### **School of Electronics and Computer Science**

### **University of Southampton**

### 

### **Runway Redeclaration Tool - Increment 2**

### **COMP2211 Software Engineering Group Project**

Team 10: Andrew John Evans (aje3g16)

Lyubomir Tsvetanov Hristov (lth1u17)

Suyash Datt Dubey (sdd1n17)

Kemal Yesildagli (ky1g17)

Denis Kovachev (dk1g17)

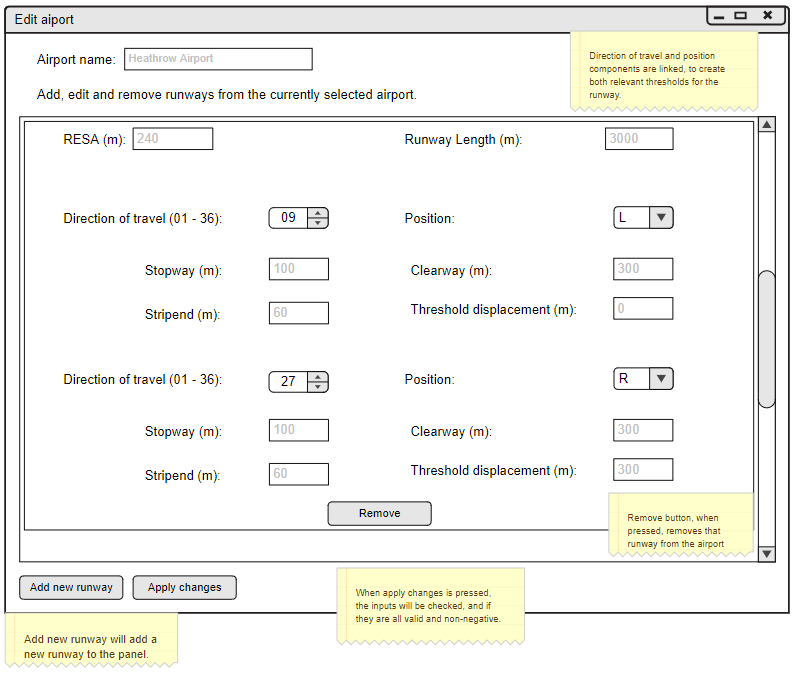
**Responses to feedback**

* Sanitisation of user inputs
  + One piece of feedback from the last sprint review is that some inputs for the obstacle specification were not checked for negative numbers - which in some cases were invalid (note, we already had checks for non-number inputs). In response, we added this to the obstacle user input and made sure to do these checks for all further user input.
* Formatting of UML diagrams
  + Took care with the formatting of our report, to make it more easily visible when presenting in sprint review meeting - no dark backgrounds as before.

