os comentários que têm “reminder nature” e os comentários feitos para uma futura implementação.

Review 3

Este *Code Smell* foi revisto e aprovado por Bruno Melo.

Ricardo Bessa

Design Patterns

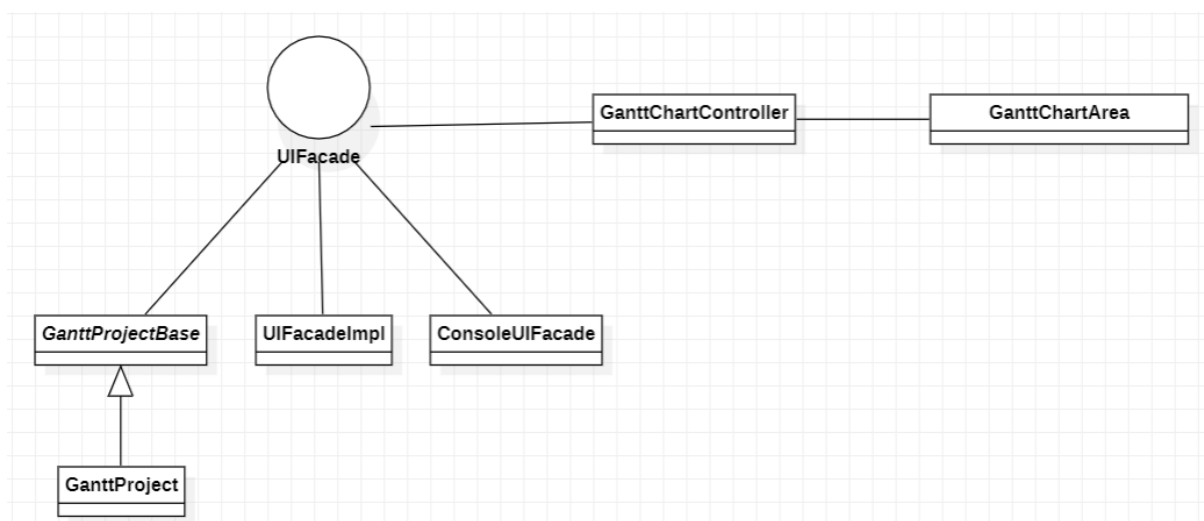
Pattern 1: Facade Pattern (Structural Design Pattern)

Localização do design pattern a partir da repository root:

- ganttproject/src/main/java/net/sourceforge/ganttproject/gui/UIFacade.java
- ganttproject/src/main/java/net/sourceforge/ganttproject/chart/gantt/GanttChartController.java
- ganttproject/src/main/java/net/sourceforge/ganttproject/export/ConsoleUIFacade.java
- ganttproject/src/main/java/net/sourceforge/ganttproject/GanttProject.java
- ganttproject/src/main/java/net/sourceforge/ganttproject/GanttProjectBase.java
- ganttproject/src/main/java/net/sourceforge/ganttproject/UIFacadeImpl.java
- ganttproject/src/main/java/net/sourceforge/ganttproject/GanttGraphicArea.java

Explicação:

Podemos encontrar um Facade Pattern na forma como a Interface UIFacade esconde um subsistema mais complexo que inclui classes como GanttProjectBase.java, UIFacadeImpl.java e ConsoleUIFacade.java, as quais poderão ser acedidas, por exemplo, através da classe GanttChartController.java. Outras classes como GanttChartArea.java podem interagir com este subsistema através da classe GanttChartController.java. Isto pode ser observado através do seguinte diagrama de classes UML simplificado.



Review 1

Este *Design Pattern* foi revisto e aprovado por Bruno Melo.

Pattern 2: Singleton Pattern (Creational Design Pattern)

Localização do design pattern a partir da repository root:

ganttproject/src/main/java/net/sourceforge/ganttproject/calendar/GPCalendarProvider.java

Explicação:

A classe GPCalendarProvider.java tem um construtor privado (line 109) e uma instância privada desta classe. Esta instância só pode ser acessada através do método getInstance() e este método garante que vai existir apenas uma única instância desta classe pois apenas cria uma nova instância caso esta estivesse null, caso contrário retorna a instância criada anteriormente.

```
70
    3 usages
71     private static GPCalendarProvider ourInstance;
```

```
109     private GPCalendarProvider(List<GPCalendar> calendars) { myCalendars = calendars; }
112
```

```
95     public static synchronized GPCalendarProvider getInstance() {
96         if (ourInstance == null) {
97             List<GPCalendar> calendars = readCalendars();
98             Collections.sort(calendars, new Comparator<GPCalendar>() {
99                 public int compare(GPCalendar o1, GPCalendar o2) { return o1.getName().compareTo(o2.getName()); }
102             });
103             ourInstance = new GPCalendarProvider(calendars);
104         }
105         return ourInstance;
106     }
```

Review 2

Este *Design Pattern* foi revisto e aprovado por Duarte Cruz.

Pattern 3: Abstract Factory Pattern (Creational)

Localização do design pattern a partir da repository root:

- biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilder.java
- biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java
- ganttproject/src/main/java/net/sourceforge/ganttproject/chart/ChartModelBase.java

Explicação:

A interface OffsetBuilder.java inclui uma classe abstrata chamada Factory. Esta Factory é estendida pela classe FactoryImpl, a qual pode ser encontrada dentro da classe OffsetBuilderImpl.java. Esta factory é usada para criar instâncias da classe OffsetBuilderImpl.java. Esta criação ocorre, por exemplo, na classe ChartModelBase.java. Desta forma é possível criar instâncias de uma certa família de instâncias sem especificar as suas classes.

```
18 usages 1 implementation dbarashev
31 public interface OffsetBuilder {
    1 inheritor dbarashev
32 public static abstract class Factory {
```

```
6 usages dbarashev
49 public Factory withTopUnit(TimeUnit topUnit) {
50     myTopUnit = topUnit;
51     return this;
52 }
53
6 usages dbarashev
54 public Factory withBottomUnit(TimeUnit bottomUnit) {
55     myBottomUnit = bottomUnit;
56     return this;
57 }
58
dbarashev
59 public Factory withStartDate(Date startDate) {
60     myStartDate = startDate;
61     return this;
62 }
63
6 usages dbarashev
64 public Factory withViewportStartDate(Date viewportStartDate) {
65     myViewportStartDate = viewportStartDate;
66     return this;
67 }
68
1 usage dbarashev
69 public Factory withEndDate(Date endDate) {
70     myEndDate = endDate;
71     return this;
72 }
73
```

```
189 public static class FactoryImpl extends OffsetBuilder.Factory {
    dbarashev
190 @Override
191 public OffsetBuilder build() {
192     preBuild();
193     return new OffsetBuilderImpl( factory: this);
194 }
195 }
196
```

```
3 usages dbarashev
454 public OffsetBuilder.Factory createOffsetBuilderFactory() {
455     OffsetBuilder.Factory factory = new OffsetBuilderImpl.FactoryImpl()
456         .withAtomicUnitWidth(getBottomUnitWidth())
457         .withBottomUnit(getBottomUnit())
458         .withCalendar(myTaskManager.getCalendar())
459         .withRightMargin(myScrollingSession == null ? 0 : 1)
460         .withStartDate(getOffsetAnchorDate())
461         .withViewportStartDate(getStartDate())
462         .withTopUnit(myTopUnit)
463         .withWeekendDecreaseFactor(
464             getTopUnit().isConstructedFrom(getBottomUnit()) ? OffsetBuilderImpl.WEEKEND_UNIT_WIDTH_DECREASE_FACTOR : 1f);
465     if (getBounds() != null) {
466         factory.withEndOffset((int) getBounds().getWidth());
467     }
468     return factory;
469 }
```

```
12 usages dbarashev
401 private OffsetManager myOffsetManager = new OffsetManager(new OffsetBuilder.Factory() {
    1 usage dbarashev
402 @Override
403 public OffsetBuilder createTopAndBottomUnitBuilder() {
404     return createOffsetBuilderFactory().build();
405 }
```

Review 3

Este *Design Pattern* foi revisto e aprovado por João Pereira.

Code Smells

Code Smell 1: Comments

Localização do design pattern a partir da repository root:

ganttproject/src/main/java/org/imgscalr/Scalr.java

Explicação:

Esta classe tem demasiados comentários com um número exagerado de indicações e explicações sobre todos os métodos. Podemos encontrar comentários bastante longos como o comentário inicial que tem um total de 161 linhas. Nesta classe podemos também encontrar o Enum Mode que tem um comentário para cada constante. No geral acabamos por encontrar mais comentários do que exatamente código. Todas estas indicações podem sugerir que o código não tem o design apropriado ou que é demasiado complexo, sendo necessário muito tempo para que novos developers o possam compreender.

```
418 public static enum Mode {
419     /**
420      * Used to indicate that the scaling implementation should calculate
421      * dimensions for the resultant image by looking at the image's
422      * orientation and generating proportional dimensions that best fit into
423      * the target width and height given
424      */
425     /** See "Image Proportions" in the {@link Scalr} class description for
426      * more detail.
427      */
428     18 usages
429     AUTOMATIC,
430     /**
431      * Used to fit the image to the exact dimensions given regardless of the
432      * image's proportions. If the dimensions are not proportionally
433      * correct, this will introduce vertical or horizontal stretching to the
434      * image.
435      * <p/>
436      * It is recommended that you use one of the other <code>FIT_TO</code>
437      * modes or {@link Mode#AUTOMATIC} if you want the image to look
438      * correct, but if dimension-fitting is the #1 priority regardless of
439      * how it makes the image look, that is what this mode is for.
440     */
441     3 usages
442     FIT_EXACT,
443     /**
444      * Used to indicate that the scaling implementation should calculate
445      * dimensions for the largest image that fit within the bounding box,
446      * without cropping or distortion, retaining the original proportions.
447     */
448 }
```

Proposta de resolução:

Simplificar o código e os comentários, eliminando indicações desnecessárias e deixando apenas as que forem essenciais. Desta forma esta classe vai ficar mais curta e fácil de compreender.

Review 1

Este *Code Smell* foi revisto e aprovado por Neel Badracim.

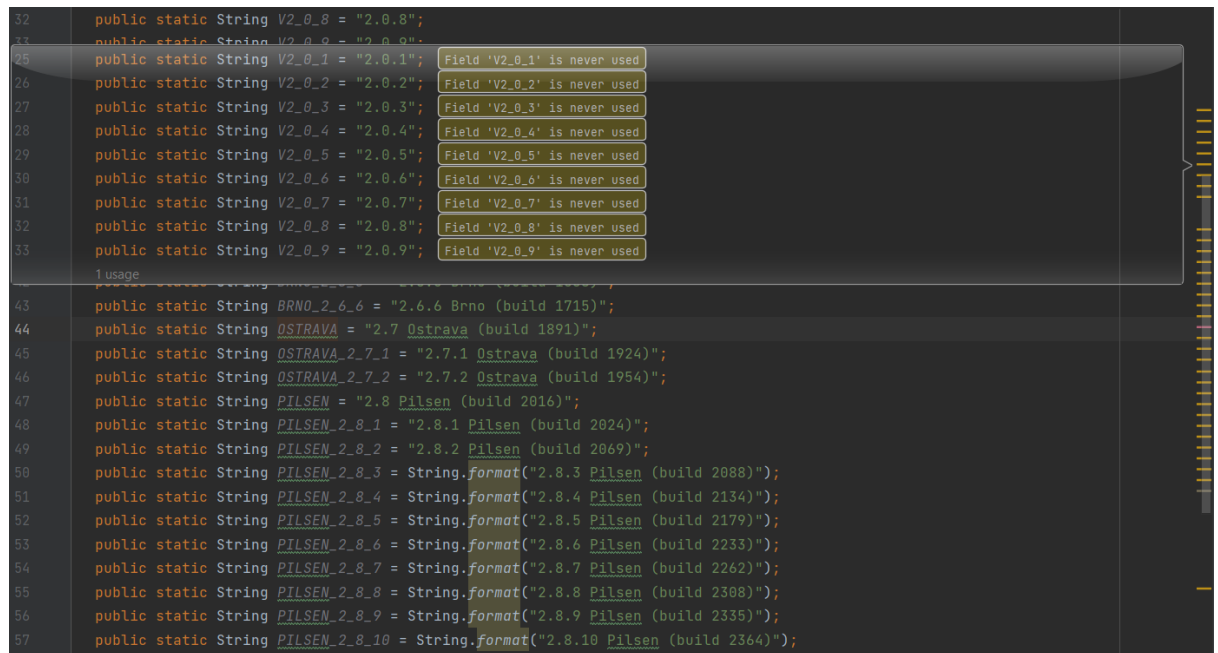
Code Smell 2: Dead Code

Localização do design pattern a partir da repository root:

ganttproject/src/main/java/net/sourceforge/ganttproject/GPVersion.java

Explicação:

Na classe abstrata GPVersion.java podemos encontrar variáveis estáticas que nunca são usadas.



```
32 public static String V2_0_8 = "2.0.8";
33 public static String V2_0_9 = "2.0.9";
34 public static String V2_0_1 = "2.0.1";
35 public static String V2_0_2 = "2.0.2";
36 public static String V2_0_3 = "2.0.3";
37 public static String V2_0_4 = "2.0.4";
38 public static String V2_0_5 = "2.0.5";
39 public static String V2_0_6 = "2.0.6";
40 public static String V2_0_7 = "2.0.7";
41 public static String V2_0_8 = "2.0.8";
42 public static String V2_0_9 = "2.0.9";
43 public static String BRNO_2_6_6 = "2.6.6 Brno (build 1715)";
44 public static String OSTRAVA = "2.7 Ostrava (build 1891)";
45 public static String OSTRAVA_2_7_1 = "2.7.1 Ostrava (build 1924)";
46 public static String OSTRAVA_2_7_2 = "2.7.2 Ostrava (build 1954)";
47 public static String PILSEN = "2.8 Pilsen (build 2016)";
48 public static String PILSEN_2_8_1 = "2.8.1 Pilsen (build 2024)";
49 public static String PILSEN_2_8_2 = "2.8.2 Pilsen (build 2069)";
50 public static String PILSEN_2_8_3 = String.format("2.8.3 Pilsen (build 2088)");
51 public static String PILSEN_2_8_4 = String.format("2.8.4 Pilsen (build 2134)");
52 public static String PILSEN_2_8_5 = String.format("2.8.5 Pilsen (build 2179)");
53 public static String PILSEN_2_8_6 = String.format("2.8.6 Pilsen (build 2233)");
54 public static String PILSEN_2_8_7 = String.format("2.8.7 Pilsen (build 2262)");
55 public static String PILSEN_2_8_8 = String.format("2.8.8 Pilsen (build 2308)");
56 public static String PILSEN_2_8_9 = String.format("2.8.9 Pilsen (build 2335)");
57 public static String PILSEN_2_8_10 = String.format("2.8.10 Pilsen (build 2364)");
```

Proposta de resolução:

Remover estas variáveis e apenas adicioná-las quando forem necessárias.

