NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Software Engineering

2ª phase – 2022/2023

Written by:

Guilherme Poças 60236 p4

Samuel Pires 60718 p3

Leticia Silva 60221 p4

João Agostinho 57538 p5

Joana Tomás 60152 p4

Github: https://github.com/GuilhermePocas/ES_gantProject.git

Index

Introduction

In this project, our team set out to extend the already existing Ganttproject, with the intent to add two new functionalities, to make the application more useful to the user. Firstly, we identified some design patterns on the existing code, as well as some code smells, and after implementing our functionalities we also collected some metrics from the code and represented the system through use cases. Finally, we made a video explaining how to use the improved Ganttproject.

Code analysis

Code Patterns

Guilherme Poças 60236

1. Factory method pattern

```
public interface ActiveActionProvider {
  /** Provides the active action, which might depend on external
influences */
  public abstract AbstractAction getActiveAction();
}
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/action/ActiveActionProvider.java

This interface is used to create an Action object, but hides the creation code on the actual class, making the creation process independent of the concrete class.

Review:
The pattern and its explanation seem correct. -Leticia Silva 60221

2. Facade pattern

```
class UIFacadeImpl extends ProgressProvider implements UIFacade
```

Location: ganttproject/src/main/java/net/sourceforge/ganttproject/UIFacadeImpl.java

This class allows other parts of the program to access functionalities and methods of other classes responsible for the UI, wrapping them in a subsystem, and making it easier for the user to use them.

Review:
Pattern well identified and explained code pattern - Samuel Pires 60718

3. Template method pattern

```
public abstract class ImporterBase implements Importer
public class ImporterFromGanttFile extends ImporterBase
public class ImporterFromTxtFile extends ImporterBase
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/importer/ImporterBase.java

The two importer classes import from different types of files, so have different responsibilities, but a similar function. Because of this, they defer some similar methods to the ImporterBase abstract class, making the code cleaner and avoiding repeated code.

Review:
The Template method pattern is well identified. - Joana Tomás 60152

Samuel Pires 60718

1. Command Pattern

```
public class ZoomActionSet {
 private final ZoomInAction myZoomIn;

 private final ZoomOutAction myZoomOut;

public ZoomActionSet(ZoomManager zoomManager) {
    myZoomIn = new ZoomInAction(zoomManager);
    myZoomIn.putValue(Action.SHORT_DESCRIPTION, null);
    myZoomOut = new ZoomOutAction(zoomManager);
    myZoomOut.putValue(Action.SHORT_DESCRIPTION, null);
  }

 public GPAction getZoomInAction() {
    return myZoomIn;
  }

public GPAction getZoomOutAction() {
    return myZoomOut;
  }
}
```

Location:
ganttproject\src\main\java\net\sourceforge\ganttproject\action\zoom\ZoomActionSet.java

The ZoomActionSet class is used as a Command Pattern in this program. This class manages the concrete commands zoom in (represented by the ZoomInAction class) and zoom out (represented by the ZoomOutAction class).

Review:
Both the identification and justification of the pattern make sense - Guilherme Poças 60236

2. Singleton Pattern

```
private static final GanttLanguage ganttLanguage = new GanttLanguage();
private GanttLanguage() {…}
public static GanttLanguage getInstance() {return ganttLanguage;}
```

Location:
ganttproject\src\main\java\net\sourceforge\ganttproject\language\GanttLanguage.java

The GanttLanguage class only has one instance created at all times, providing a way to access it through the getInstance() method. It creates the instance at initialization and has a private contructor, making it impossible to create another instance.

Review:
Correctly recognized pattern with a good explanation - João Agostinho 57538

3. Facade Pattern

```
    // Geometry of the window
    addAttribute("x", "" + x, attrs);
    addAttribute("y", "" + y, attrs);
    addAttribute("width", "" + width, attrs);
    addAttribute("height", "" + height, attrs);
    addAttribute("maximized", String.valueOf(this.isMaximized), attrs);
    emptyElement("geometry", attrs, handler);

    // ToolBar position
    addAttribute("position", "" + toolBarPosition, attrs);
    addAttribute("icon-size", "" + iconSize, attrs);
    addAttribute("show", "" + buttonsshow, attrs);
    emptyElement("toolBar", attrs, handler);
    addAttribute("show", "" + bShowStatusBar, attrs);
    emptyElement("statusBar", attrs, handler);
```

[…]

Location:
ganttproject\src\main\java\net\sourceforge\ganttproject\GanttOptions.java

The GanttOptions class is used as a Facade Pattern in this program. This class acts as a middleman between the program and the options' classes, hiding these and their complexity.

Review:
This pattern is well identified, and the justification is correct - Joana Tomás 60152

Leticia Silva 60221

1. Proxy Pattern

```
public class ReadOnlyProxyDocument implements Document {

  private final Document myDelegate;

  public ReadOnlyProxyDocument(Document delegate) {
    myDelegate = delegate;
  }

  @Override
  public String getFileName() {
    return myDelegate.getFileName();
  }

  @Override
  public boolean canRead() {
    return myDelegate.canRead();
  }
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/document/ReadOnlyProxyDocument.java

This proxy class is something that acts as a simplified version of the original object, providing controlled access of a functionality.

Review:
The identified pattern is correct - Guilherme Poças 60236

2. Template Pattern

```
abstract class SearchServiceBase<SR extends SearchResult<SO>, SO>
implements SearchService<SR, SO>
```

```
public class ResourceSearchService extends
SearchServiceBase<ResourceSearchService.MySearchResult, HumanResource>
```

```
public class TaskSearchService extends
SearchServiceBase<TaskSearchService.MySearchResult, Task>
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/search/ResourceSearchService.java

The two classes differ on the responsibilities but have very similar functionality and order of operations, so the abstract class SearchServiceBase makes the code understandable.

Review:
Well identified and explained code pattern - Samuel Pires 60718

3. Factory Pattern

```
public interface ParserFactory {
  GPParser newParser();

  GPSaver newSaver();
}
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/parser/ParserFactory.java

The interface ParserFactory hides the creation code of the objects GPParser and GPSaver, so the creation process does not depend on concrete classes.

Review:
This pattern is well identified - Joana Tomás 60152

João Agostinho 57538

1. Singleton Pattern

```java
public class OffsetManager {
  private static OffsetManager offsetManagerInstance;

  private OffsetManager() {
  }

  public static OffsetManager getInstance() {
    if (offsetManagerInstance == null) {
      offsetManagerInstance = new OffsetManager();
    }
    return offsetManagerInstance;
  }
```

Location:
biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetManager.java

This implementation makes the default constructor private (OffsetManager()) and has a public static method (getInstance()) that acts as the constructor by calling the private constructor to instantiate the class (OffsetManager).

Review:
Pattern well identified and explained - Samuel Pires 60718

2. Command Pattern

```java
class BooleanOptionAction extends AbstractAction {
    private BooleanOption myOption;

    BooleanOptionAction(BooleanOption option) {
        super("");
        this.myOption = option;
    }

    public void actionPerformed(ActionEvent e) { this.myOption.toggle(); }
}
```

Location:
ganttproject/build/classes/java/main/net/sourceforge/ganttproject/gui/options/BooleanOptionAction.class

This class is a command pattern in this program as it is used to toggle a BooleanOption when an action is performed.

Review:
Pattern well identified, but could specify which object is the invoker. -Leticia Silva 60221

3. Template method pattern

```
public abstract class GPAction extends AbstractAction implements Listener, EventHandler<ActionEvent> {
```

Location:
ganttproject/build/classes/java/main/net/sourceforge/ganttproject/action/GPAction.class

```
protected String getIconFilePrefix() { return null; }
```

This method is used as a template for other classses such as ExitAction.