**Discussion of key design choices / decisions**

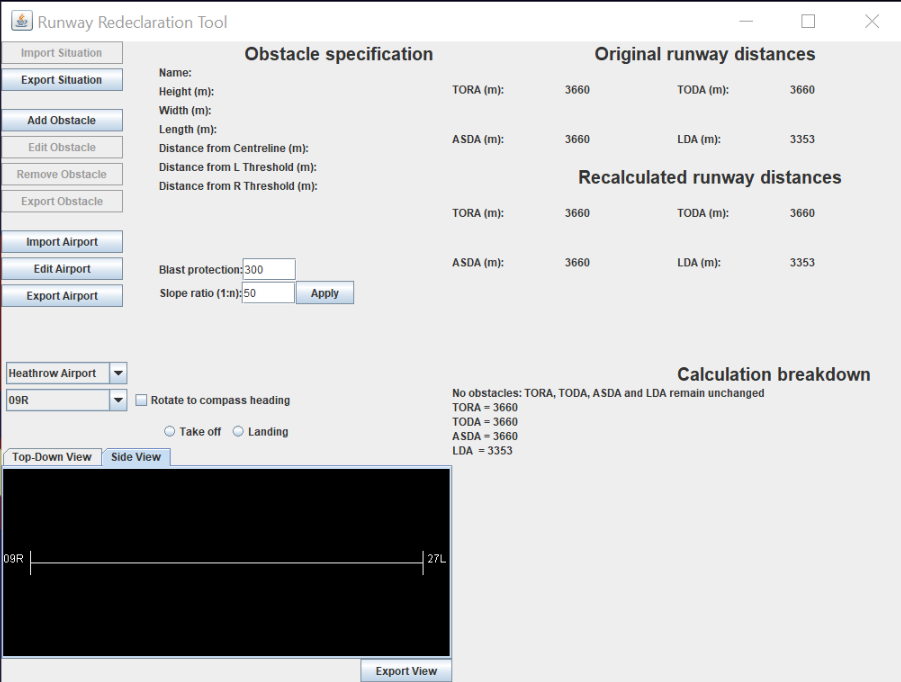
* Recalculation breakdown
  + The recalculation breakdown is added to the Runway class and to the GUI. It describes the various formulas and values used to recalculate the TORA, TODA, ASDA and LDA for a given obstacle depending on the take off/landing scenario.
* Runway and Threshold Selection
  + We have implemented a drop down menu that is used to select different thresholds. When a user selects a threshold then a visualisation of the runway that threshold belongs to is going to appear.
* Side-on visualisation
  + We have implemented a side view panel, which displays the current situation on the selected runway and if there is an obstacle it visualises the runway parameters depending on where the obstacle is located and whether a plane is taking off or landing. We have 2 radio buttons that can be used to specify the latter. These decisions were made after careful considerations and analysis of the different scenarios that can appear on a runway.
* UK Compatibility Integration
  + We started by creating a storyboard for the GUI that displays the various parameters for the runways of the currently selected airport. The decision was made to make this a dialog, to ensure that the main menu is not overcrowded.
  + We aimed for speed of input as a priority, hence why there is a lot of information of the main frame, but similarly did not want to overcrowd.
  + By allowing one to enter and see the output quickly, it should help the various stakeholders make decisions about the continued operation of the runway quicker This potentially allows safer operation, more profit and fewer disruptions to passengers travel schedules.
* XML Input/Output
  + We have added XML import and export functionality with the XMLImporter and the XMLExporter classes.
  + Made use of existing libraries, to allow for the XML structure to be built and modified quickly, and reducing the chances of bugs.
  + The implemented features consist of importing and exporting the airport (with its runway data) and importing and exporting the current obstacle, with checks of invalid parameters.
* Export displays in different formats
  + To allow for the easiest export of the view, we made the decision for the runway views to be drawn to buffered images, allowing easy drawing to the GUI and to files.
  + To start with, we chose the formats to output to. We began with the ones in the specification (JPEG, PNG, GIF) and through research and prior knowledge, we identified that BMP would also be a useful output format.
  + Added a JFileChooser to allow the user to specify the name and location of the the exported view.
  + We have implemented the functionality, which enables the user to export the current side view. This is very useful when you want to compare or store a specific situation.

**Key storyboards and screenshots**

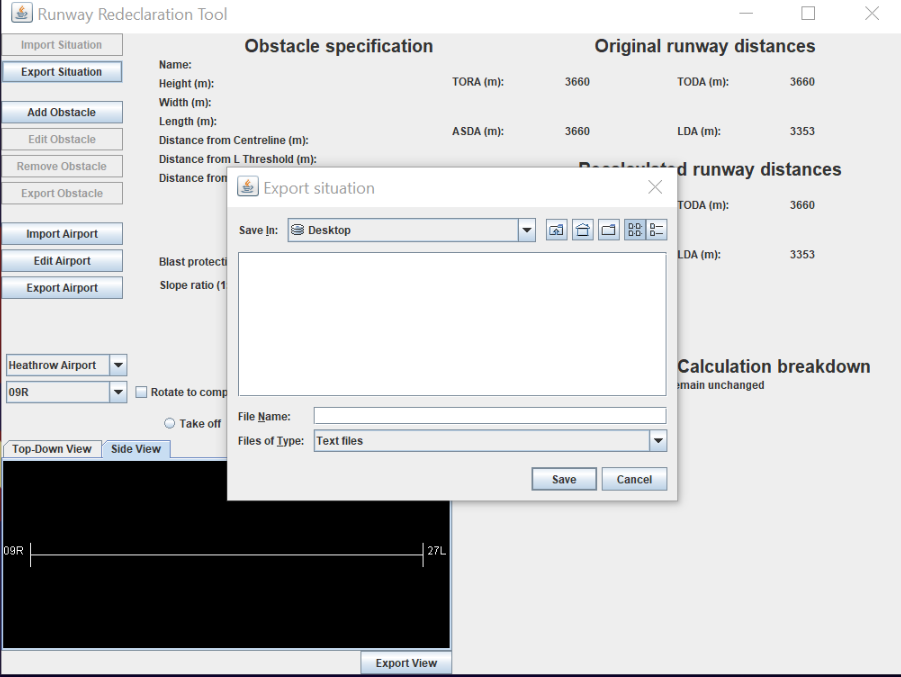
****

Input dialog for the currently selected airport, allowing you to add, edit and remove runways. Each runway has two thresholds. Some data is shared among thresholds. The information provided is used to calculate the relevant distances, such as TODA, TORA, ASDA and LDA. These panels are replicated for multiple runways in the currently selected airport.