Review 2

Este *Code Smell* foi revisto e aprovado por Bruno Melo.

Code Smell 3: Comments e Speculative generality

Localização do design pattern a partir da repository root:

org.ganttproject.impex.htmlpdf/src/main/java/org/ganttproject/impex/htmlpdf/itext/ITextEngine.java

Explicação:

Nesta classe podemos encontrar o método run (linha 137) que contém algum código que precisa de ser corrigido e completado no futuro, tal como os comentários mencionam (linhas 143 e 148).

```
134     private void registerFonts() {
135         Thread fontReadingThread = new Thread(new Runnable() {
136             @Override
137             public void run() {
138                 var logger :LoggerApi<Logger> = GPLogger.create("Export.Pdf.Fonts");
139                 try {
140                     // Random waiting seems silly, depending on the available
141                     // resources (CPU speed, number of processes running etc)
142                     // this might take longer or shorter...
143                     // FIXME Add some better way of determining whether the fonts can be
144                     // read already
145                     Thread.sleep( millis: 10000);
146                     logger.debug( msg: "Scanning font directories...");
147                 } catch (InterruptedException e) {
148                     // TODO Auto-generated catch block
149                     GPLogger.logToLogger(e);
150                 }
151                 registerFontDirectories();
152                 synchronized (ITextEngine.this.myFontsMutex) {
153                     myFontsReady = true;
154                     myFontsMutex.notifyAll();
155                 }
156                 logger.debug( msg: "Scanning font directories completed");
157             }
158         });
```

Proposta de resolução:

Corrigir o código antes de avançar para outras tarefas ou apagá-lo até tal ser possível .

Review 3

Este *Code Smell* foi revisto e aprovado por Duarte Cruz.

2ª Fase

Use Case Diagrams

Uma versão mais completa destes Use Case Diagrams pode ser encontrada em:

https://docs.google.com/document/d/16Lq4JsX9LKJYTsGeWHNZLvbmNWCoY3tcUKj8JC-V8RY/edit?usp=share_link

Para efeitos da elaboração deste relatório resolvemos usar uma versão resumida da versão acima mencionada.

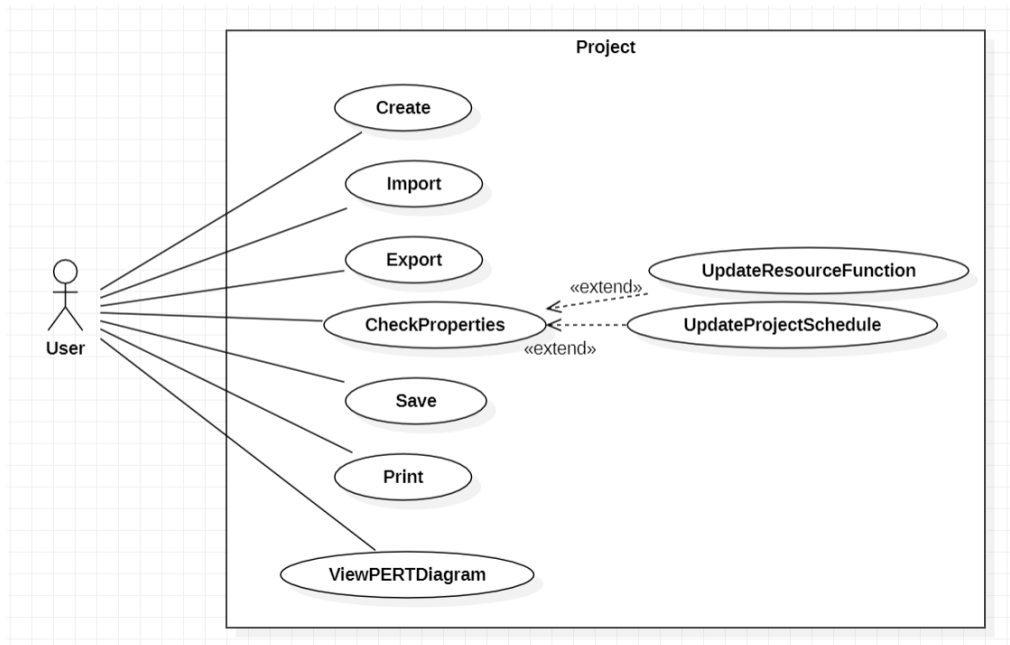
A distribuição de responsabilidade de análise e revisão de cada Use case pode ser consultada na seguinte tabela:

Use Case	Elaborado por	Revisto Por
Project	Bruno Melo	João Pereira
Edit	Duarte Cruz	Neel Badracim
Tasks	João Pereira	Ricardo Bessa
Resources	Neel Badracim	Duarte Cruz
Help	Ricardo Bessa	Bruno Melo

Use case 1 - Project

Elaborado por: Bruno

Revisto por: João



Primary Actors:

User.

Secondary Actors:

None.

Sub-use case example:

Use case: CreateProject
ID: 1
Brief description: The User creates a new project in the application.
Primary Actors: User.
Secondary Actors: None.
Preconditions: None.
Main flow: <ol style="list-style-type: none">1. The use case starts when the User selects the option “New” in the Project menu or using the “CTRL + N” shortcut.2. The User inserts the name and optional additional information in the system.3. The system validates the inserted data.4. The system creates a new project for the User.

Postconditions: 1. A new project is created
Alternative flows: 1. CreateProject:Cancel

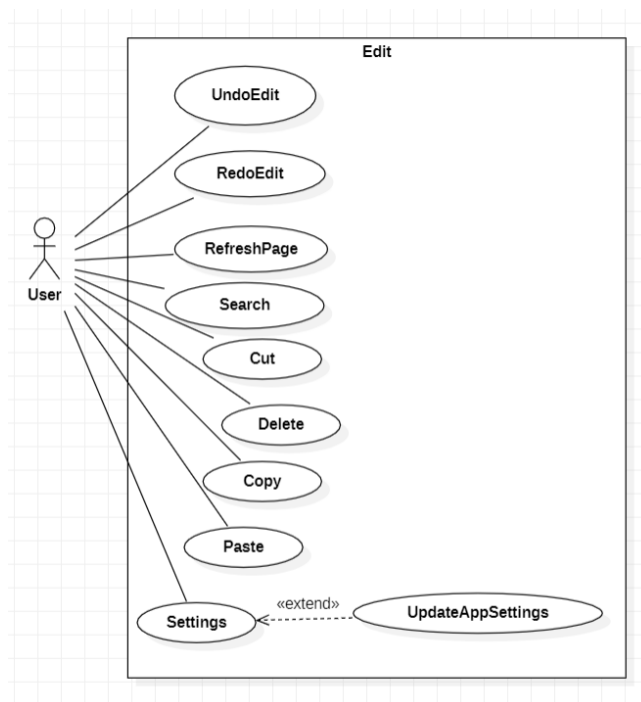
Other sub-use cases:

- ImportProject
 - ImportProject:Cancel
 - ImportProject:WrongFileFormat
- ExportProject
 - ExportProject:Cancel
- CheckProjectProperties
 - CheckProjectProperties:Close
- UpdateResourceFunction
- UpdateProjectSchedule
- SaveProject
 - SaveProject:Cancel
- PrintProject
 - PrintProject:Cancel
- ViewPERTDiagram

Use case 2 - Edit

Elaborado por: Duarte

Revisto por: Neel



Primary Actors:

User.

Secondary Actors:

None.

Sub-use case example:

Use case: UndoEdit
ID: 2
Brief description: Reverses the last action made by the User in the application.
Primary Actors: User.
Secondary Actors: None.
Preconditions: A change in the application was made by the User.
Main flow: <ol style="list-style-type: none"> 1. The use case begins when the User selects "Undo" in the Edit menu or using the "CTRL + Z" shortcut. 2. The action made is undone.
Postconditions: The system goes back into the previous state.

Alternative flows: None.

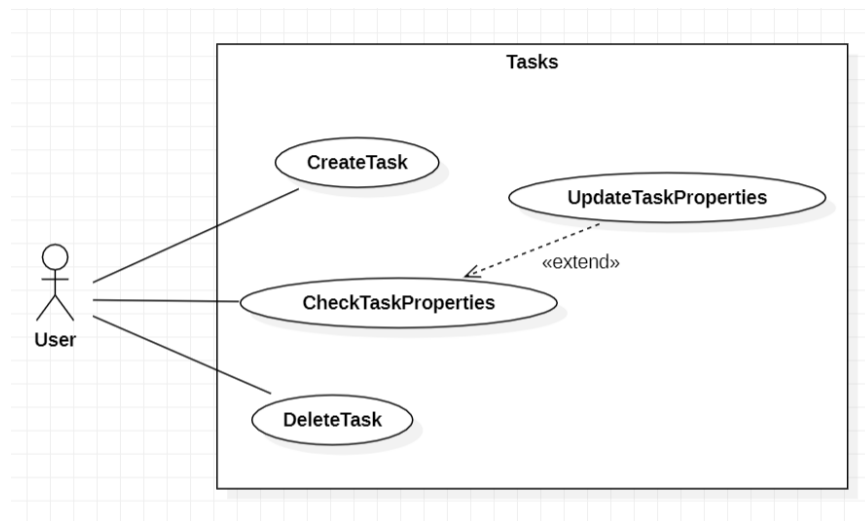
Other sub-use cases:

- RedoEdit
- RefreshPage
- Search
 - SearchNotFound
- Cut
- Delete
- Copy
- Paste
- Settings
- UpdateAppSettings
 - UpdateAppSettings:Cancel

Use case 3 - Tasks

Elaborado por: João

Revisto por: Ricardo



Primary Actors:

User.

Secondary Actors:

None.

Preconditions:

None.

Sub-use case example:

Use case: CreateTask
ID: 3
Brief description: The User creates a new task.
Primary Actors: User.
Secondary Actors: None.
Preconditions: None.
Main flow: 1. The User selects "New Task" or uses the "CTRL + T" shortcut. 2. The System creates a new task.
Postconditions: A new task is created.
Alternative flows: None.

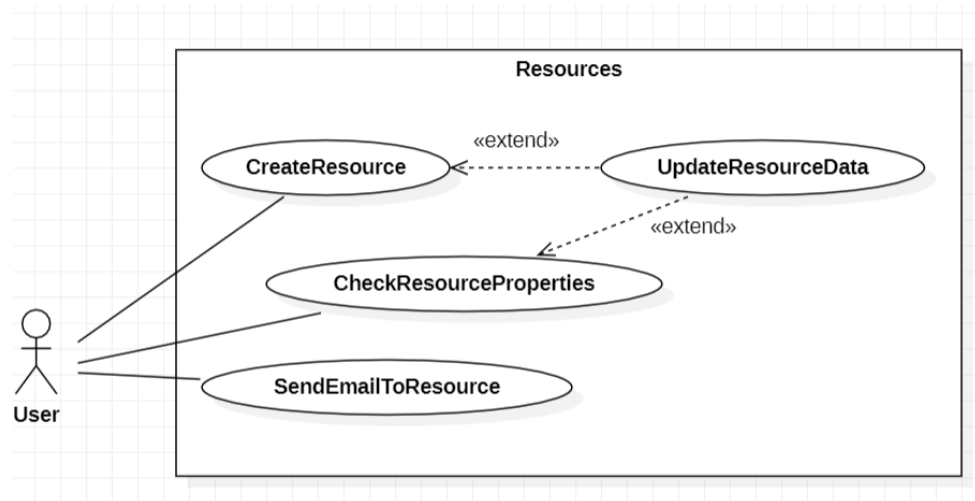
Other sub-use cases:

- CreateTask
- CheckTaskProperties
 - CheckTaskProperties:Close
- UpdateTaskProperties

Use case 4 - Resources

Elaborado por: Neel

Revisto por: Duarte



Primary Actors:

User.

Secondary Actors:

None.