```
class ExitAction extends GPAction {
    private final GanttProject myMainFrame;

    ExitAction(GanttProject mainFrame) {
        super( name: "quit");
        this.myMainFrame = mainFrame;
    }

    protected String getIconFilePrefix() { return "exit_"; }
```

Location:
ganttproject/build/classes/java/main/net/sourceforge/ganttproject/action/project/ExitAction.class

Review:
The pattern is well identified, but the template for the other action classes is the GPAction class itself, not the getIconFilePrefix method - Guilherme Poças 60236

Joana Tomás 60152

1. Visitor Pattern

```
public abstract class TaskVisitor {
  public String visit(TaskManager taskManager) throws Exception
```

Location: ganttproject/src/main/java/net/sourceforge/ganttproject/export/TaskVisitor.java

This class is an example of a visitor pattern because the original object is now passed to one of the visitor's methods as an argument, providing the method access to all necessary data contained within the object.

Review:
This pattern is well identified. - Leticia Silva 60221

2. Chain of Responsibility Pattern

```
public abstract class AbstractTagHandler implements TagHandler
public class TaskDisplayColumnsTagHandler extends AbstractTagHandler
public class RoleTagHandler  extends AbstractTagHandler
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/parser/AbstractTagHandler.java

This pattern lets us to pass requests along a chain of handlers. Upon receiving a request, each handler decides either to process the request or to pass it to the next handler in the chain.

Review:
Pattern is rightly pointed, and the rationalization is logical - João Agostinho 57538

3. Facade Pattern

```
public class TimelineFacadeImpl implements MouseInteraction.TimelineFacade
```

Location:
ganttproject\src\main\java\net\sourceforge\ganttproject\chart\mouse\ChangeTaskProgressInteraction.java

This class is responsible for the timeline of the program and encapsulates multiple classes in order to allow outside classes to use their methods, simplifying the code.

Review:
Pattern's justification is weirdly worded - Guilherme Poças 60236

Code Smells

Guilherme Poças 60236

1. Dead Code

```
private final Action[] myDelegates;
myDelegates = delegates;
```

Location: ganttproject\src\main\java\net\sourceforge\ganttproject\action\ArtefactAction.java

The variable myDelegates stores an array of Action type objects, but it is never read from, only assigned values, so it serves no purpose and only clutters the code.

Solution: delete the variable.

Review:
Code Smell and it's explanation are correct and the solution is proper. -Leticia Silva 60221

2. Speculative generality

```
public static List<MutableTreeTableNode>
breadthFirstSearch(MutableTreeTableNode rootNode) {
  final List<MutableTreeTableNode> result = Lists.newArrayList();
  breadthFirstSearch(rootNode, new
Predicate<Pair<MutableTreeTableNode,MutableTreeTableNode>>() {
    public boolean apply(Pair<MutableTreeTableNode, MutableTreeTableNode>
parent_child) {
      result.add(parent_child.second());
      return true;
    }
  });
  return result;
}
```

Location: ganttproject/src/main/java/net/sourceforge/ganttproject/TreeUtil.java

This method for Tree breadth first searching is never used, so it’s not currently needed, and it doesn’t help the code.

Solution: if there is no use for the method, delete it.

Review:
Code Smell well identified code smell with correct and understanding explanation and solution. - Samuel Pires 60718

3. Data Class

```
public class Pair<F, S>
private final F myFirst;
private final S mySecond;
```

Location: ganttproject/src/main/java/net/sourceforge/ganttproject/util/collect/Pair.java

This class serves only to save two values of different types, and only has two getter methods, and one creation method, so it has no real functionality or necessity.

Solution: Instead of using this class, utilize two different variables, or if they're the same type, an array.

Review:
Code Smell is correct and the solution is appropriate for solving the problem. - Joana Tomás 60152

Samuel Pires 60718

1. Dead Code

```
public interface TaskInfo {

}
```

Location: ganttproject\src\main\java\net\sourceforge\ganttproject\task\TaskInfo.java

The TaskIngo class is completely empty, with no variables, methods or any purpose. It's not used anywhere in the program.

Solution: Delete the class

Review:
Code Smell, Dead Code, is correct and the solution is apropriated. -Leticia Silva 60221

2. Refused Request

```
  @Override
  public void setVisibleTasks(List<Task> visibleTasks) {
    // TODO Auto-generated method stub
  }
```

Location:
ganttproject\src\main\java\net\sourceforge\ganttproject\chart\ChartModelResource.java

The ChartModelResource class inherits the method setVisibleTasks from the ChartModel interface which it implements but does not implement it.

Solution: Define this method, not in the interface, but in the other only subclass that implements it (GantModelImpl).

Review:
The smell is adequately identified and justified - Guilherme Poças 60236

3. Message Chain

```
protected String getDefaultFileName() {
Document document =
myWizard.getUIFacade().getGanttChart().getProject().getDocument();
    if (document == null) {
      return "document.gan";
    }
    return document.getFileName();
  }
```

Location:
ganttproject\src\main\java\net\sourceforge\ganttproject\gui\FileChooserPageBase.java

This method contains a considerably long message chain, calling methods that return an object and repeating this process for that object multiple times in the same line.

Solution: Create a method in this class the returns the wanted resource. In case that is not possible, make the function calls inside of each object in order not to have a message chain in the same line.

Review:
Well identified code smell, with a good explanation and solution - João Agostinho 57538

Leticia Silva 60221

1. Long method

```
private Document createDocument(String path, File relativePathRoot, String
user, String pass) {
  assert path != null;
  path = path.trim();
  String lowerPath = path.toLowerCase();
  if (lowerPath.startsWith("http://") || lowerPath.startsWith("https://")) {
    try {
      if (user == null && pass == null) {
        WebDavServerDescriptor server = myWebDavStorage.findServer(path);
        if (server != null) {
          user = server.getUsername();
          pass = server.getPassword();
        }
      }
      return new HttpDocument(path, user, pass,
myWebDavStorage.getProxyOption());
    } catch (IOException e) {
      GPLogger.log(e);
      return null;
    } catch (WebDavException e) {
      GPLogger.log(e);
      return null;
    }
  } else if (lowerPath.startsWith("cloud://")) {
    var patchedUrl = DocumentKt.asDocumentUrl(lowerPath);
    if (patchedUrl.component2().equals("cloud")) {
      return new GPCloudDocument(
        null,

DocumentUri.LocalDocument.createPath(patchedUrl.component1().getPath()).getPa
rent().getFileName(),
        patchedUrl.component1().getHost(),

DocumentUri.LocalDocument.createPath(patchedUrl.component1().getPath()).getFi
leName(),
        null
      );
    }
  } else if (lowerPath.startsWith("ftp:")) {
    return new FtpDocument(path, myFtpUserOption, myFtpPasswordOption);
  } else if (!lowerPath.startsWith("file://") && path.contains("://")) {
    // Generate error for unknown protocol
    throw new RuntimeException("Unknown protocol: " + path.substring(0,
path.indexOf("://")));
  }
  File file = new File(path);
  if (file.toPath().isAbsolute()) {
    return new FileDocument(file);
  }
  File relativeFile = new File(relativePathRoot, path);
  return new FileDocument(relativeFile);
}
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/document/DocumentCreator.java

This method is too long which makes it more difficult to comprehend, is doing more than one thing and produces more code smells.

Solution: Create new methods with fragments of the code.

Review:
The identified smell and it's solution are correct - Guilherme Poças 60236

2. Dead Code

```
public class PreferenceServiceImpl implements IPreferencesService {

  @Override
  public IStatus applyPreferences(IExportedPreferences preferences)
throws CoreException {
    // TODO Auto-generated method stub
    return null;
  }

  @Override
  public void applyPreferences(IEclipsePreferences node,
IPreferenceFilter[] filters) throws CoreException {
    // TODO Auto-generated method stub

  }
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/PreferenceServiceImpl.java

This class, PreferenceServiceUmpl , isn't used, has only the methods inherited from the interfaceIPreferencesService but they aren't implemented and doesn't have variables.