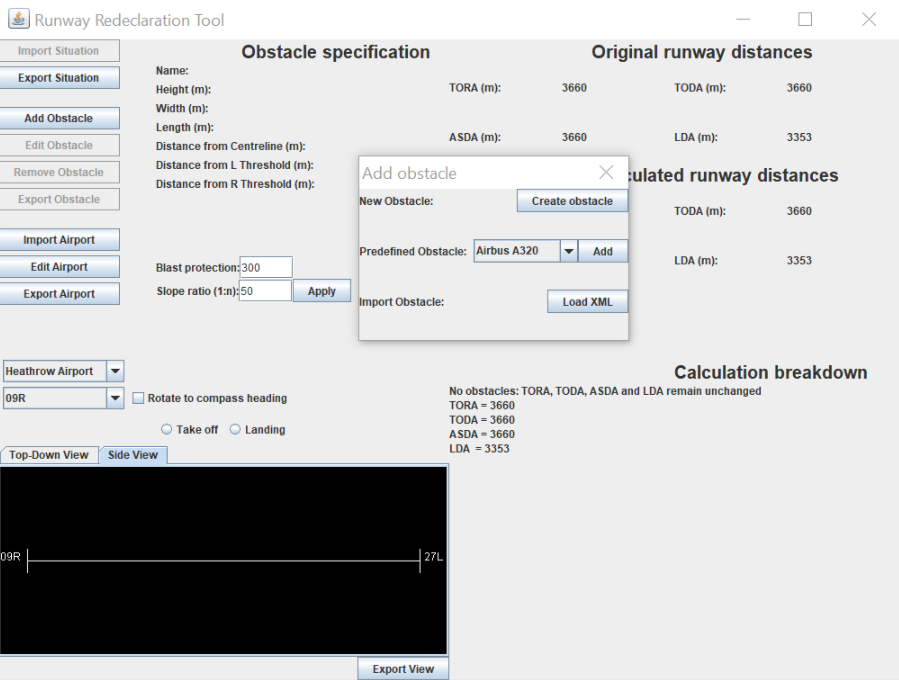
**1. GUI on Start Up**



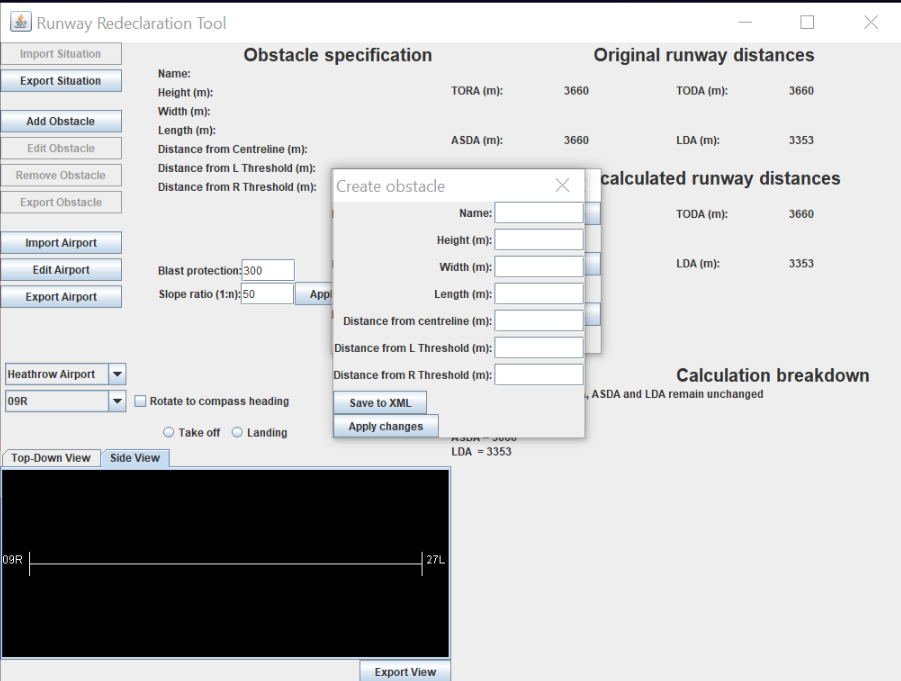
**2. Export Situation Dialog (Other export and import file selector dialogs are similar)**