Sub-use case example:

Use case: CreateResource
ID: 4
Brief description: The User creates a new resource.
Primary Actors: User.
Secondary Actors: None.
Preconditions: None.
Main flow: <ol style="list-style-type: none">1. The User selects "New Resource" in the Resources menu.2. The System opens the resource properties menu.3. Extend(UpdateResourceData).4. The System validates the data.5. The System creates a new resource with the validated data.
Postconditions: A new resource is created.
Alternative flows: <ol style="list-style-type: none">1. SaveProject:Cancel

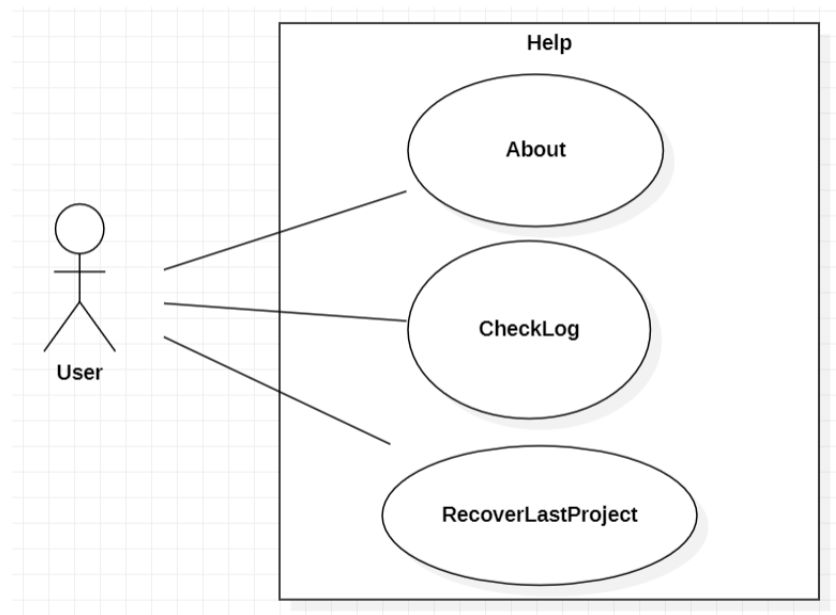
Other sub-use cases:

- CreateResource
- CreateResource:Cancel
- CheckResourceProperties
- CheckResourceProperties:Close
- UpdateResourceData
- SendEmailToResource

Use case 5 - Help

Elaborado por: Ricardo

Revisto por: Bruno



Primary Actors:

User.

Secondary Actors:

None.

Sub-use case example:

Use case: See About
ID: 5
Brief description: The system displays information about the application for the User.
Primary Actors: User.
Secondary Actors: None.
Preconditions: None.
Main flow: <ol style="list-style-type: none">1. The use case begins when the User selects “About” in the Help menu.2. The system displays a pop up with relevant information of the application for the User.
Postconditions: <ol style="list-style-type: none">1. The system has displayed the application details

Alternative flows:
None.

Other sub-use cases:

- CheckLog
- RecoverLastProject

Métricas

Todas as métricas presentes neste relatório foram obtidas utilizando o IntelliJ IDEA com o plugin MetricsReloaded.

A distribuição de responsabilidade de análise e revisão de cada métrica pode ser consultada na seguinte tabela:

Métrica	Elaborado por	Revisto Por
Chidamber-Kemerer	Duarte Cruz	Ricardo Bessa
Complexity Metrics	Neel Badracim	João Pereira
Dependency Metrics	Ricardo Bessa	Duarte Cruz
Lines of Code Metrics	João Pereira	Bruno Melo
Martin Packaging Metrics	Bruno Melo	Neel Badracim

Chidamber-Kemerer

Elaborado por: Duarte Cruz

Revisto por: Ricardo Bessa

Todas as métricas obtidas podem ser consultadas no ficheiro em anexo sobre a métrica

Chidamber-Kemerer ou então em:

<https://docs.google.com/spreadsheets/d/1ldVYIDyGgQoXbje5IIQloZ4C3SYOVGFTbH6XznW9rk/edit?usp=sharing>

Significado das siglas:

CBO	number of classes to which a class is coupled
DIT	maximum inheritance path from the class to the root class
LCOM	lack of Cohesion of Methods
NOC	number of immediate subclasses of a class
RFC	Response for class
WMC	number of methods defined in class

Tabelas com os 10 valores mais elevados em cada métrica

	CBO			LCOM
net.sourceforge.ganttproject.langu age.GanttLanguage	133.0		net.sourceforge.ganttproject.Gantt ProjectBase	31.0
net.sourceforge.ganttproject.Gantt Project	121.0		net.sourceforge.ganttproject.export .ConsoleUIFacade	31.0
net.sourceforge.ganttproject.action. GPAAction	106.0		net.sourceforge.ganttproject.gui.UI Util	22.0
net.sourceforge.ganttproject.GPLog gger	91.0		net.sourceforge.ganttproject.Gantt ProjectImpl	21.0
net.sourceforge.ganttproject.task.T askManagerImpl	89.0		net.sourceforge.ganttproject.UIFac adeImpl	20.0
net.sourceforge.ganttproject.resour ce.HumanResource	78.0		net.sourceforge.ganttproject.Prefer enceServiceImpl	19.0
net.sourceforge.ganttproject.chart. ChartModelBase	74.0		biz.ganttproject.core.calendar.Alwa ysWorkingTimeCalendarImpl	17.0
net.sourceforge.ganttproject.gui.UI Util	71.0		net.sourceforge.ganttproject.task.T askManager.TaskBuilder	17.0
net.sourceforge.ganttproject.resour ce.HumanResourceManager	66.0		net.sourceforge.ganttproject.Gantt ResourcePanel	16.0
biz.ganttproject.core.option.GPOpti onGroup	65.0		net.sourceforge.ganttproject.gui.UI Configuration	16.0

	DIT		NOC
net.sourceforge.ganttproject.GanttTreeTable	8.0	net.sourceforge.ganttproject.action.GPAction	89.0
net.sourceforge.ganttproject.ResourceTreeTable	8.0	net.sourceforge.ganttproject.test.task.TaskTestCase	32.0
net.sourceforge.ganttproject.GPTreeTableBase	7.0	net.sourceforge.ganttproject.parser.AbstractTagHandler	19.0
net.sourceforge.ganttproject.GanttProject	7.0	net.sourceforge.ganttproject.action.OkAction	16.0
net.sourceforge.ganttproject.action.edit.EditMenu	7.0	biz.ganttproject.core.option.DefaultEnumerationOption	14.0
net.sourceforge.ganttproject.action.project.ProjectMenu	7.0	biz.ganttproject.core.option.GPAbstractOption	13.0
net.sourceforge.ganttproject.action.view.ViewMenu	7.0	net.sourceforge.ganttproject.io.SaverBase	13.0
net.sourceforge.ganttproject.gui.GTextField	7.0	biz.ganttproject.impex.csv.RecordGroup	11.0
net.sourceforge.ganttproject.gui.ProjectMRUMenu	7.0	net.sourceforge.ganttproject.action.CancelAction	9.0
net.sourceforge.ganttproject.GanttGraphicArea	6.0	net.sourceforge.ganttproject.export.ExporterBase.ExporterJob	9.0
	RFC		WMC
net.sourceforge.ganttproject.GanttProject	374.0	net.sourceforge.ganttproject.task.TaskManagerImpl	173.0
net.sourceforge.ganttproject.task.TaskManagerImpl	260.0	net.sourceforge.ganttproject.GanttProject	159.0
biz.ganttproject.impex.msproject2.ProjectFileImporter	248.0	org.ganttproject.chart.pert.ActivityOnNodePertChart	142.0
net.sourceforge.ganttproject.UIFacadeImpl	194.0	biz.ganttproject.impex.msproject2.ProjectFileImporter	135.0
biz.ganttproject.impex.msproject2.ProjectFileExporter	190.0	net.sourceforge.ganttproject.task.TaskImpl	131.0
net.sourceforge.ganttproject.GPTreeTableBase	189.0	net.sourceforge.ganttproject.GPTreeTableBase	113.0
net.sourceforge.ganttproject.gui.GanttTaskPropertiesBean	185.0	net.sourceforge.ganttproject.gui.options.OptionsPageBuilder	113.0
org.ganttproject.impex.htmlpdf.itext.ThemeImpl	182.0	net.sourceforge.ganttproject.UIFacadeImpl	109.0
net.sourceforge.ganttproject.GanttOptions	180.0	net.sourceforge.ganttproject.chart.ChartModelBase	109.0
net.sourceforge.ganttproject.task.T	169.0	net.sourceforge.ganttproject.Gantt	106.0

askImpl		Options	
---------	--	---------	--

Análise das métricas obtidas

Nesta análise foram apenas ponderadas as métricas mais relevantes a este projeto, tendo assim sido ignoradas para efeitos de análise a métrica NOC e LCOM

CBO - Number of classes to which a class is coupled

Como não é de estranhar a classe que está mais interligada a outras é a GanttLanguage, isto é devido ao facto que é esta classe que trata da tradução do projeto. Sendo assim cada classe, com um output visível pelo utilizador, têm de chamar esta classe.

As outras classes no top 10 do CBO são maioritariamente classes de sistema, ou seja, classes que chamam outras. Porém, há uma notável exceção na classe GPLogger, isto pois é esta classe que avisa o utilizador de erros que tenham acontecido durante o runTime. Por tanto, onde é possível que surja um erro esta classe está lá.

DIT - Maximum inheritance path from the class to the root class

Como não é de admirar 7/10 classes com o resultado mais alto da métrica DIT ou são Menus, ou então são classes estruturadas em Árvore. Destas 10 classes que ocupam o pódio na métrica DIT queremos salientar a classe GanttProject que mesmo não estando estruturada nem em Árvore, nem em Menu está nesta tabela devido à sua complexidade.

RFC - Response for class

Como podemos observar, a classe que está engloba mais métodos e que chama mais métodos de outras classes é a GanttProject. Isto já é expectável visto que esta é a principal classe sistema, logo a classe que vai "ligar" tudo.

WMC - Number of methods defined in class

Podemos observar que pela razão mencionada no RFC a classe GanttProject ocupa a 2ª posição do ranking do WMC. Porém o 1º lugar vai para a classe TaskManagerImpl. Esta classe ocupa o pódio pois está de certa forma a servir como o "cérbero" da class GanttProject, garantindo assim que todas as tasks a serem executadas estão em "sincronia"