Solution: Delete this class.

Review:
Well identified code smell with satisfactory explanation and solution - Samuel Pires 60718

3. Comments

```
private static String connectStringArray(String[] a) {
  if (a == null)
    return null;

  String s = "";
  for (int i = 0; i < a.length; i++) {
    if (i > 0)
      s += " ";
    s += a[i];
  }

  return s;
}
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/util/BrowserControl.java

This method has no comments, and so it is hard to understand what the code is doing.

Solution: Add comments to the method to clarify what the code should do.

Review:
Code smell, and solution are correct - Joana Tomás 60152

João Agostinho 57538

1. Dead Code

```
public void setFirstText(String text) {
}

public void setSecondText(String text) {
}

public void setFirstText(String text, int milliseconds) {
}

public void setSecondText(String text, int milliseconds) {
}
```

Location:
ganttproject/build/classes/java/main/net/sourceforge/ganttproject/gui/GanttStatusBar.java

These methods are completely empty and have no use for the program.

Solution: Removing these methods.

Review:
Code Smell well spotted code with good explanation and solution - Samuel Pires 60718

2. Data Class

```
public class CalendarChartItem extends ChartItem {
    private final Date myDate;

    public CalendarChartItem(Date date) {
        super((Task)null);
        this.myDate = date;
    }

    public Date getDate() { return this.myDate; }
}
```

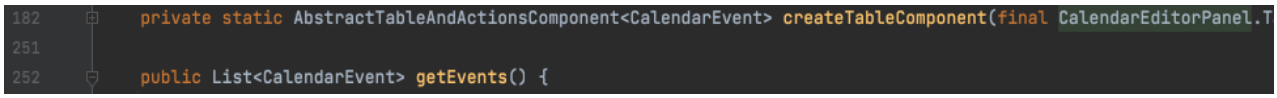
Location:
ganttproject/build/classes/java/main/net/sourceforge/ganttproject/chart/item/CalendarChartItem.class

This class is only used to store a date and only has one method to return it.

Solution: Instead of a class, use an array to store this data.

Review:
Code Smell is well identified, and the solution seems correct. -Leticia Silva 60221

3. Long Method



Location:
ganttproject/build/classes/java/main/net/sourceforge/ganttproject/calendar/CalendarEditorPanel.class

This method is very extensive (more than 60 lines), making it hard to read and comprehend.

Solution: Creating private auxiliary methods to distribute its function.

Review:
Code smell is well identified and the solution is also correct. - Joana Tomás 60152

Joana Tomás 60152

1. Duplicated code

```
public ResourceNewAction(HumanResourceManager hrManager, ProjectDatabase
projectDatabase, RoleManager roleManager, TaskManager taskManager, UIFacade
uiFacade) {
  super("resource.new", hrManager);
  myUIFacade = uiFacade;
  myRoleManager = roleManager;
  myTaskManager = taskManager;
  myProjectDatabase = projectDatabase;
}

private ResourceNewAction(HumanResourceManager hrManager, ProjectDatabase
projectDatabase, RoleManager roleManager, TaskManager taskManager, UIFacade
uiFacade, IconSize size) {
  super("resource.new", hrManager, null, size);
  myUIFacade = uiFacade;
  myRoleManager = roleManager;
  myTaskManager = taskManager;
   myProjectDatabase = projectDatabase;
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/action/resource/ResourceNewAction.java

Both code fragments look identical, making the code more difficult to change and update.

Solution: The constructor that has more arguments calls the other and sends the parameters they have in common, avoiding repeated code.

Review:
Smell solution doesn't make much sense because they're both constructors - Guilherme Poças 60236

2. Large class

```
public class OptionsPageBuilder
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/gui/options/OptionsPageBuilder.java

This class contains many lines of code. This occurs because this class has more responsibilities than it should.

Solution: Some of the responsibilities in this class can be moved to another class, making it simpler.

Review:
This smell isn't suitable because the class isn't that large and mostly made of unimplemented methods - João Agostinho 57538

3. Long Method

```
 public Component buildPageComponent() {
  final GanttLanguage i18n = GanttLanguage.getInstance();
  final Box result = Box.createVerticalBox();

  myWeekendsPanel = new WeekendsSettingsPanel(getProject(), getUiFacade());
  myWeekendsPanel.setAlignmentX(Component.LEFT_ALIGNMENT);
  myWeekendsPanel.initialize();
  result.add(myWeekendsPanel);

  result.add(Box.createVerticalStrut(15));

  myProjectStart = getProject().getTaskManager().getProjectStart();
  myProjectStartOption = new DefaultDateOption("project.startDate",
myProjectStart) {
    private TimeDuration getMoveDuration() {
      return
getProject().getTaskManager().createLength(getProject().getTimeUnitStack().ge
tDefaultTimeUnit(),
          getInitialValue(), getValue());
    }

    @Override
    public void setValue(Date value) {
      super.setValue(value);
      TimeDuration moveDuration = getMoveDuration();
      if (moveDuration.getLength() != 0) {
        updateMoveOptions(moveDuration);
      }
    }

    @Override
    public void commit() {
      super.commit();
      if (!isChanged()) {
        return;
      }
      try {
        moveProject(getMoveDuration());
      } catch (AlgorithmException e) {
        getUiFacade().showErrorDialog(e);
      }
    }
  };

  Box myMoveOptionsPanel = Box.createVerticalBox();
  myMoveOptionsPanel.setAlignmentX(Component.LEFT_ALIGNMENT);

  Box dateComponent = Box.createHorizontalBox();
  OptionsPageBuilder builder = new OptionsPageBuilder();
  dateComponent.add(new
JLabel(i18n.getText(builder.getI18N().getCanonicalOptionLabelKey(myProjectSta
rtOption))));
  dateComponent.add(Box.createHorizontalStrut(3));
  dateComponent.add(builder.createDateComponent(myProjectStartOption));
  dateComponent.setAlignmentX(Component.LEFT_ALIGNMENT);
  myMoveOptionsPanel.add(dateComponent);
  myMoveOptionsPanel.add(Box.createVerticalStrut(5));
```