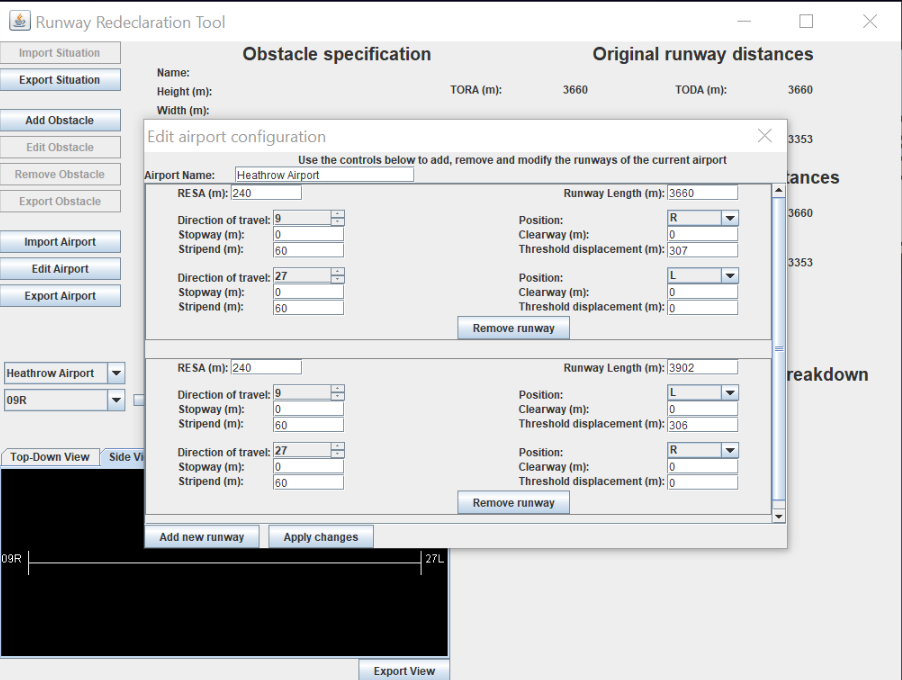
**3. Add Obstacle Dialog**



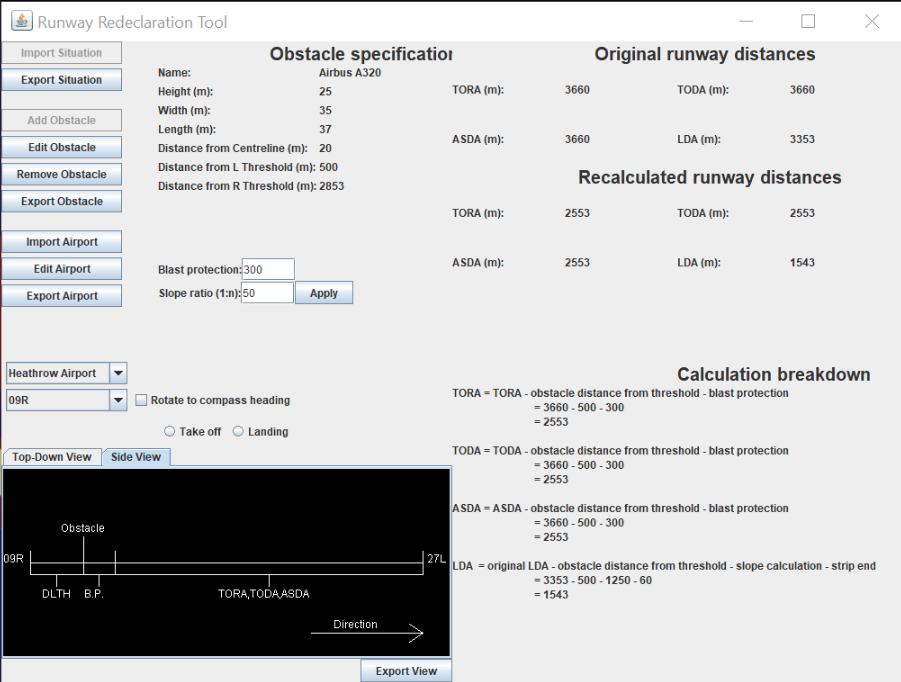
**4. Create Obstacle Dialog**



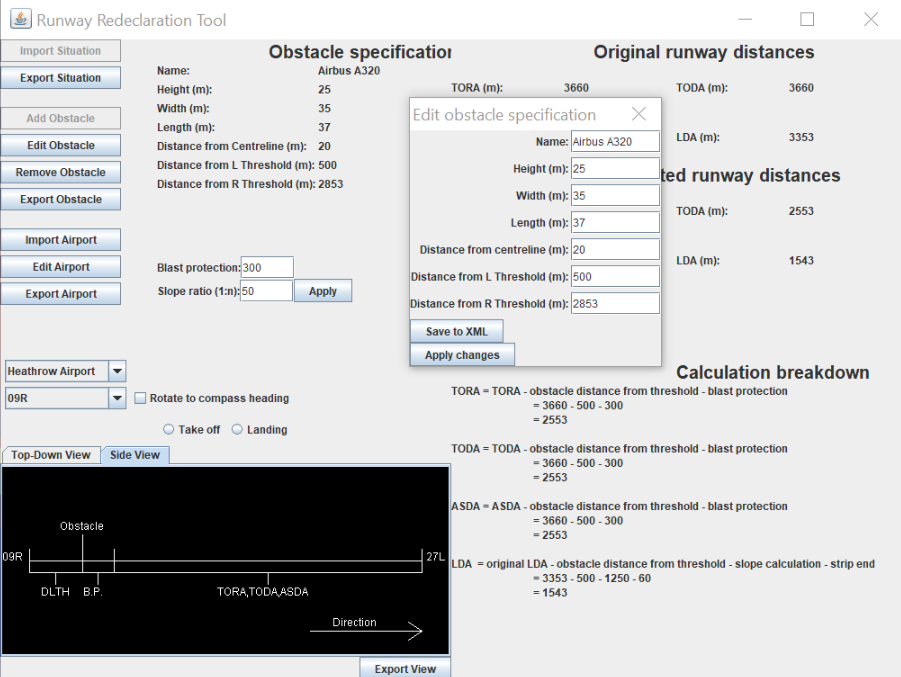
**5. Edit Airport**

****

**6. GUI with Obstacle Loaded**

****

**7. Edit Obstacle Dialog**

****

**Testing during this increment**

For testing of the UK compatibility/edit airport GUI, there was no easy way to automate these tests. We performed a variety of tests on different inputs for each field, testing the boundaries. For each field, we checked either side of the boundary - for all fields it we tested at below zero, zero and above zero (the three relevant classes for each field). Also, using debugging tools in IntelliJ to check the state of the airport after applying changes and using the XML export functionality, we checked that any edits to the airport configuration were reflected correctly in the state of the program. We then performed a similar set of tests for the ‘edit obstacle’ user input, making sure that the inputs are properly sanitised and changes are properly reflected in the state of the program.

Testing for exporting displays was relatively simple. With the view written to the buffered image already, we just had to check error cases for the file selection and that the outputted image matched the one on the view. We tested input file names with and without extensions for our error cases, and in all cases, the correct extension was appended. We didn’t test this for compatibility on other platforms, an improvement for next time. The Java implementation details suggests that output of these file types should be supported on all JVM implementations.

Testing for XML Input/Output: We checked if the exported XML file contained the correct data. For input, we made sure that invalid parameters were not allowed such as taking a String when it requires an Integer, and negative parameters were also not allowed.