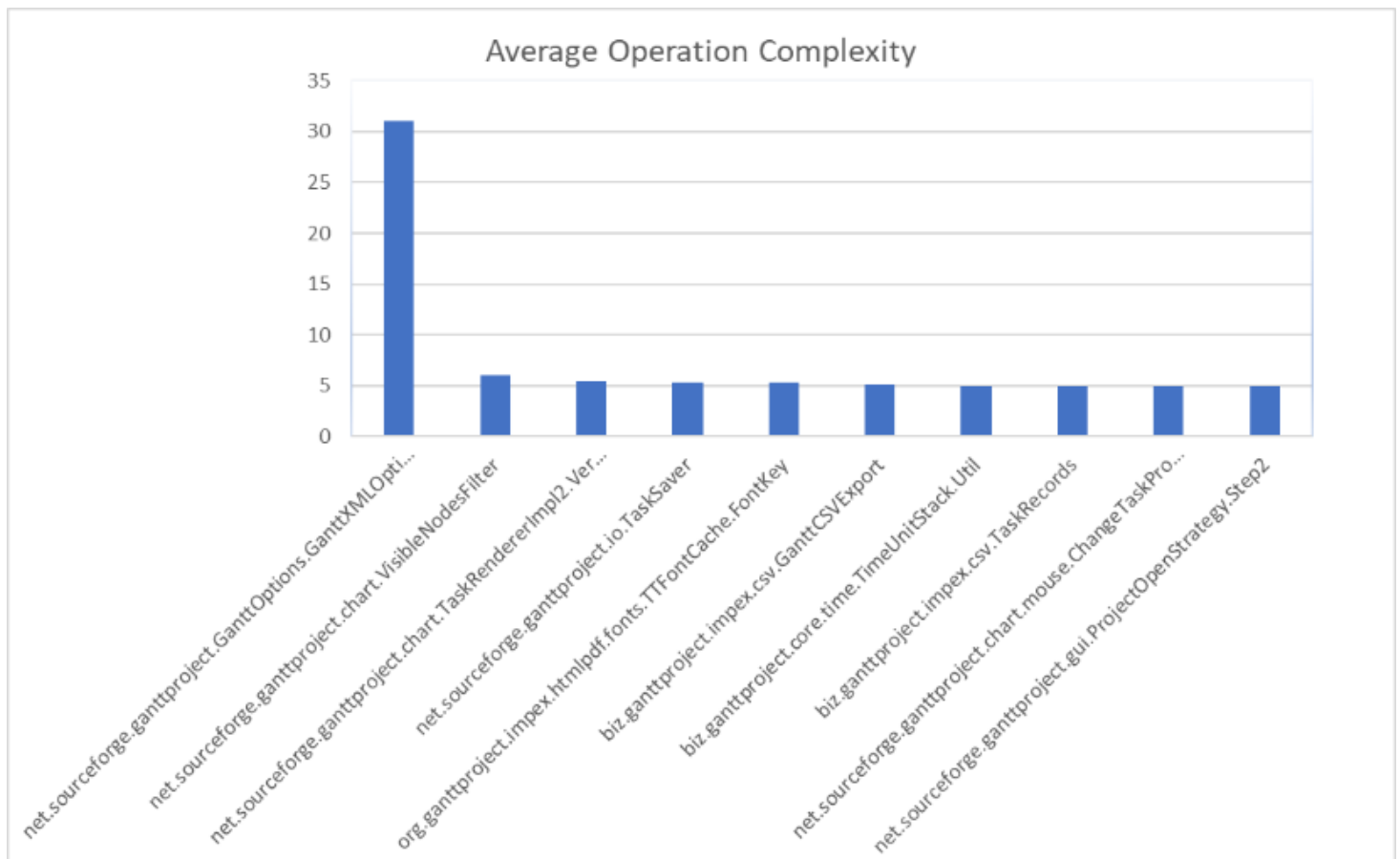
Complexity Metrics

Elaborado por: Neel Badracim

Revisto por: João Pereira

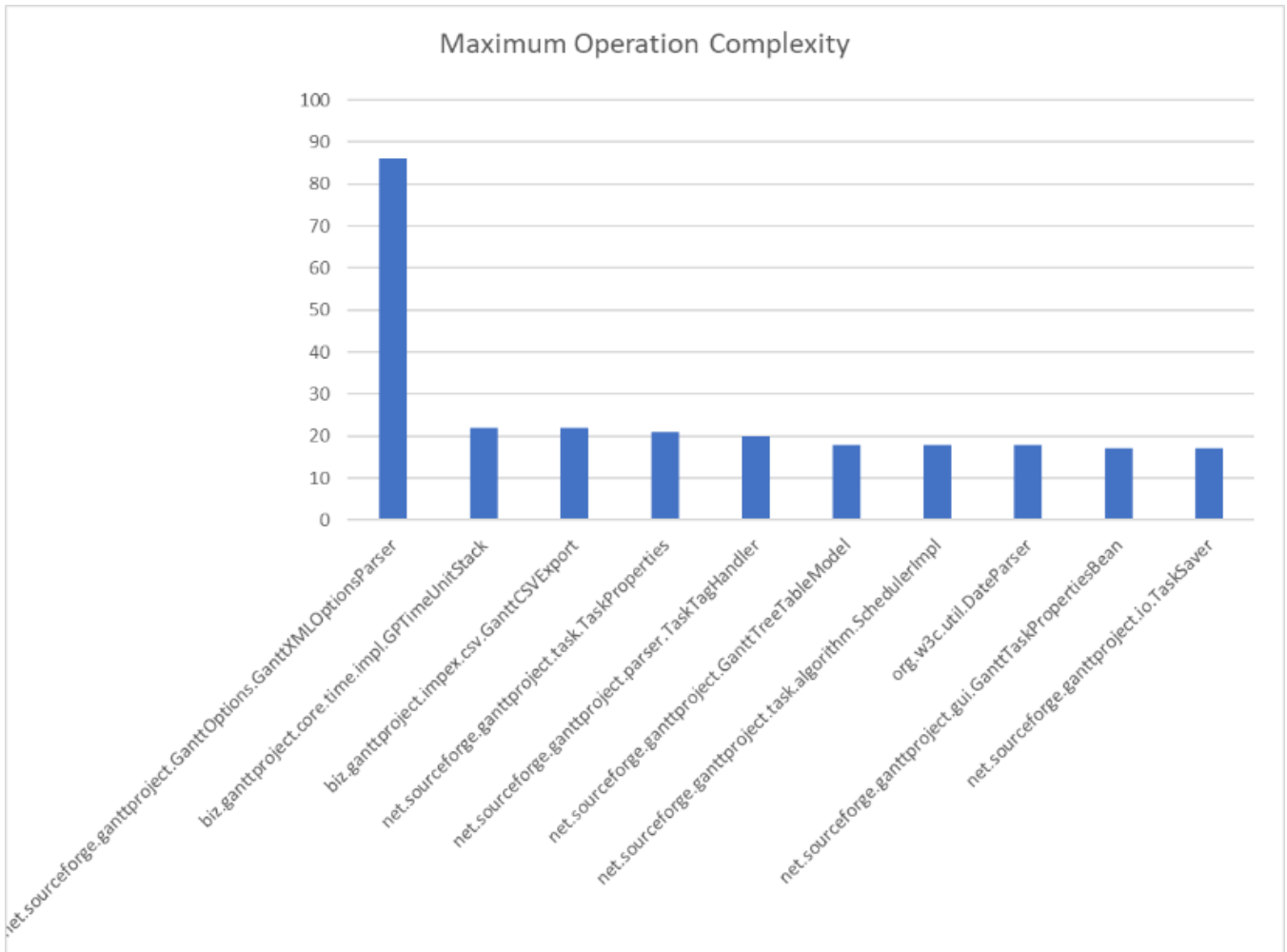
Para uma consulta mais detalhada consultar o seguinte link:
https://docs.google.com/spreadsheets/d/1Hkqu5N6TV-thTcfuX_8-HMI3Io_3BhFpifG9LqtgkGE/edit#gid=0

- **OCavg - Average Operation Complexity**



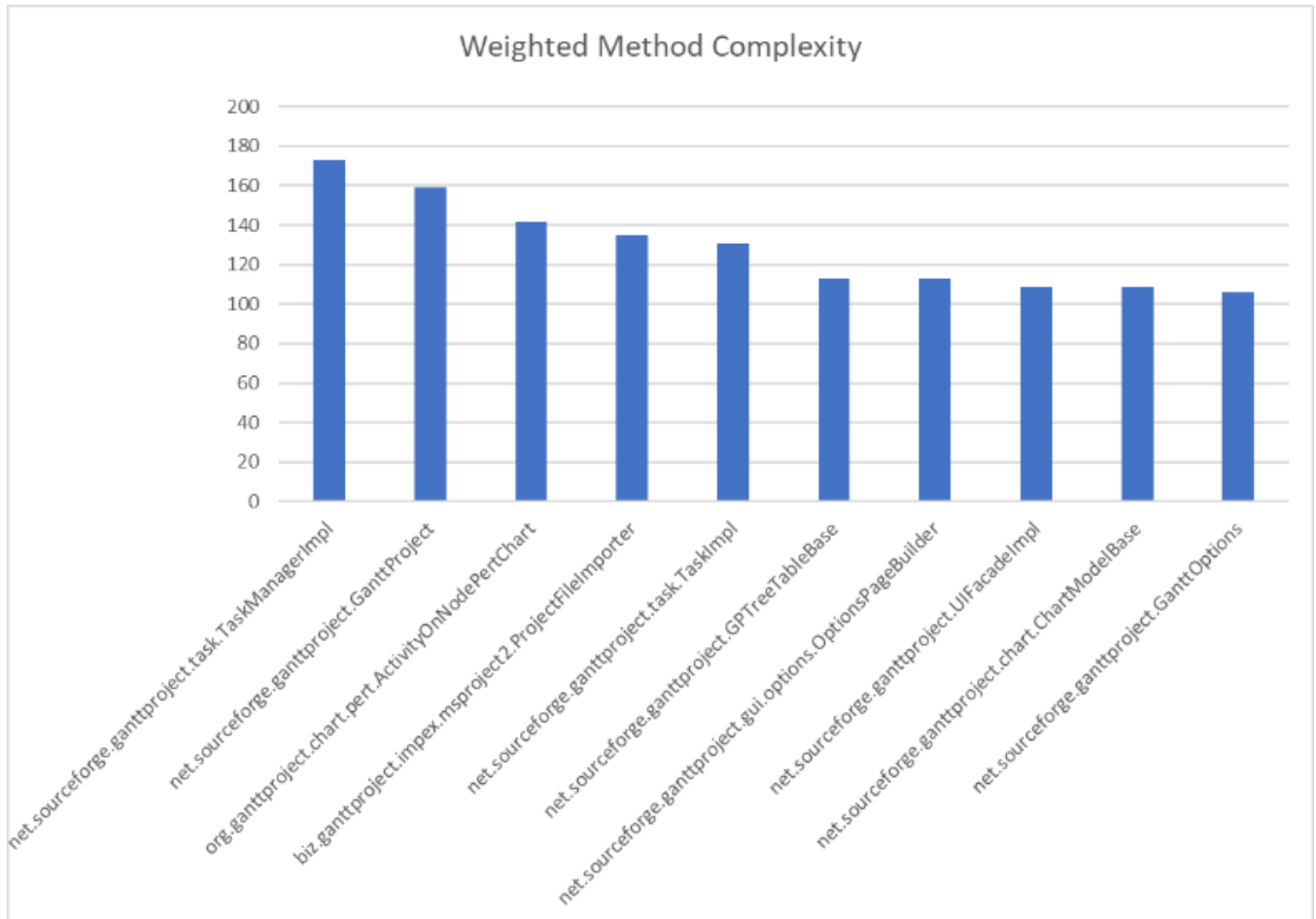
Esta métrica calcula a média da complexidade dos métodos não abstratos de cada classe. Como podemos ver pelo gráfico acima, a classe *GanttXMLOptionsParser* tem uma média muito superior às outras classes. Este fenómeno pode ser explicado pelo facto de haver um método (*public void startElement*) com bastantes chamadas *if* e *else if*, o que vai de logo aumentar muito a complexidade no pior caso, pois até o algoritmo chegar à condição que é satisfazível pode percorrer todas as verificações.

- **OCmax - Maximum Operation Complexity**



Esta métrica calcula a complexidade máxima de um método não abstrato de cada classe. De acordo com o gráfico podemos concluir que a classe *GanttXMLOptionsParser*, semelhante ao que acontece na métrica anterior, tem valores muito mais elevados do que as outras classes, devido às inúmeras condições *if* no método *public void startElement*.

- **WMC - Weighted Method Complexity**



Esta métrica calcula o valor total da complexidade de todos os métodos da classe. Neste caso, não há nenhuma classe que tenha valores extremos em comparação com as restantes, embora haja classes com valores ligeiramente maiores como a classe *TaskManagerImpl* e a classe *GanttProject*. Estes valores podem ser justificados devido à grande dimensão das classes, constituídas por muitos métodos. Podem chegar a esta conclusão pelo facto da média da complexidade dos métodos destas classes serem valores baixos, onde no caso das classes acima mencionadas os valores são, respetivamente, 1.99 e 1.61, e em que os valores da complexidade máxima por método é 4 e 2, respetivamente.

Dependency Metrics

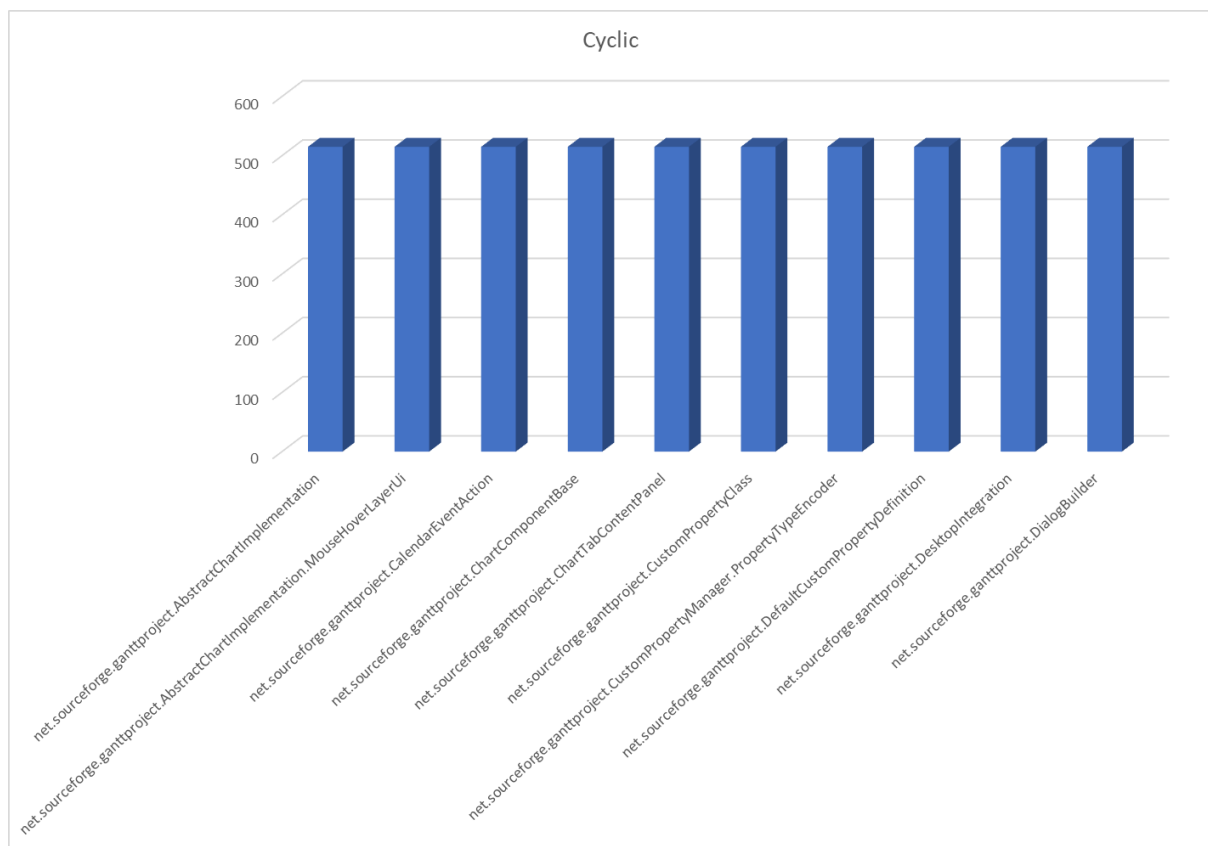
Elaborado por: Ricardo Bessa

Revisto por: Duarte Cruz

Uma versão mais completa das dependency metrics abaixo apresentadas e de algumas outras pode ser consultada aqui:

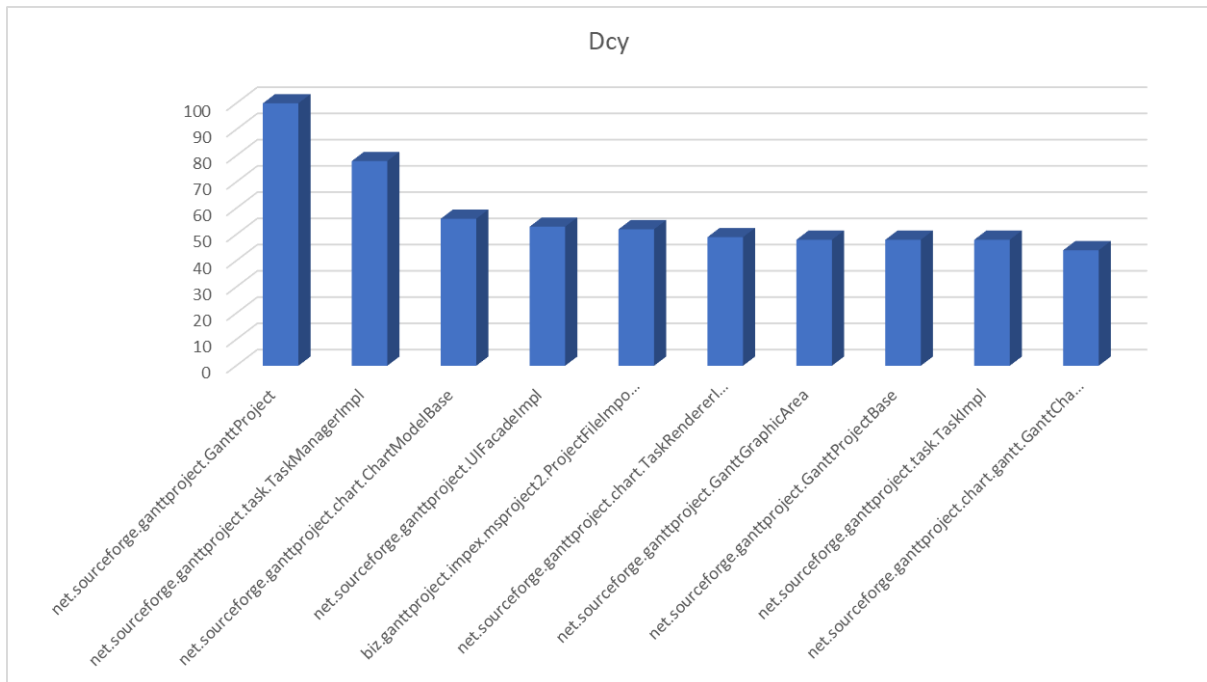
https://docs.google.com/spreadsheets/d/1iBibRX6mjh5Kuyi_VFQfaR6_uroM_WQQo8FwcCeCKD4/edit?usp=share_link

- **Cyclic: Number of Cyclic Dependencies**



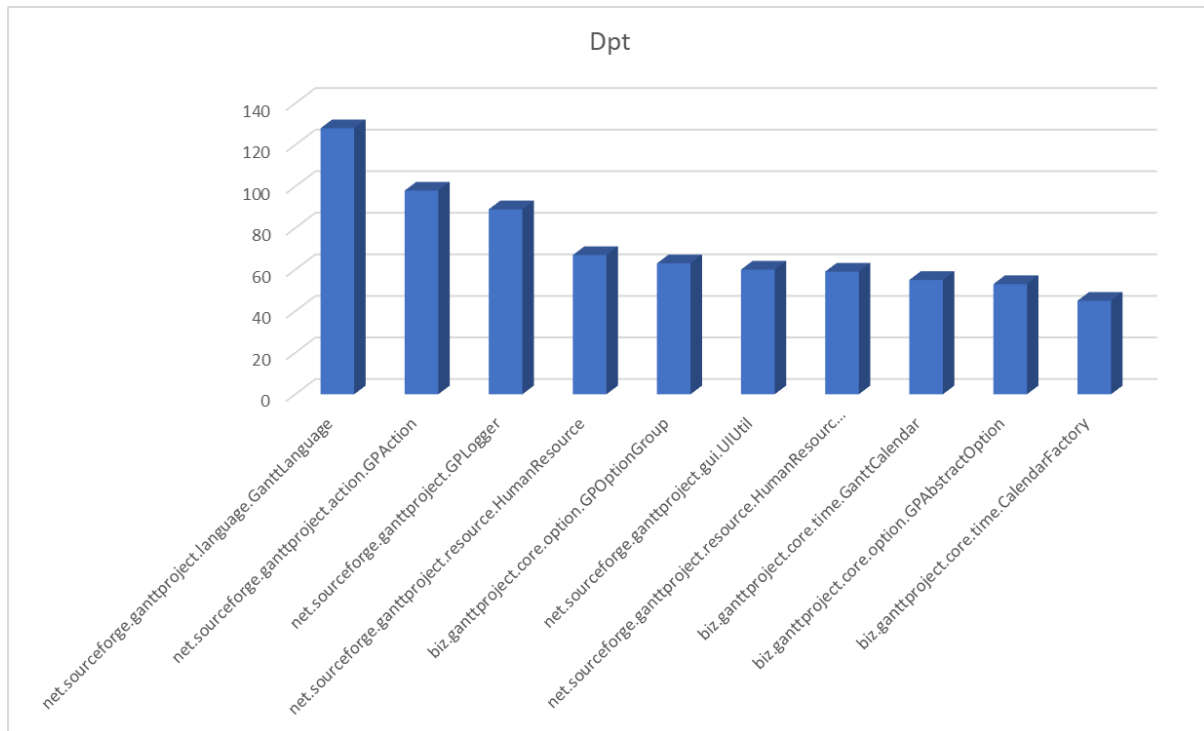
A análise desta métrica permitiu concluir que a classe que possui mais dependências cíclicas é a `AbstractChartImplementation`. Este tipo de dependências ocorre quando duas classes diferentes dependem uma da outra para funcionar corretamente. Neste caso, por exemplo, a classe `AbstractChartImplementation` tem uma instância da classe `ChartComponentBase` que é usada na implementação das suas funcionalidades logo depende dela. Por outro lado, esta classe `ChartComponentBase` declara um método abstrato `getImplementation()` que retorna uma instância da classe `AbstractChartImplementation`, logo vai também depender dela, existindo assim uma dependência cíclica entre estas duas classes.

- **Dcy: Number of Dependencies**



A análise desta métrica permitiu concluir que a classe GanttProject é a que depende de um maior número de outras classes, o que faz sentido tendo em conta que esta é uma das principais classes da aplicação e é essencial para o funcionamento da mesma, englobando alguns dos seus principais métodos. A classe TaskManagerImpl também depende de muitas classes pois é responsável por gerir as funcionalidades de todas as tasks existentes na ferramenta.

- **Dpt: Number of Dependants**



A análise desta métrica permitiu concluir que a classe GanttLanguage possui o maior número de classes que dependem dela, o que faz sentido tendo em conta que muitas classes necessitam de traduzir diferentes outputs de acordo com as diversas linguagens disponíveis na ferramenta. Outra classe da qual muitas classes dependem é a GPAction, visto que esta é utilizada sempre que o utilizador executa uma certa ação, por exemplo ao clicar num botão.

Lines of Code Metrics

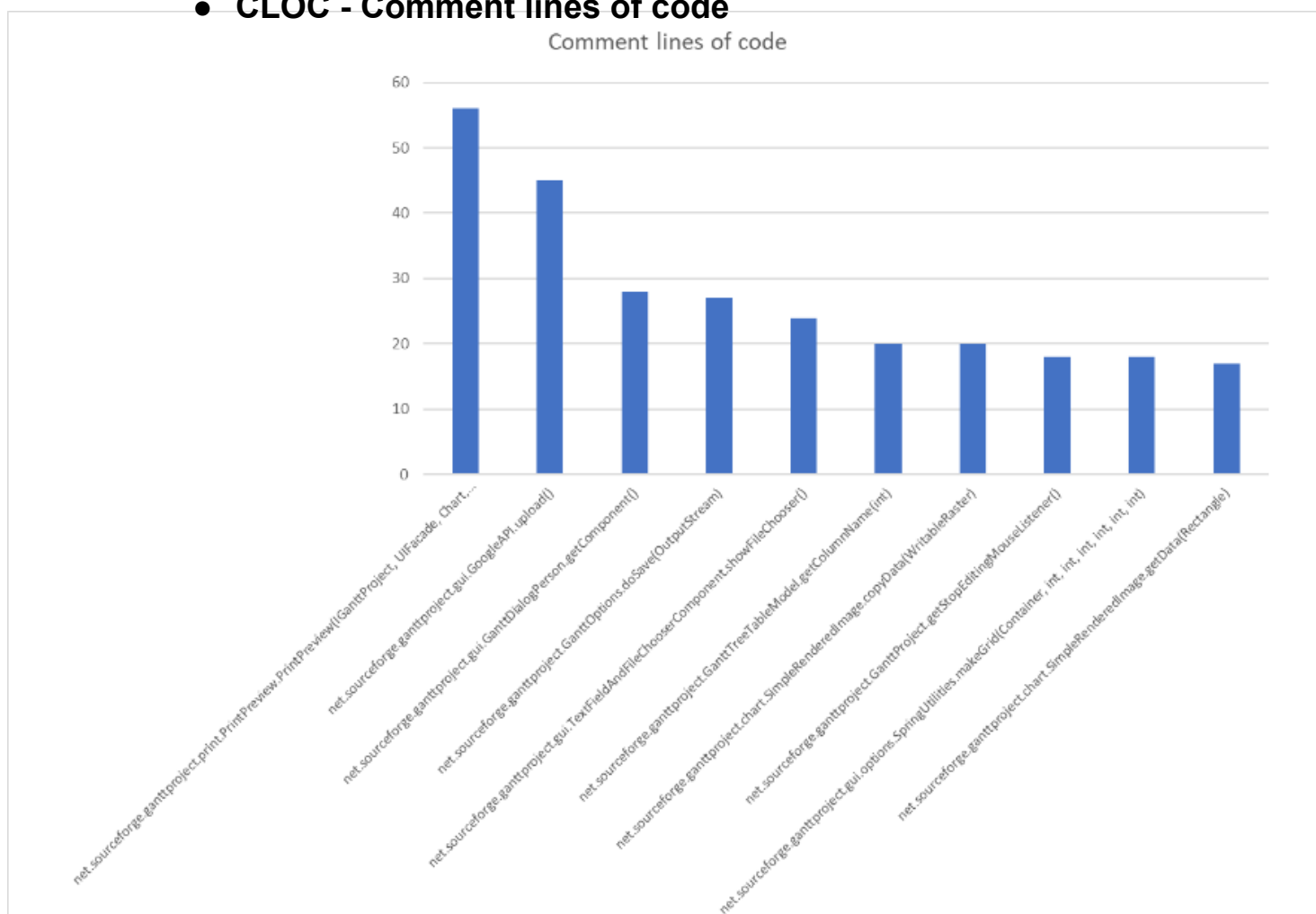
Elaborado por: João Pereira

Revisto por: Bruno Melo

Nesta análise foram apenas ponderadas as métricas mais relevantes a este projeto, tendo assim sido ignoradas para efeitos de análise a métrica NCLOC.

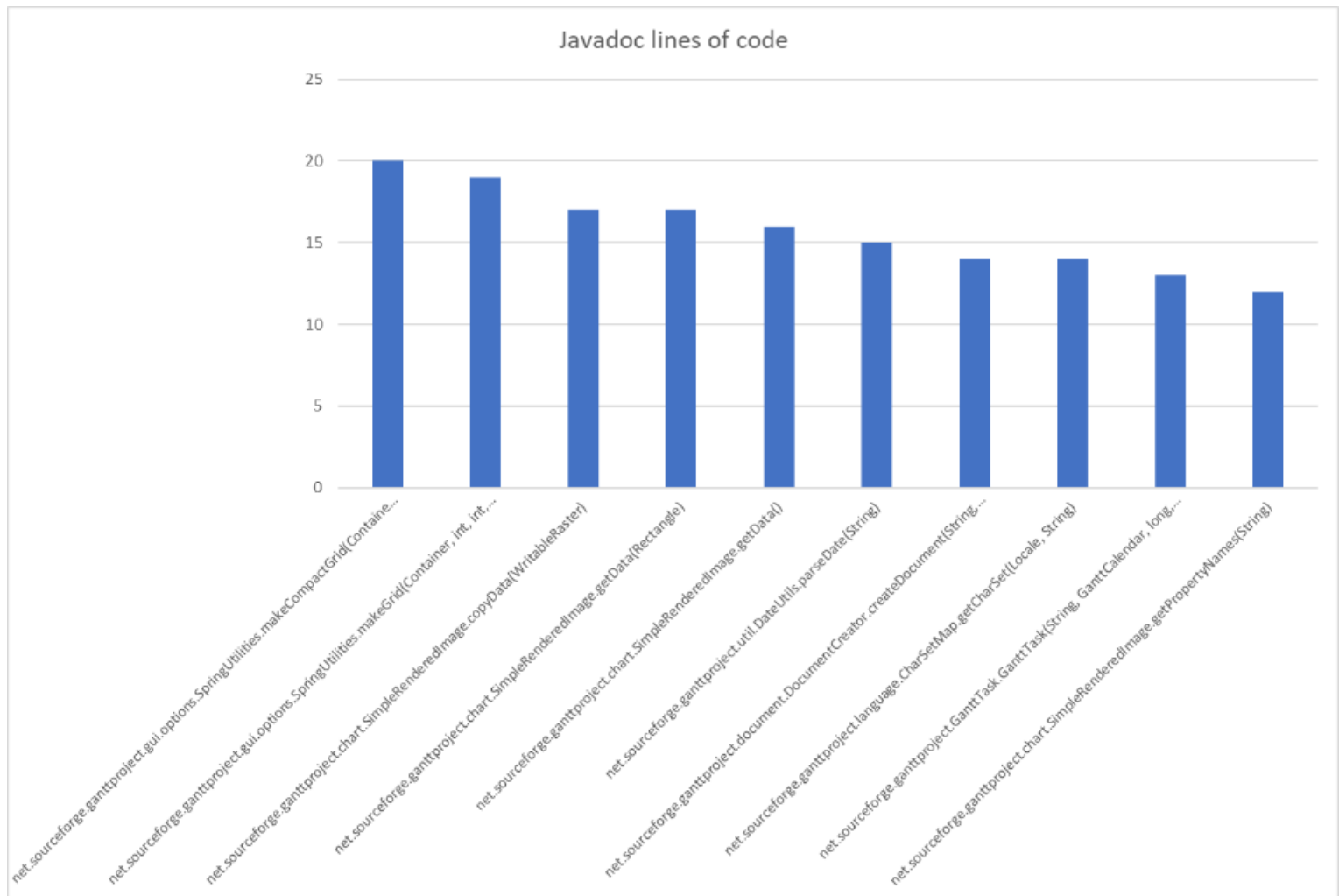
Para uma consulta mais detalhada consultar o seguinte link:
https://docs.google.com/spreadsheets/d/1Bcm_AnHlwnJZIJ4oiGKmawjb8AM1XKGP2C6RWdLHnU0/edit#gid=0