```
  myMoveStrategyPanelWrapper = new JPanel(new BorderLayout()) {
    @Override
    public void paint(Graphics g) {
      if (isEnabled()) {
        super.paint(g);
        return;
      }
      final BufferedImage buf = new BufferedImage(getWidth(), getHeight(),
BufferedImage.TYPE_INT_RGB);
      super.paint(buf.getGraphics());
      final float[] my_kernel = { 0.0625f, 0.125f, 0.0625f, 0.125f, 0.25f,
0.125f, 0.0625f, 0.125f, 0.0625f };
      final ConvolveOp op = new ConvolveOp(new Kernel(3, 3, my_kernel),
ConvolveOp.EDGE_NO_OP, null);
      Image img = op.filter(buf, null);
      g.drawImage(img, 0, 0, null);
    }
  };
  myMoveStrategyPanelWrapper.setAlignmentX(Component.LEFT_ALIGNMENT);

  myMoveAllTasks = new
JRadioButton(i18n.getText("project.calendar.moveAll.label"));
  myMoveAllTasks.setAlignmentX(Component.LEFT_ALIGNMENT);

  myMoveStartingTasks = new
JRadioButton(MessageFormat.format(i18n.getText("project.calendar.moveSome.lab
el"),
      i18n.formatDate(CalendarFactory.createGanttCalendar(myProjectStart))));
  myMoveStartingTasks.setAlignmentX(Component.LEFT_ALIGNMENT);

  ButtonGroup moveGroup = new ButtonGroup();
  moveGroup.add(myMoveAllTasks);
  moveGroup.add(myMoveStartingTasks);
  moveGroup.setSelected(myMoveAllTasks.getModel(), true);

  Box moveStrategyPanel = Box.createVerticalBox();
  myMoveDurationLabel = new JLabel();
  myMoveDurationLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
  moveStrategyPanel.add(myMoveDurationLabel);
  moveStrategyPanel.add(myMoveAllTasks);
  moveStrategyPanel.add(myMoveStartingTasks);

  myMoveStrategyPanelWrapper.add(moveStrategyPanel, BorderLayout.CENTER);
  myMoveOptionsPanel.add(Box.createVerticalStrut(3));
  myMoveOptionsPanel.add(myMoveStrategyPanelWrapper);

  UIUtil.createTitle(myMoveOptionsPanel,
i18n.getText("project.calendar.move.title"));
  result.add(myMoveOptionsPanel);