- **CLOC - Comment lines of code**



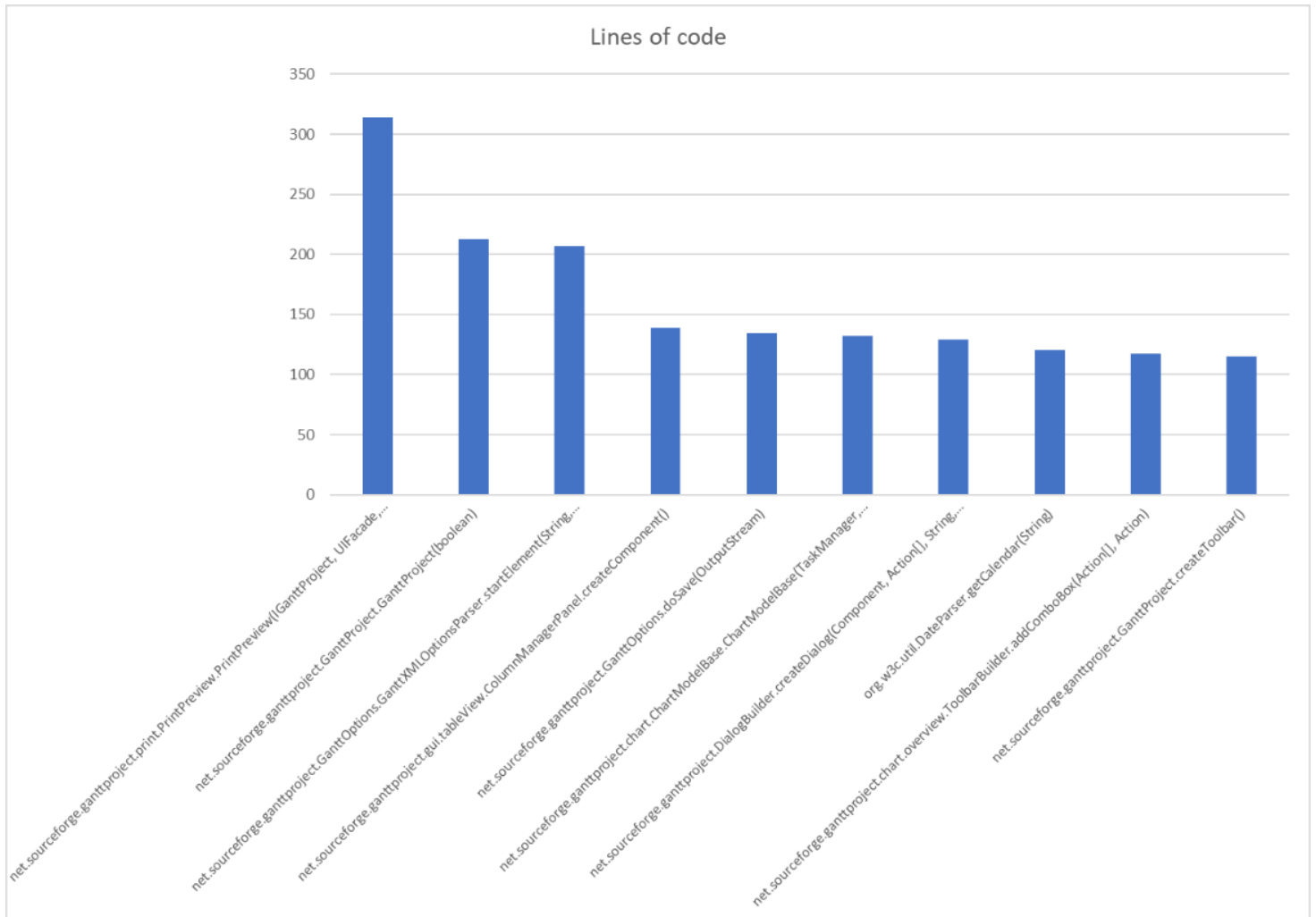
Esta métrica calcula o número de linhas comentadas em cada método do projeto. Os dois primeiros métodos têm valores elevados em comparação com os restantes métodos apresentados no gráfico. Isto pode ser justificado devido a esses métodos terem grandes partes de código comentado ao invés de serem comentários informativos sobre a função do método.

- **JLOC - Javadoc lines of code**



Esta métrica calcula o número de linhas ocupadas por comentários do tipo Javadoc em cada método. Ao analisarmos o gráfico, não há nenhum valor que se destaque dos outros, mas ao verificarmos os dois métodos com mais linhas podemos verificar que a maioria das linhas do comentário é referente a cada parâmetro do método, onde cada um desses dois métodos tem seis parâmetros. Com isto, podemos concluir que estes dois métodos têm um excesso de parâmetros, o que causa com que os comentários do Javadoc sejam longos.

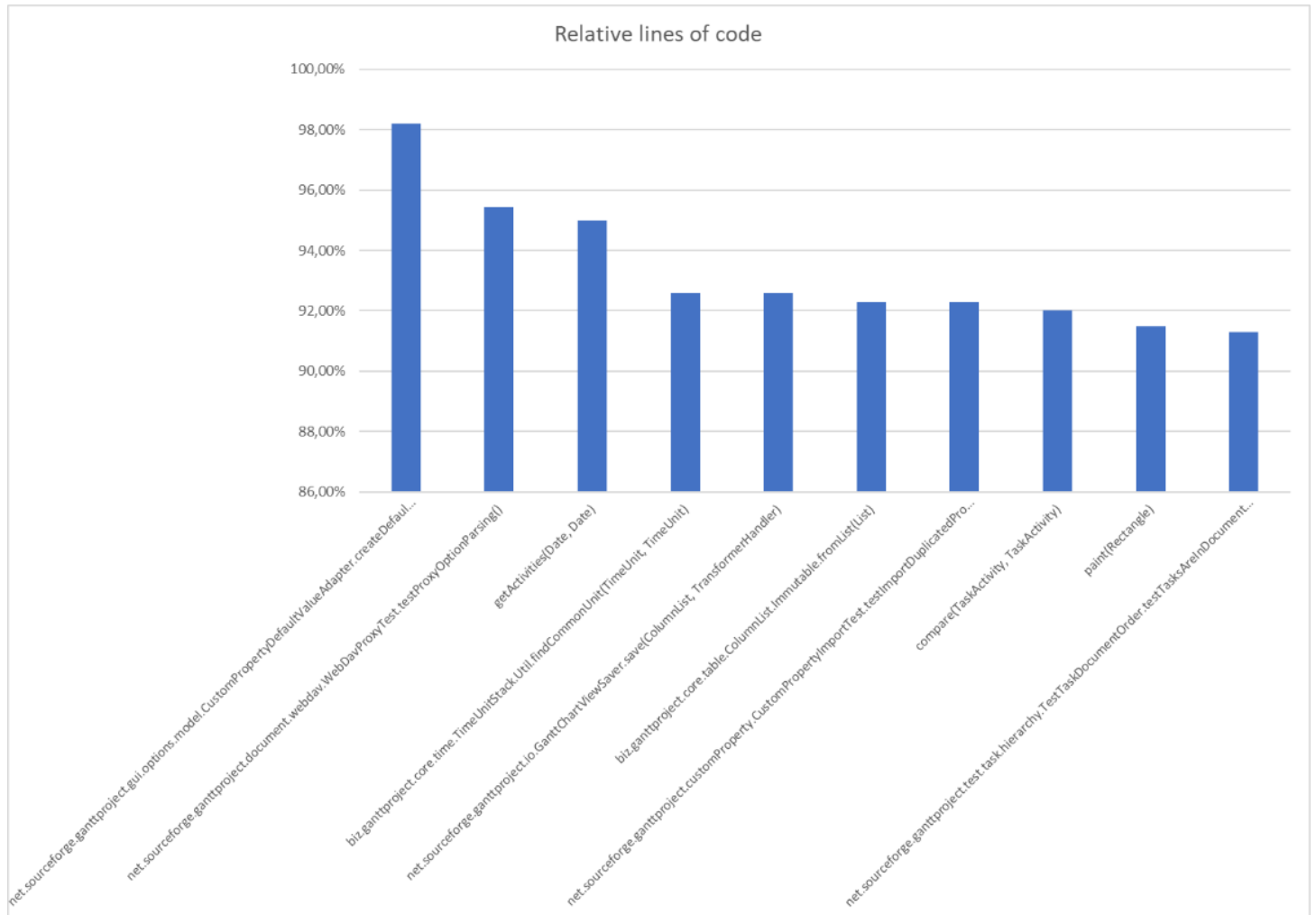
- **LOC - Lines of code**



Esta métrica calcula o número de linhas de cada método. Ao analisar o gráfico, podemos concluir que o método *PrintPreview* tem um valor muito elevado de linhas. Este método é responsável por gerar a janela de pré-visualização antes de imprimir o diagrama. Embora essa janela tenha muita informação a ser disposta como a própria pré-visualização do diagrama em folha, como opções de formatação da impressão.

Uma observação interessante é que o método *startElement* da classe *GanttXMLOptionsParser* aparece no gráfico na terceira posição, sendo dos métodos mais longos do projeto. Esta mesma classe é destaque nas *Complexity Metrics* por ser a classe com uma complexidade média mais elevada, em que o método que causa essa subida é o próprio *startElement*.

- **RLOC - Relative lines of code**



Esta métrica calcula a proporção entre as linhas de código do método em relação às linhas de código da classe onde o mesmo se encontra, ou seja, o valor é a percentagem de linhas de código da classe que pertencem ao método. Valores altos nesta métrica podem mostrar uma fraca abstração das classes que contém esses métodos, como por exemplo, ao analisar o gráfico, o método *createDefaultValueOption*, tendo um valor de 98,20%, mostra que grande parte da classe é ocupada pelo método e, ao verificar, a classe onde este método está inserido é apenas constituída por este método.

Martin Packaging Metrics

Elaborado por: Bruno Melo

Revisto por: Neel Badracim

Todas as métricas obtidas podem ser consultadas no ficheiro em anexo sobre a métrica Martin Packaging ou então em:

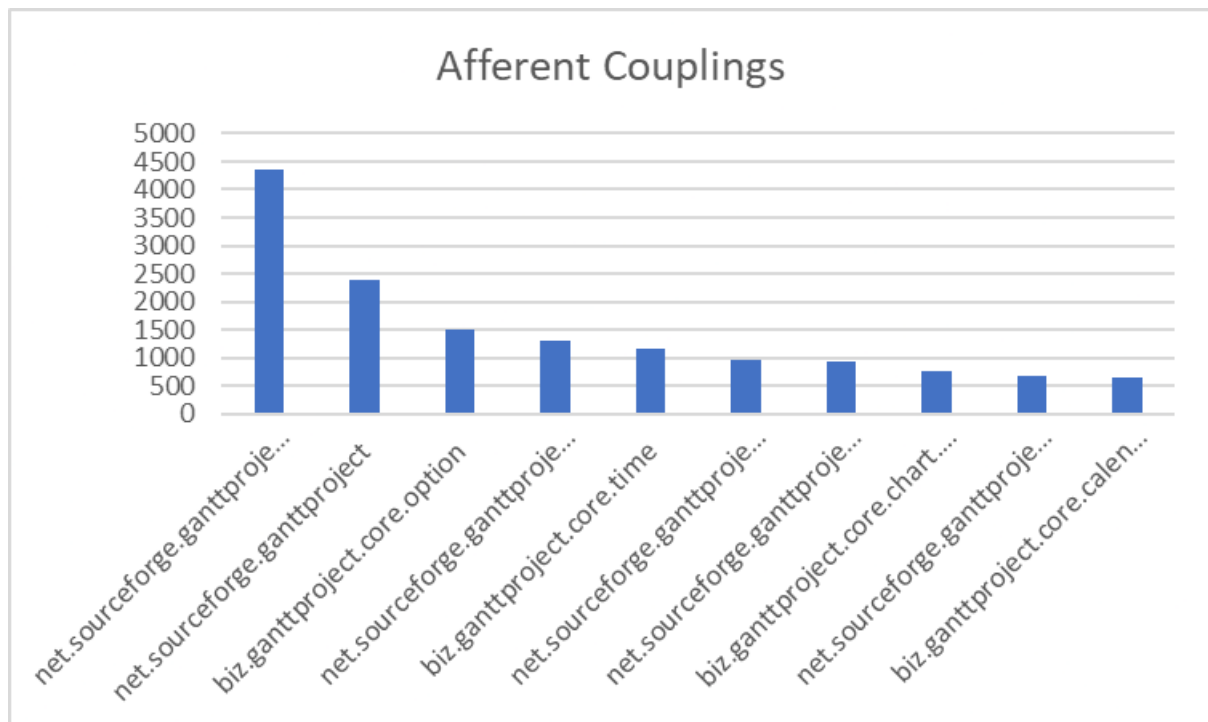
<https://docs.google.com/spreadsheets/d/1ldVYIDyGgQoXbje5IIQloZ4C3SYOVGFTbH6XznW9rk/edit?usp=sharing>

Significado das siglas:

A	Abstractness
AC	Afferent Couplings
EC	Efferent Couplings
D	Distance from the main sequence
I	Instability