  updateMoveOptions(getProject().getTaskManager().createLength(0));
  return OptionPageProviderBase.wrapContentComponent(result,
getCanonicalPageTitle(), null);
  return true;
}
```

Location:
ganttproject/src/main/java/net/sourceforge/ganttproject/gui/options/ProjectCalendarOption
PageProvider.java

This method contains too many lines of code. This probably means that is more complex than it needs to be.

Solution: Make auxiliary methods to simplify the code.

Review:
This smell doesn't seem right as the method is not long enough to be a code smell - João Agostinho 57538

User Stories

1º Functionality

User story:

As a user, I want to be able to divide a given task into smaller objectives, so that I can organize better with other people working on the same task and complete it gradually.

When creating a task, or in a previously created task, it will be possible to create a list of objectives, each objective with its weight in percentage of the original task.

For example, in the task "Make a report", we can have the objectives:

- "Make the cover" – 25%;
- "Make the index" – 25%;
- "Do the development" – 50%;

When each objective is completed, it contributes its weight to the progress of the original task.

2º Functionality

User story:

As a coordinator, I want to be notified when a task reaches a certain point in its duration, so that I can make sure that the task is progressing and will be finished in time.

In the properties of a task, it will be possible to select a email notification option and a corresponding value in percentage, which when selected means that when the total time value of the task passes, an email is sent to the coordinators of that task.

For example, in the task "Make a report" that lasts 10 days, if we select this option with a value of 90%, then at the beginning of the 10th day an email will be sent to the coordinators' email addresses.

Implementation

1st Functionality:

- The user can manually input percentage on a task with objectives, but only if there is percentage not being used by any objective. For example, if a task has one objective worth 60%, then the user can still change the remaining 40%, but if we add an objective worth 40%, then the user can not alter its percentage manually.
- If the task is a supertask, or a mark, then it's not possible to add objectives.
- We can see a task's objectives by moving the cursor over it, in a popup. The checked objectives appear in green, and the unchecked ones in red.
- In the task's properties menu, if we click cancel, all changes are discarded.
- When saving the project, the objectives are saved as well.
- When creating objectives, we can create objectives with 0 percentage value, but when we click ok, all the objectives with 0 percentage are deleted.
- The sum of all objective percentages must be 100, when we exceed that value, the maximum allowed value is set instead. For example, we create an objective worth 90%, if we then create another worth 40%, the percentage value will be 10% instead.
- When the sum of all objective percentages reaches 100, the creation of new objectives is disabled

This functionality's tests are found in ganttproject-tester/test/First_Functionality. There are tests for the 3 main classes of the functionality (there are more classes, but these can be hard to test)

2nd Functionality:

- If the user opens a GanttProject that had a task with email notification activated on a percentage which was supposed to be sent while the app was closed, it sends it immediately.
- If the user changes the task's start to an earlier date while having the email notification activated, it might happen that the selected percentage is no longer available, in which case it sends the email immediately.
- A popup message will appear on screen warning that a late email was sent if one of these two situations happen.
- When the email is sent late, it is sent with the updated percentage. For example had the user set the percentage to 10% but one of the above two situations happened making the task now at 60%, the email is sent saying the task has reached 60% (current) instead of 10% (selected).
- The email is "sent" even if there are no coordinators, meaning that the popup message still appears if the email is "sent" late.
- After an email is sent, the email notification field deactivates itself. It can then be reactivated for a new email notification.
- If a task has already ended, the email notification field stays completely disabled.

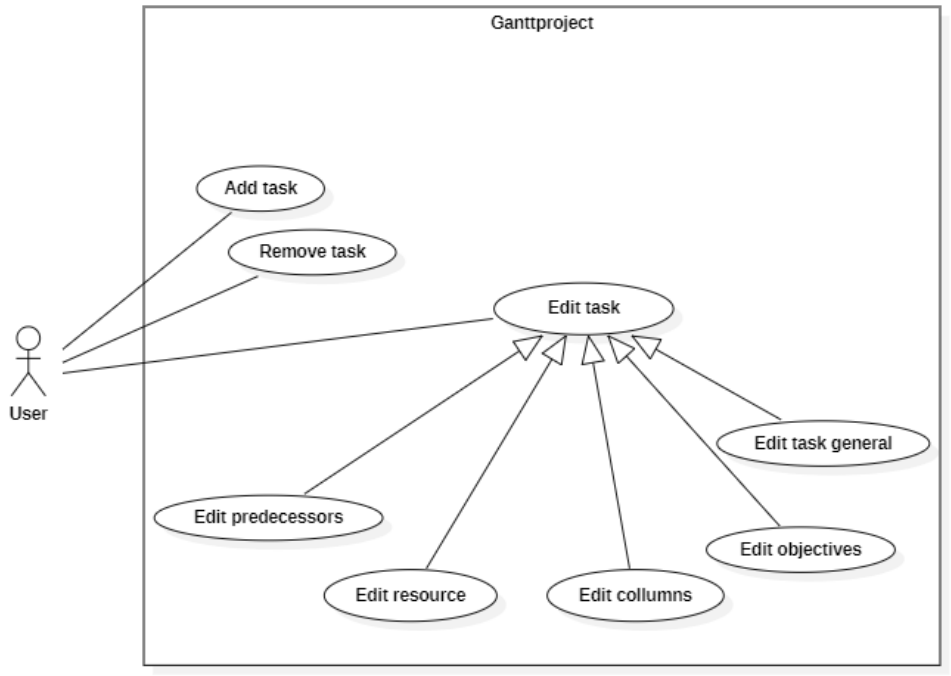