Gráficos com os 10 valores mais elevados em cada métrica

AC - Afferent Couplings

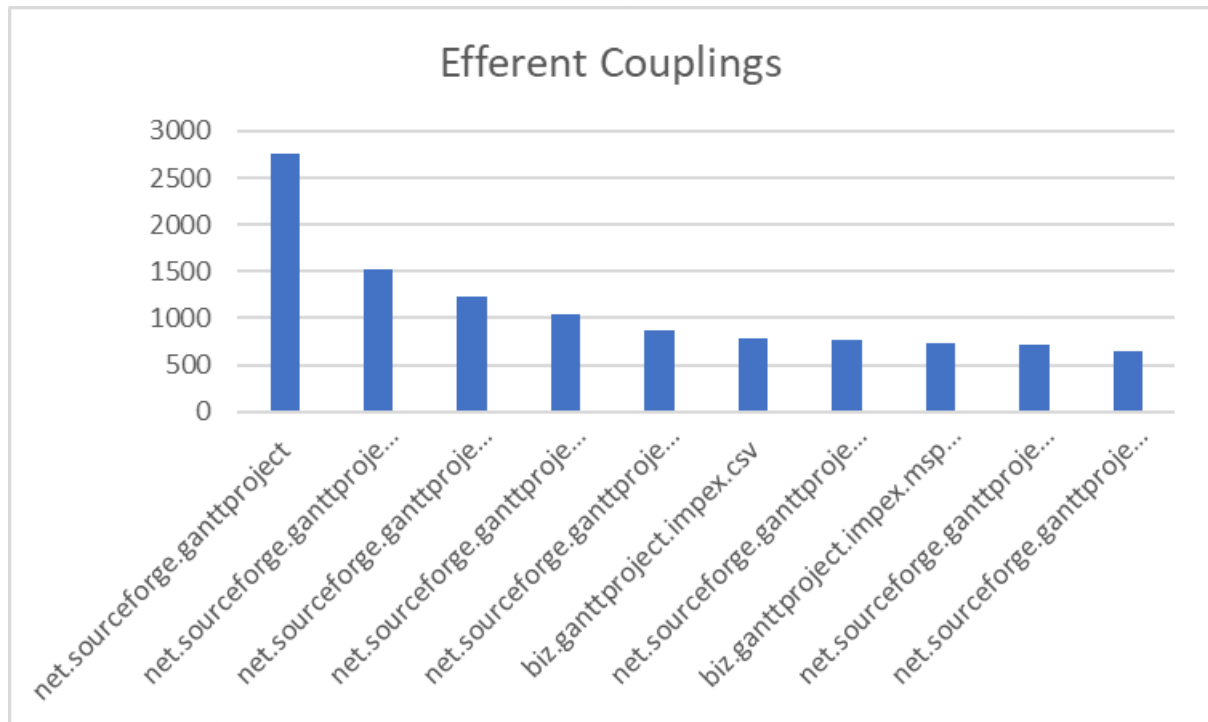


O número de classes em outros pacotes que dependem de classes dentro do pacote é um indicador da responsabilidade do pacote (afferent couplings).

Tendo em conta a natureza do projeto, não é de espantar que a o package net.sourceforge.ganttproject.task seja o pacote com maior taxa de AC, visto que todo o projeto depende da criação e gestão de tarefas, justificando-se assim a diferença

significativa do valor desta métrica para este package, comparativamente com os restantes packages.

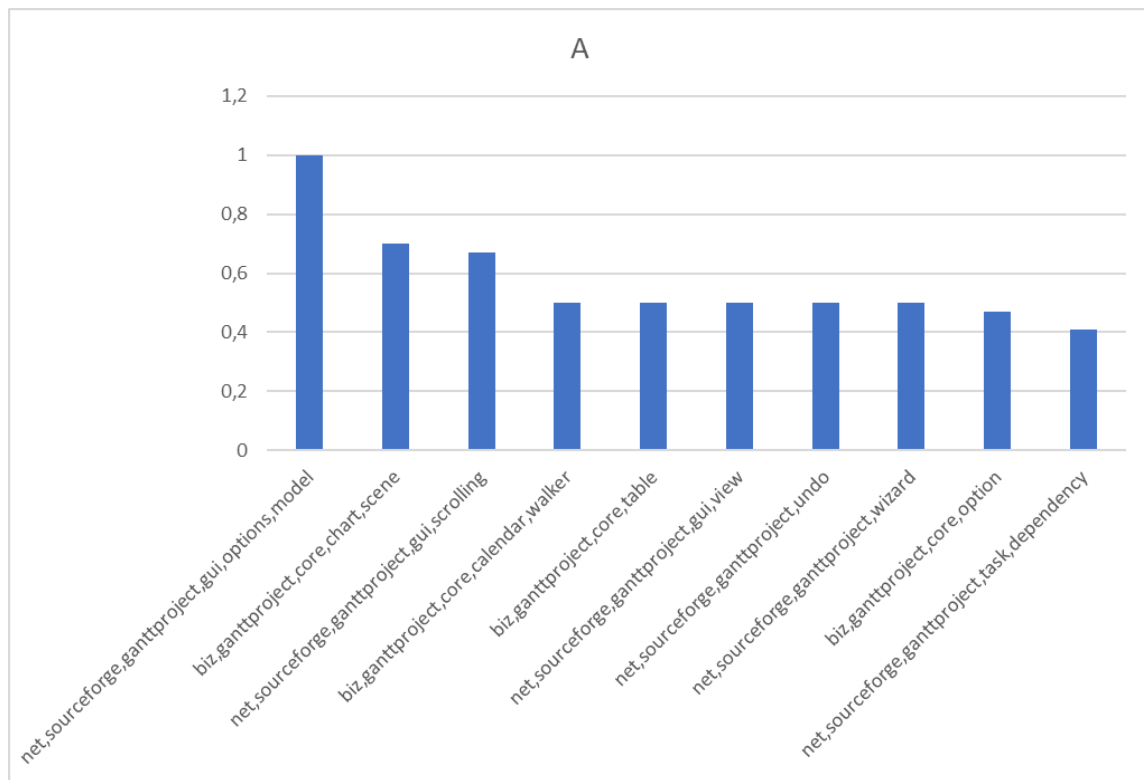
EC - Efferent Couplings



O número de classes em outros pacotes das quais as classes num dado pacote dependem é um indicador da dependência do pacote de externalidades (efferent couplings).

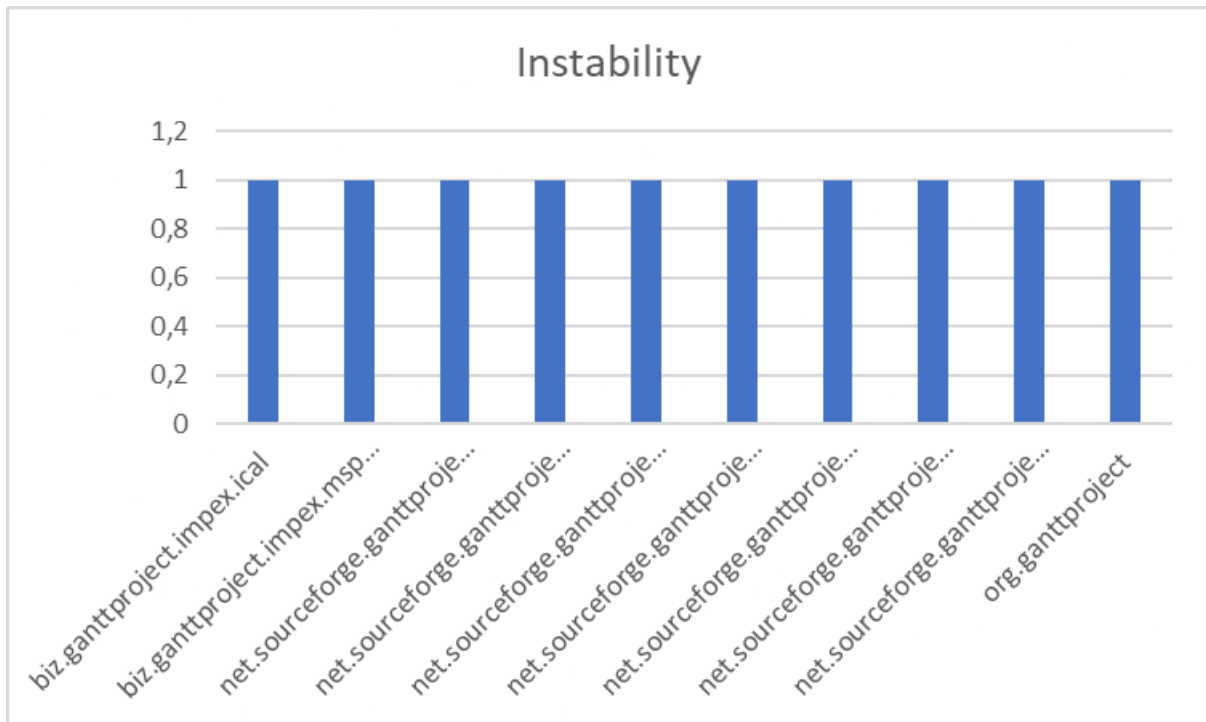
Como seria de esperar, o package `net.sourceforge.ganttproject` é o package com o maior valor de EC, dado que é um package geral que inclui grande parte das funcionalidades do Gantt Project. Como tal, existe um número imenso de classes neste pacote que dependem de classes em pacotes exteriores, de modo a concluir o seu propósito definido.

A - Abstractness



A proporção do número de classes abstratas (e interfaces) no pacote analisado para o número total de classes no pacote analisado. O intervalo dessa métrica é de 0 a 1, com A=0 indicando um pacote completamente concreto e A=1 indicando um pacote completamente abstrato.

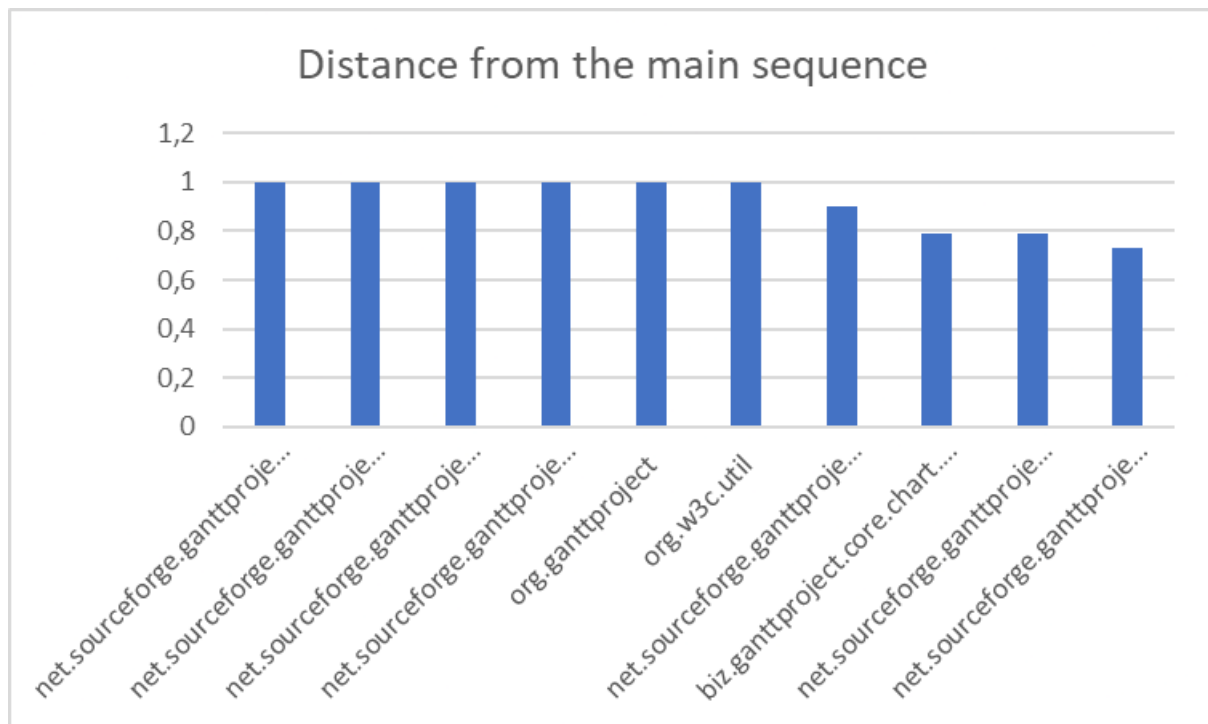
I - Instability



Essa métrica é um indicador da resiliência do pacote à mudança. O intervalo dessa métrica é de 0 a 1, com $I=0$ indicando um pacote completamente estável e $I=1$ indicando um pacote completamente instável.

Estas packages são instáveis porque dependem de packages exteriores. Sendo assim, alterações nesses packages exteriores implicam a necessidade de alterar o pacote dependente. Por outro lado, alterações nas packages instáveis não afetam os restantes packages.

D - Distance from the main sequence



Essa métrica é um indicador do equilíbrio do pacote entre abstração e estabilidade. Um pacote diretamente na sequência principal é otimamente equilibrado em relação à sua abstração e estabilidade. Pacotes ideais são completamente abstratos e estáveis ($I=0$, $A=1$) ou completamente concretos e instáveis ($I=1$, $A=0$). O intervalo dessa métrica é de 0 a 1, com $D=0$ indicando um pacote que coincide com a sequência principal e $D=1$ indicando um pacote o mais distante possível da sequência principal.

Todos os packages listados aqui têm taxas de I e A próximas de 0, refletindo-se numa taxa de D próxima ou igual a 1, visto que nenhum destes packages é completamente abstrato e estável ou completamente concreto e instável.