As discussed with both teachers in their pratical classes, it was not possible to create Unit Tests for this functionality's classes (EmailSender and EmailScheduler):

- EmailSender could not have unit tests because the class only does 2 things: establish a SMTP connection and send an email. These functionalities can't really be tested with Unit Tests so it was not possible to create them.

- EmailScheduler had various problems, one of them being that it only works with actual real life time, but the main one was: it receives a Task object as a parameter in its constructor and every single other method is completely reliant on it. Although they mainly work with primitive types, they are all gotten from the Task object and can't be inputted directly into the methods. The Task object (in this case TaskImpl) is so complex that it cannot be realistically created manually in a test class. It takes a TaskManagerImpl object as an argument, which itself takes another two complex objects as arguments (and these two might take even more, it's a never-ending rabbit hole).

Use Cases

Guilherme Poças 60236



Actors: the current user of the program.

Add task
Description: The user adds a task to the program.

Delete task
Description: The user deletes a task(s) of the program.

Edit task (abstract)
Description: The user edits the properties of an existing task.

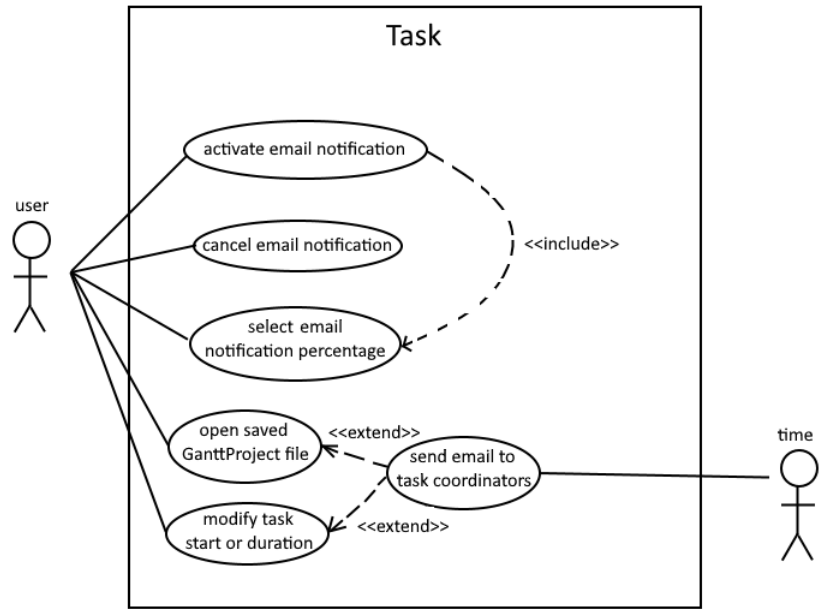
Edit predecessors
Description: The user edits the properties of an existing task, adding, removing, or editing its relationship with other existing tasks.

Edit columns
Description: The user edits the properties of an existing task, adding, removing, or editing its columns, allowing for custom data to be saved.

Edit task objectives
Description: The user edits the properties of an existing task adding, removing, or editing its objectives.

Edit task general
Description: The user edits the general properties of an existing task.

Samuel Pires 60718



Actor(s): The current user of the program and time.

Activate Email Notification
Description: The user activates the Email Notification feature on the task.

Cancel Email Notification
Description: The user deactivates the Email Notification feature on the task.

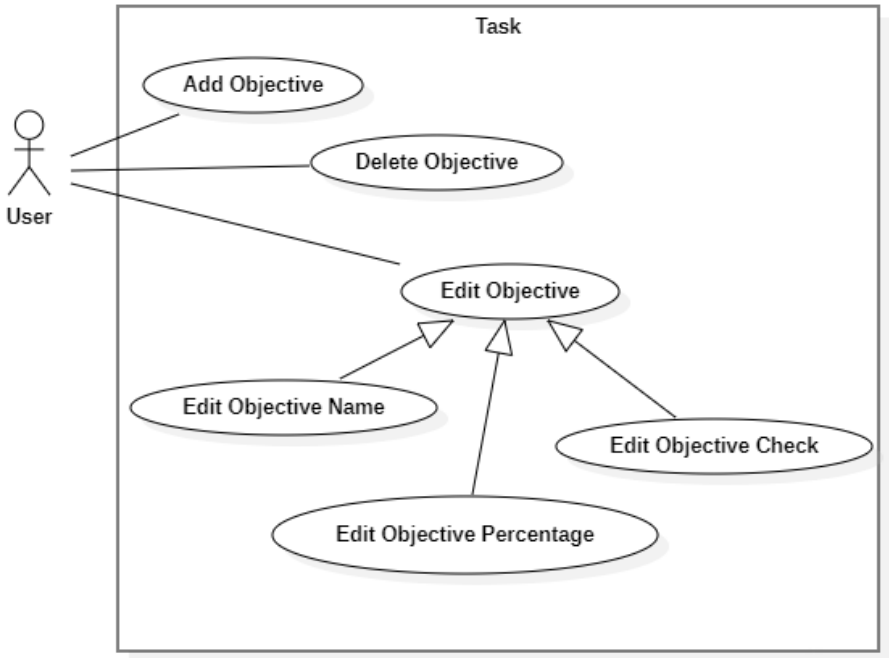
Select Email Notification Percentage
Description: The user selects an available percentage on the task's duration for the Email Notification feature.

Open Saved GanttProject File
Description: The user edits the name of an existing objective of a task in the program.

Modify Task Start or Duration
Description: The user changes the task's start and/or duration values.

Send Email to Task Coordinators
Description: If the Email Notification feature is activated, when enough time has passed to reach the current task's selected percentage, an email is sent to all its coordinators. It can also happen that the user opens a saved GanttProject file which has an email that should've been sent while the app was closed, or changes the task's start and/or duration in a way in which the selected percentage has already passed. In both of these cases, the email is sent immediately.

Leticia Silva 60221



Actor(s): The current user of the program.

Add Objective
Description: The user adds an objective to a task in the program.

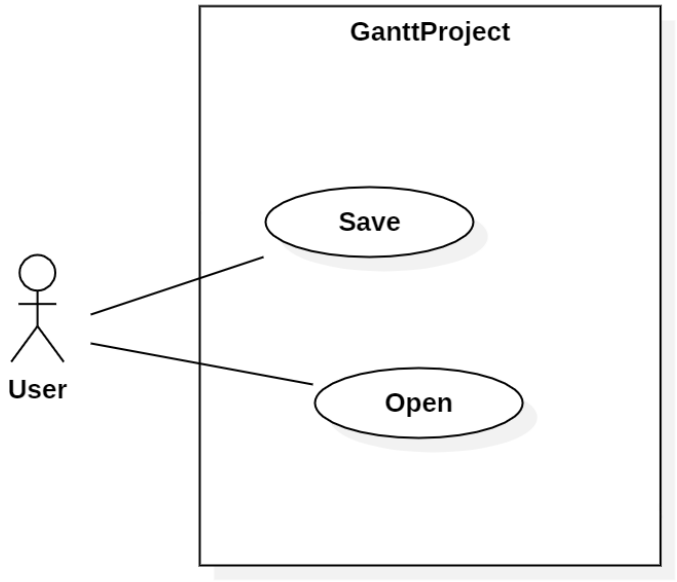
Delete Objective
Description: The user deletes an objective to a task in the program.

Edit Objective (Abstract)
Description: The user edits an existing objective of a task in the program.

Edit Objective Name
Description: The user edits the name of an existing objective of a task in the program.

Edit Objective Percentage
Description: The user edits the percentage of an existing objective of a task in the program.

Edit Objective Check
Description: The user checks or unchecks an existing objective of a task in the program.

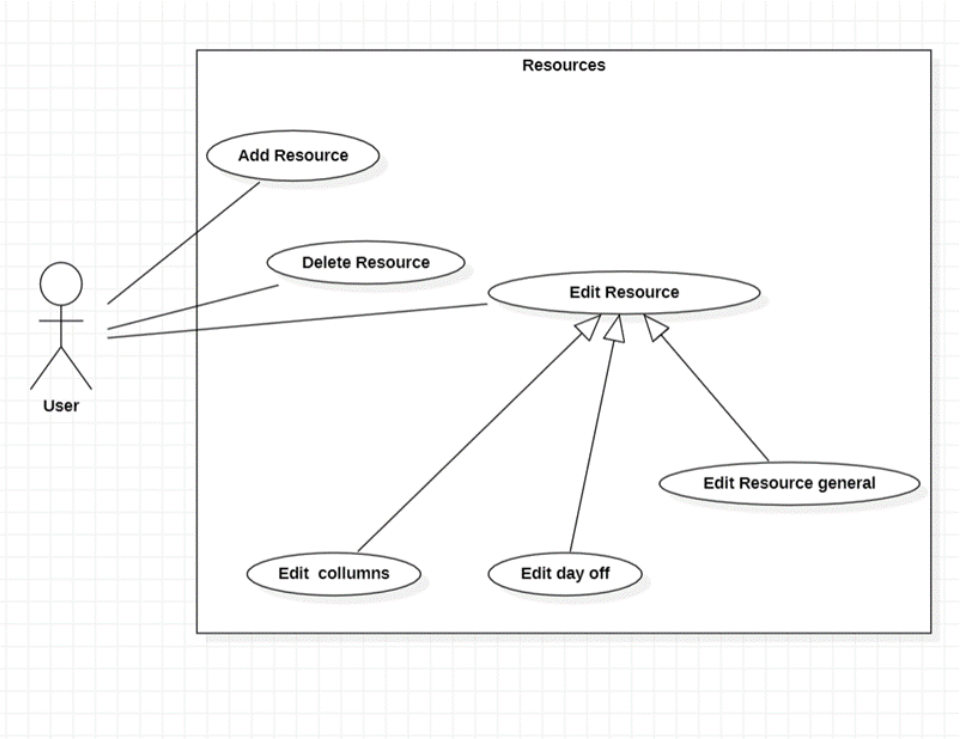


Actor(s): The current user of the program.

Save Project
Description: The user saves the project, and its content is saved in a file.

Open Project
Description: The user opens an existing project with its content previously saved.

Joana Tomás 60152



Actor

The current user of the program.

Add Resource

Description: The user adds a resource.

Delete Resource

Description: The user deletes a resource.

Edit Resource (Abstract)

Description: The user edits an existing resource, or something related to the resource.

Edit columns

Description: The user edits the properties of an existing resource, adding, removing, or editing its columns, allowing for custom data to be saved.

Edit days off

Description: The user can add, delete, or edit days off, by changing the beginning and ending date.

Edit Resource general

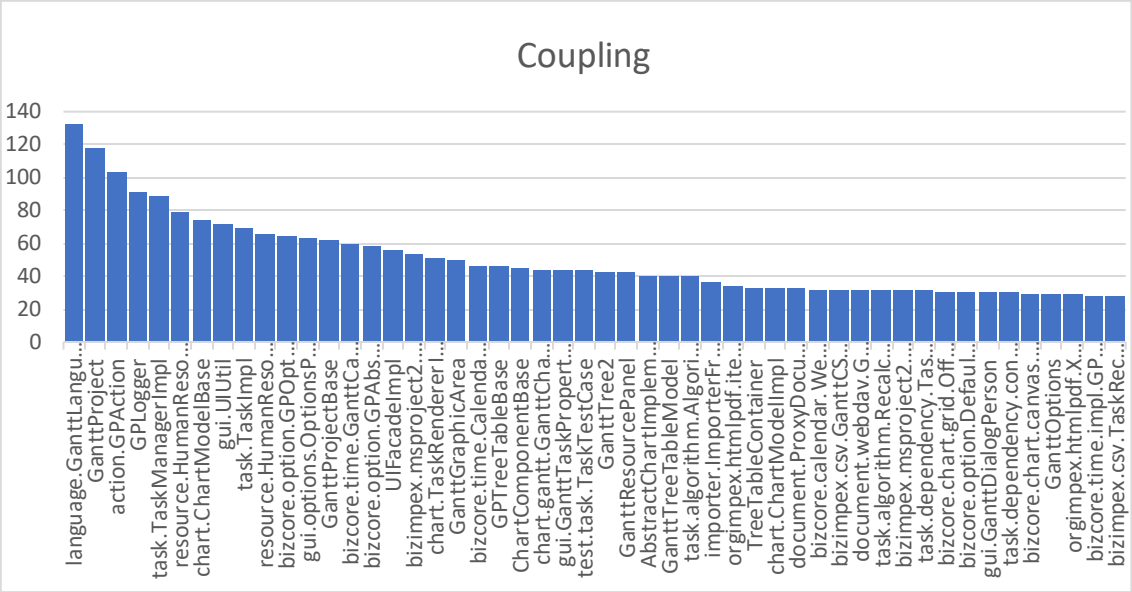
Description: The user edits the general properties of an existing resource.

Codebase Metrics

Guilherme Poças 60236

Chidamber-Kemerer metrics

Coupling between objects



A classes coupling is defined by the number of classes that depend on it, or that it depends on. On the collected metrics we can see that, as expected, central classes to the program have a high coupling, while outer classes that are used less have a lower coupling. For example, the GanttLanguage class has the highest coupling, since it's used by a lot of other classes.

Generally high coupling should be avoided, since it complicates the code, like in the GanttProject class, that has the second highest coupling since it uses a lot of other classes. We can begin to see the code smell "god class" here, and this class could have some responsibilities moved to other parts of the code, reducing the number of classes it uses, and it's coupling.

Inherence depth

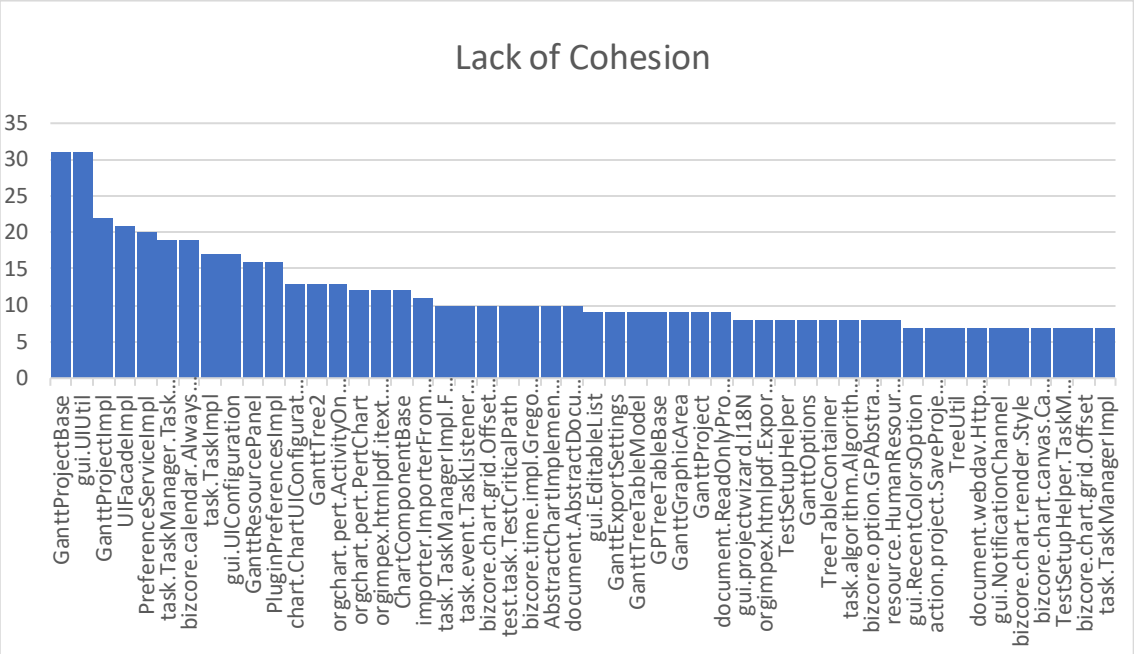
This metric measures the number of inheritance steps between a given class and the java Object class. Highly specialized classes have a high inherence depth, such as the GanttTreeTable, while more standalone classes that don't extend anything don't.

Number of children

This metric is closely related to the one before, measuring the number of classes that extend a given class. In the code the most extended class is the GPAction class, that is extended by 86 separate classes.

Lack of cohesion of methods

Cohesion is the “strength” of a given class so that it’s responsibilities can’t be easily split into new different classes. We measure this in the relation of methods the methods in the class, that is, if they share a variable, or call each other. If they all the methods in a class are highly related, then the class Is strongly cohesive, and the value is close to 1.



In the program we can see some classes that have a high lack of cohesion, and once again, the ganttProject class comes up, which is more proof that it might be a "god class” and needs to be separated into smaller classes.

Class response

The class response is the number of methods it can possibly call, be it their methods, or other class methods that it can access. Generally, we want a small class response, since it increases the complexity of the class, and decreases cohesion and stability. Once again, the highest class response in the project belongs to the ganttProject class since it handles so many different classes, and has so many responsibilities, which might be a problem.

Method complexity

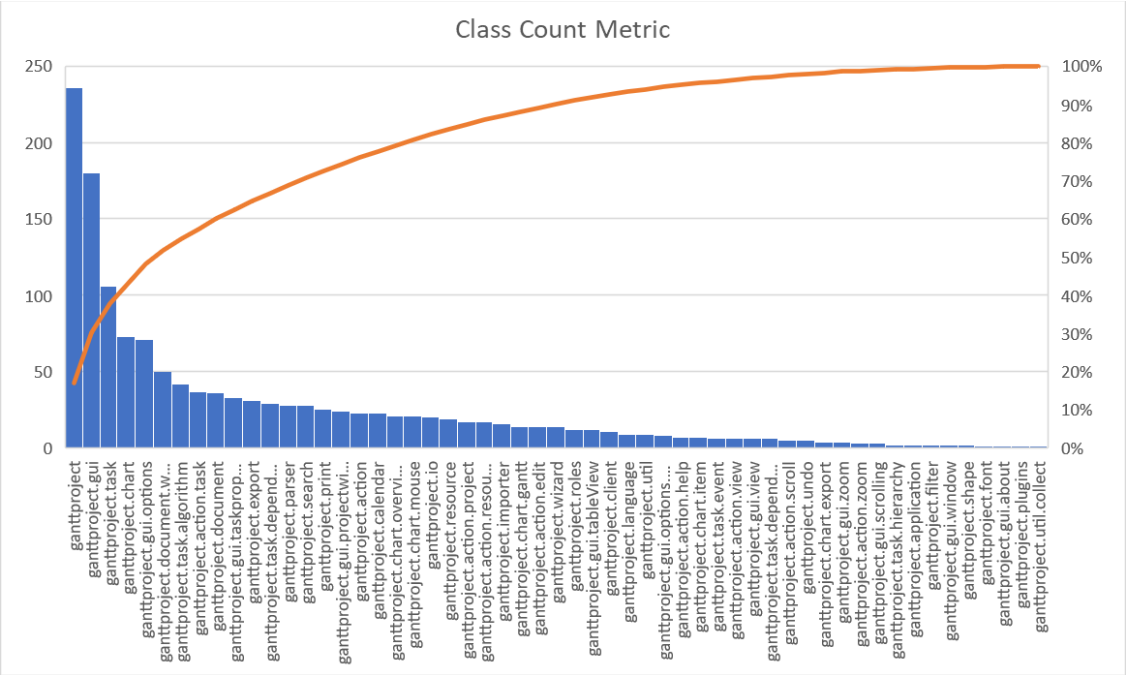
This metric measures the cyclomatic complexity of the methods in each class, which is the number of independent paths the program can take, for example an if statement would have a complexity of 2. Lower complexity methods easier to understand and to test. In the program, the higher complexity classes are the ones responsible for tasks, and once again, the ganttProject class.

Samuel Pires 60718

Class Count Metrics

Number of Classes

This metric represents how many classes each package contains (non recursive):



As expected, this metric follows a Pareto Distribution, meaning that a minority of the packages contain the majority of the classes while the majority of the packages contain only a small portion of the classes.

As one can observe, “ganttproject” (the root package in this metric) contains the majority of the classes (236), while two of its direct subpackages: “gui” and “task” contain another big portion of the classes (180 and 106, respectively). Together, these three packages contain about 40% of the classes (522 out of 1385).

This becomes a problem when talking about the project’s complexity. While it is understandable to have this many classes in singular packages in a project as big as this, one should avoid storing an excessive number of classes in a single package, as can be seen here in the above-mentioned folders, which all have more than 100 classes inside.

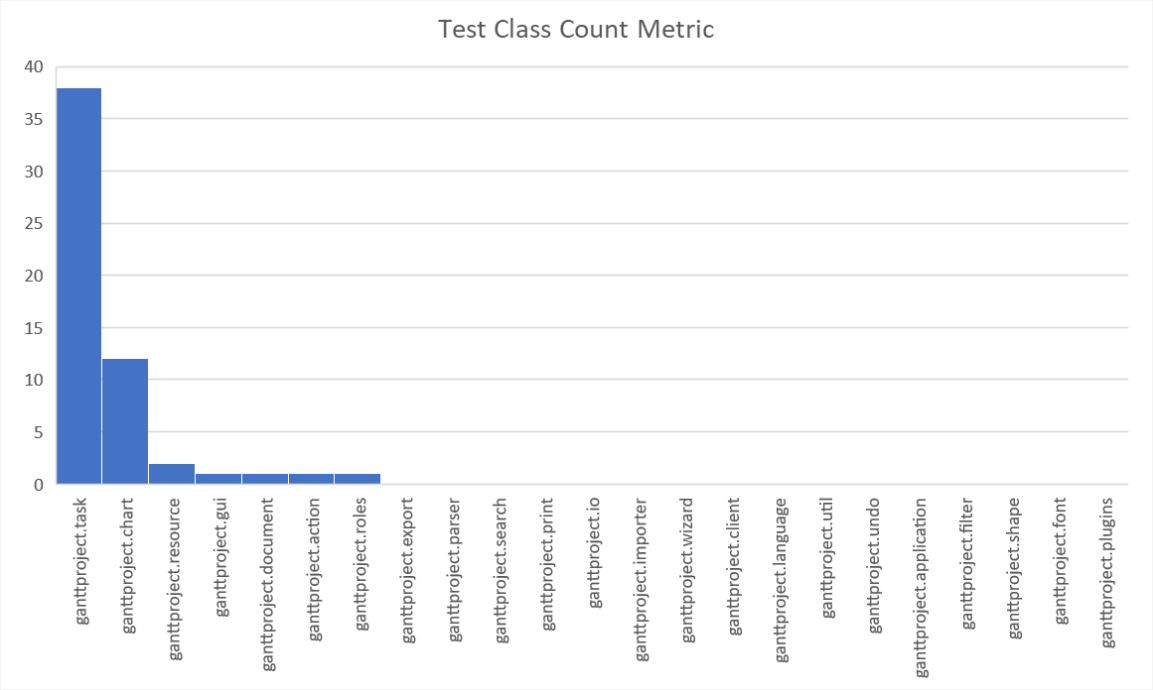
I can see this metric being related to one of the most commonly observed code smells in the first phase of the project: Dead Code. By opening these classes, a few of them will be either dead code or completely unused in the rest of the project.