Potenciais problemas no código

Classes com um DIT elevado - Chidamber-Kemerer

As classes com um DIT elevado podem complicar a expansão futura da aplicação uma vez que caso se mude uma função na raiz da árvore teremos de garantir que todos os nós descendentes continuem a executar sem falhas. Estes problemas poderão surgir caso se tente adicionar mais funcionalidades às classes GanttTreeTable e ResourceTreeTable

Classes com um elevado número de dependências

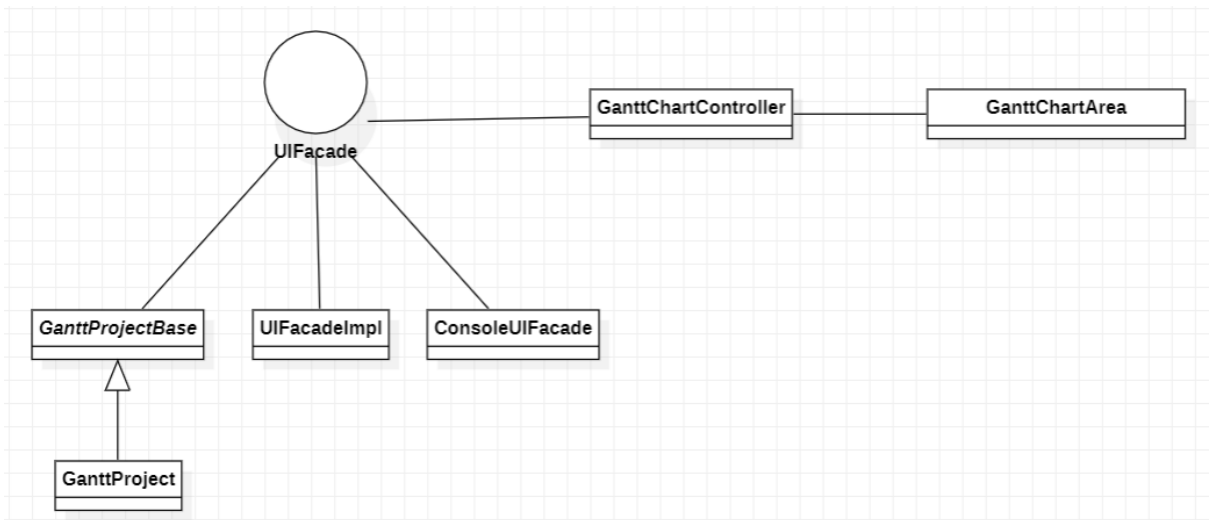
Classes que dependem de muitas classes poderão ser problemáticas visto que quando se introduz uma alteração nas classes da qual dependem estas poderão sofrer alterações inesperadas que possam comprometer as suas funcionalidades. O mesmo acontece com dependências cíclicas em que a alteração numa classe poderá afetar a outra da qual depende e vice-versa. Classes das quais dependam muitas outras classes também poderão ser problemáticas pois trata-se de um indicador de que se tratam de classes com uma responsabilidade muito elevada e alguma alteração nas mesmas poderá trazer consequências para as classes que dela dependam tal como acima mencionado.

Fraca abstração de classes - Lines of code metrics

Um método com um valor elevado na métrica RLOC pode ser sinal de uma fraca abstração por parte da classe que o contém, pois indica que há classes com poucos métodos para justificar a criação da própria classe.

Design Patterns identificadas

UIFacade (Structural Design Pattern) - Chidamber-Kemerer



Esta classe foi identificada na 1ª fase do projeto como sendo uma classe que iria “esconder” um subsistema mais complexo por de trás dela. Podemos verificar a partir da métrica RFC que as nossas suspeitas confirmam-se.

Chain of Responsibility Pattern(Behavioral) - Chidamber-Kemerer



A classe **GanttProject** foi identificada na 1ª fase do projeto como sendo uma classe que iria “delegar” bastante responsabilidades a outras classes. Como podemos observar a partir duma análise à métrica RFC, esta suspeita confirma-se. Inclusive podemos observar que é a classe que recebe mais informações de outras classes (RFC - 374). O mesmo se pode concluir a partir da análise das dependency metrics.

Novas implementações

Para as novas implementações que fizemos no projeto, resolvemos usar as API's da Google. Para tal foi necessário fazer bastante pesquisa visto que para a versão que estávamos a usar do gradle não havia praticamente nenhuma documentação existente.

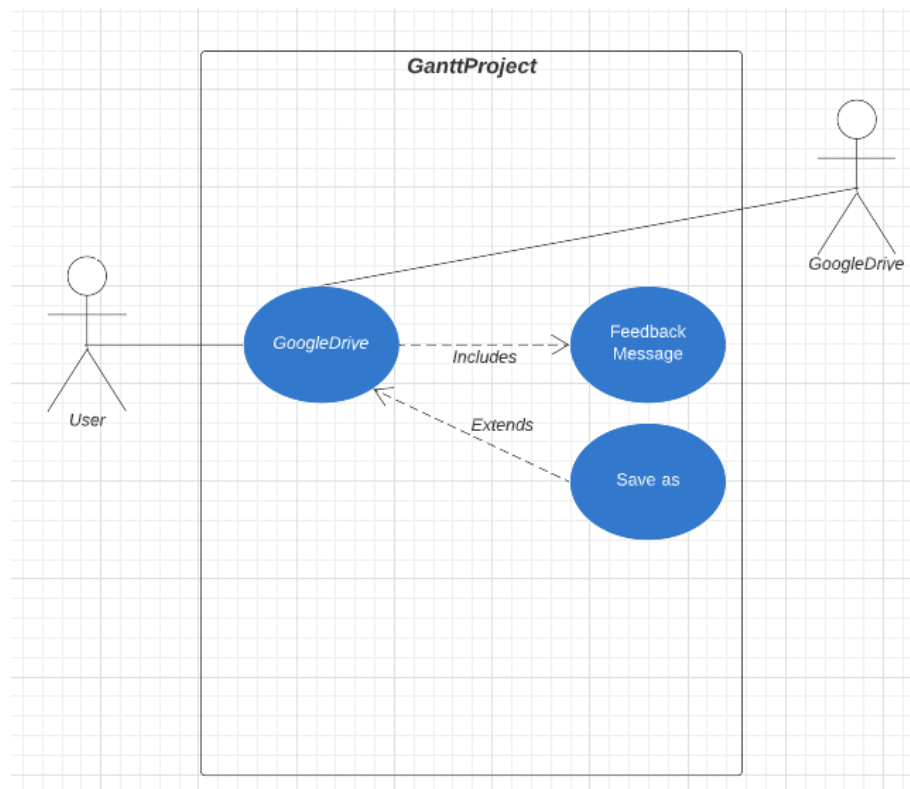
Com isto dito e feito, encontramos ainda diversas dificuldades nas importações das bibliotecas que para além de termos de usar bibliotecas de 2017, a forma como o gradle está configurado neste projeto torna a importação de bibliotecas externas num projeto bastante confuso e demorado.

Todos os links usados na nossa pesquisa podem ser encontrados na secção da webgrafia.

Funcionalidade 1 - Google Drive

User Story: Como utilizador do Ganttproject gostaria de ter a possibilidade de dar upload dos meus Gantt charts diretamente para o Google Drive, para poder acessá-los a partir de qualquer dispositivo.

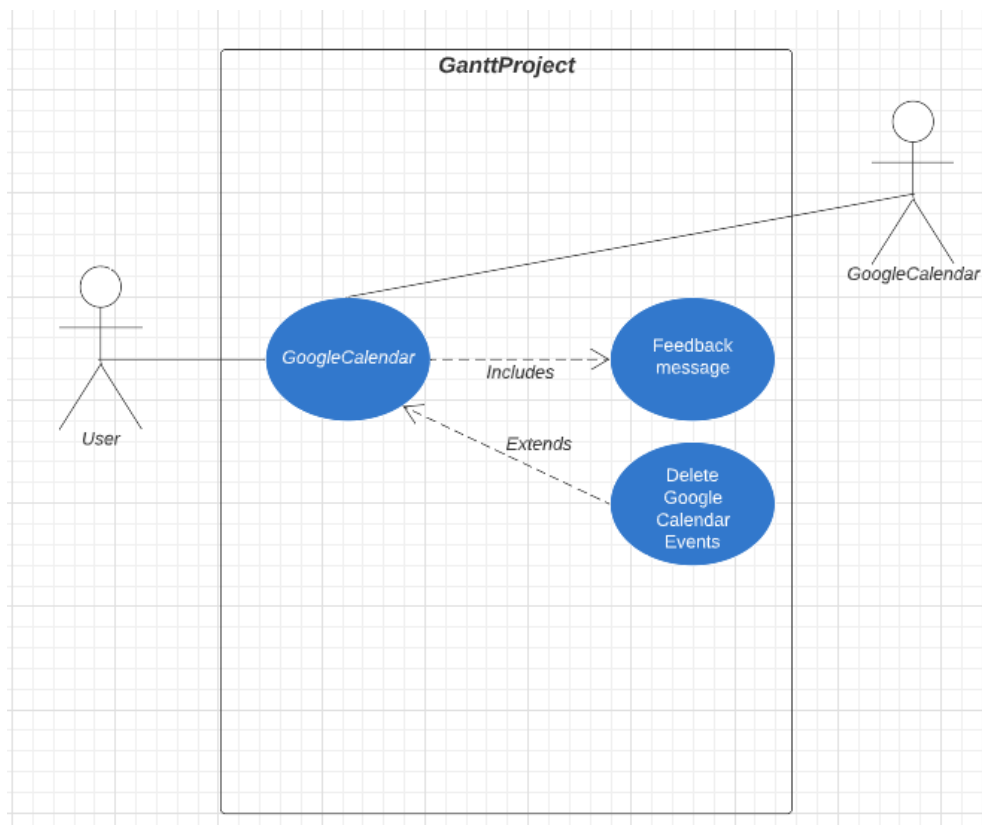
Use Case: O utilizador (ator principal) clica no botão “Upload to Google Drive” que se encontra disponível na janela "Projeto". O utilizador poderá guardar o ficheiro no seu computador caso ainda não o tenha feito. De seguida, o sistema externo da Google Drive (ator secundário) faz o upload do ficheiro selecionado para a Google Drive do utilizador, sendo depois mostrada uma mensagem de feedback que confirma ao utilizador se o upload foi realizado com sucesso ou se ocorreu algum erro (caso o utilizador tente dar upload de um ficheiro que não existe).



Funcionalidade 2 - Google Calendar

User Story: Como utilizador do Ganttproject gostaria de poder ser notificado das datas de início e fim das tarefas, assim como da sua taxa de completude através do Google Calendar, de forma a manter-me informado do progresso do projeto e das tarefas que o constituem.

Use case: O utilizador (ator principal) clica no botão “Google Calendar” que se encontra disponível na janela "Projeto". O sistema externo Google Calendar (ator secundário) faz o update do calendário do utilizador. Se o update é realizado com sucesso é mostrada uma mensagem correspondente. Caso o utilizador tenha removido uma *task* anteriormente adicionada ao seu calendário, quando este voltar a fazer o update esta será também removida do calendário.



Vídeo

<https://www.youtube.com/watch?v=LHV7sCs3BSA>

Conclusão

O grupo desenvolveu as funcionalidades propostas com sucesso, trabalhando em equipa para ultrapassar as adversidades que foram surgindo ao longo da implementação das mesmas. Foi realizado um trabalho de pesquisa no sentido de encontrar soluções para problemas relacionados sobretudo com versões desatualizadas ou incompatíveis das diversas bibliotecas utilizadas pelo GanttProject e das novas bibliotecas relacionadas com as APIs utilizadas. Desta forma, concluímos que o projeto foi desenvolvido com sucesso e a ferramenta foi modernizada, trazendo melhorias para a experiência de todos os utilizadores, e que todos os elementos do grupo mostraram esforço e empenho na realização deste projeto. Para além disso, foram consolidados os conhecimentos adquiridos ao longo do semestre sobre engenharia de software e também relacionados com a utilização de bibliotecas e APIs externas.

Anexos

Repositorio da 1ª Fase: https://github.com/MagoPT/ES2223_ganttproject

Repositorio da 2ª Fase: <https://github.com/joaopereira12/ganttproject>

1º Relatório:

https://drive.google.com/file/d/1zGi_NPUgSCMvvmg6NGvGfUEOccY7Imvw/view?usp=share_link

Use cases:

https://docs.google.com/document/d/16Lq4JsX9LKJYTsGeWHNZLvbmNWCoY3tcUKj8JC-V8RY/edit?usp=share_link

Métricas:

Chidamber-Kemerer:

https://docs.google.com/spreadsheets/d/1ldVYIDyGgQoXbjve5IIQloZ4C3SYOVGFTbH6XznW9rk/edit?usp=share_link

Complexity Metrics:

https://docs.google.com/spreadsheets/d/1Hkqu5N6TV-thTcfuX_8-HMI3Io_3BhFpifG9LqtgkGE/edit?usp=share_link

Dependency Metrics:

https://docs.google.com/spreadsheets/d/1iBlbRX6mjh5Kuyi_VFQfaR6_uroM_WQQo8FwcCeCKD4/edit?usp=share_link

Lines of Code Metrics:

https://docs.google.com/spreadsheets/d/1Bcm_AnHlwnJZIJ4oiGKmawjb8AM1XKGP2C6RWdLHnU0/edit?usp=share_link

Martin Packaging Metrics:

https://docs.google.com/spreadsheets/d/1tnyuKzk7UxmyaK_f2EPitFxsZkNT1ZkOPzdWDsfxpc8/edit?usp=sharing

Webgrafia

API:

Google Drive API: <https://developers.google.com/drive/api>

Google Calendar API: <https://developers.google.com/calendar/api>

Maven dependencies (utilizado par importar as bibliotecas da google):

google-api-client:

<https://mvnrepository.com/artifact/com.google.api-client/google-api-client>

google-api-services-drive:

<https://mvnrepository.com/artifact/com.google.apis/google-api-services-drive>

google-oauth-client-jetty:

<https://mvnrepository.com/artifact/com.google.oauth-client/google-oauth-client-jetty>

google-api-services-calendar:

<https://mvnrepository.com/artifact/com.google.apis/google-api-services-calendar>

Programas

Desenvolvimento de código:

- IntelliJ IDEA

Plugins utilizados:

- MetricsReloaded

Diagramas:

- StarUML

Elaboração de documentos:

- Microsoft Excel
- Google Docs
- Google Sheets