By analysing this metric, I see two things that could be done to better this project in its complexity related to folders and classes:

- Partition the classes inside the packages which have an excessive amount of them, as to simplify their organization in this project and make it more understandable and manageable.
- Remove any classes that are purely dead code or are unused.

Number of Test Classes

This metric represents how many test classes each package, or, as I look at it, each part of the project contains (using only direct subpackages to root one in this metric and recursive in order to simplify):



Just by taking a quick glance at this metric, we can see that a big majority of the parts of this project do not have tests, and most of those that do have very few. Other than that, the "task" package instantly strikes the eye has having by far the most tests (38 out of 53), well above half.

This project deals with tasks more than anything else, they are the core of this project and everything else is constructed and used around them so it's understandable to have this many test classes dedicated to them. However, a great amount of the parts of this project do not have any test classes, and that is a problem.

Not having enough test classes does not contribute directly to code smells, since they only test the code's functionality, not its structure. However, it poses a problem to the future of the project: refactoring or extending a project with not enough tests may introduce new bugs and/or code smells.

Note: the metrics files do not exactly match since the examined files are in different parts of the project and MetricsExtended did not detect TestClasses, among other problems, so I had to sort them out.

Leticia Silva 60221

Complexity metrics

Cyclomatic Complexity

Cyclomatic complexity is a metric used to indicate the complexity of a program. It calculates several linearly independent paths that can generate all possible paths through a method.

Average operation method

This metric calculates the average Cyclomatic Complexity of all non-abstract methods in each class. Inherited methods are not counted.

For example, the class GanttXMLOptionsParser has "long method" code smells. Specifically the method startElement() has more than 200 lines with a lot of if statements which indicates that the method is very complex(has higher cyclomatic complexity values) and should be divided into different methods.

Maximum operation method

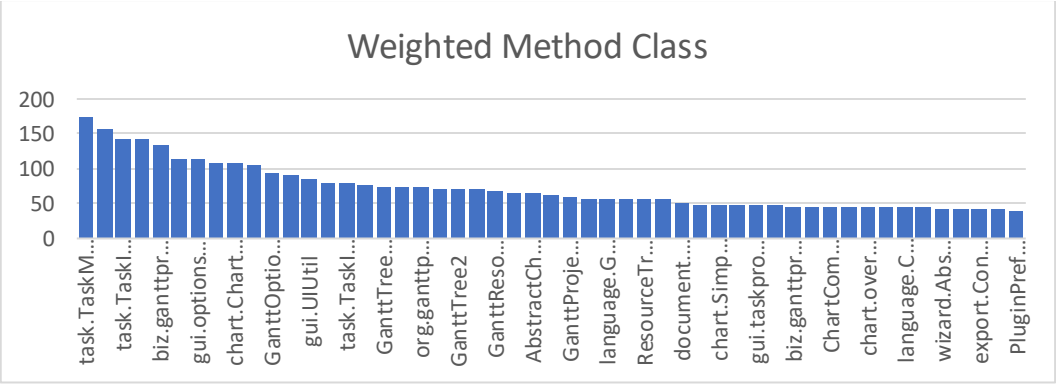
This metric calculates the maximum Cyclomatic Complexity of the non-abstract methods in each class. Inherited methods are not counted.

The class mentioned above has a high cyclomatic complexity value (it is the maximum complexity of a method calculated in the class), which means the code is more complex, increase the risk to modify and is harder to comprehend.

Weighted method

This metric calculates the total cyclomatic complexity of the methods in each class.

A class with a low cyclomatic complexity is better, ensures good readability and maintainability of the code. Reducing the complexity of the class makes writing test cases easier. If the class has high cyclomatic complexity probably is a long class and is doing too many things therefore should be split.



Joana Tomás 60152

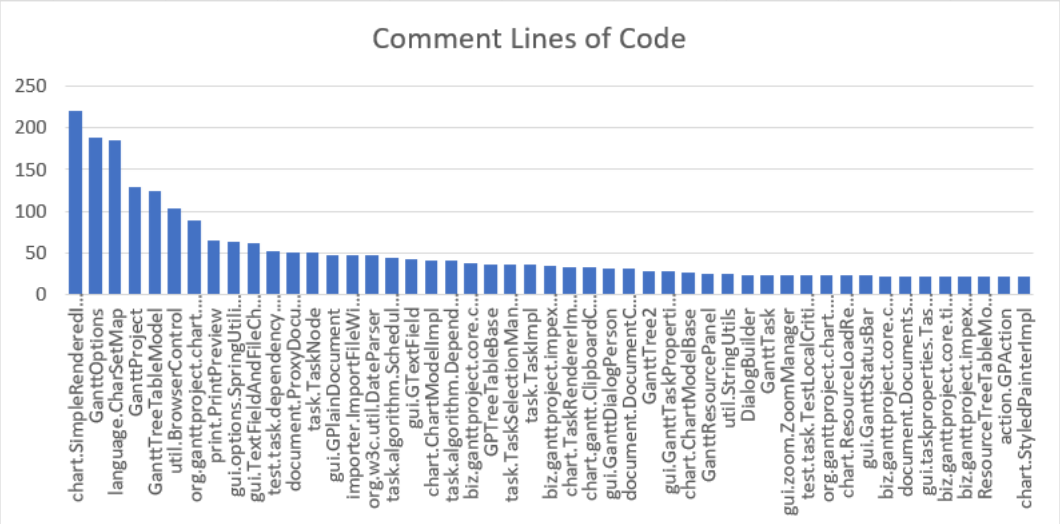
Line of Code Metrics

Comment Lines of Code

This metrics calculates the number of lines that contains comments in each class. Empty lines are not counted.

SimpleRenderImage is a class that has excessive lines of commented code, making the class very extensive and hard to read.

On the other hand, the Borders class, for example, doesn't have any lines of commented code, so it's more difficult to understand some parts of the code.



Javadoc Lines of Code

This metrics is very similar to the previous. It has the same purpose but instead of using regular comments, now we are calculating the number of lines that contains Javadoc comments. Empty lines are not counted.

Lines of Code

This metrics calculates the number of lines of code in each class. Empty lines don't count, but commented ones do.

The GanttProject class has many lines of code, which can be evidence of a code smell called God class. The solution for this problem is to split this class into smaller ones.

The WebStartIDClass is empty, so it might be a data class, and as such, this class is not necessary.

Demo Video

Demo video link: https://youtu.be/ZODpFQYfm1w