Testing if the side view is displayed correctly is done by putting some predefined obstacles on the runway, for which we already know how the runway parameters should change depending on the scenario, and then comparing the visualisation with the expected one. We have covered all different cases and experimented with some randomly chosen values.

Testing for threshold and runway selection are easily testable with JUnit since we can just use an actionListener to check whether when a threshold is selected, the current threshold is updated. Moreover, the runway selected can also be observed in the GUI since when a user selects a threshold then the whole runway with its 2 thresholds is being displayed.

Testing for calculation breakdown: We checked if the correct formulas, values and calculations was being displayed for various scenarios of taking off and landing on different runways.

**Sprint 2 burndown chart (Key)**

**1:** UK Compatibility Integration

**6:** Recalculation breakdown

**7:** XML Input/Output

**11:** Runway and Threshold Selection

**14:** Side-on visualisation

**17:** Export displays in different formats

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tasks | 17 | 17 | 17 |  | | | | | | | | | | |  |
|
| 7 | 7 | 7 |
|
| 17 | 17 | 17 |  | | | | | | | |
| 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |  | | | |
| 6 | 6 | 6 |
| 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 1 | 1 | 1 | 1 |
|
| 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |
|
|
|
| Day | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**Increment 3 Plan**

|  |  |  |
| --- | --- | --- |
| XL | 13 | Top-down visualisation |
| L | 8 | Graphic Threshold orientation |
| L | 9 | Graphic Runway Strip Rotation |
| S | 12 | GUI Notifications |
| M | 15 | Ability to zoom and pan the views |
| S | 18 | Accessibility for color blind people |

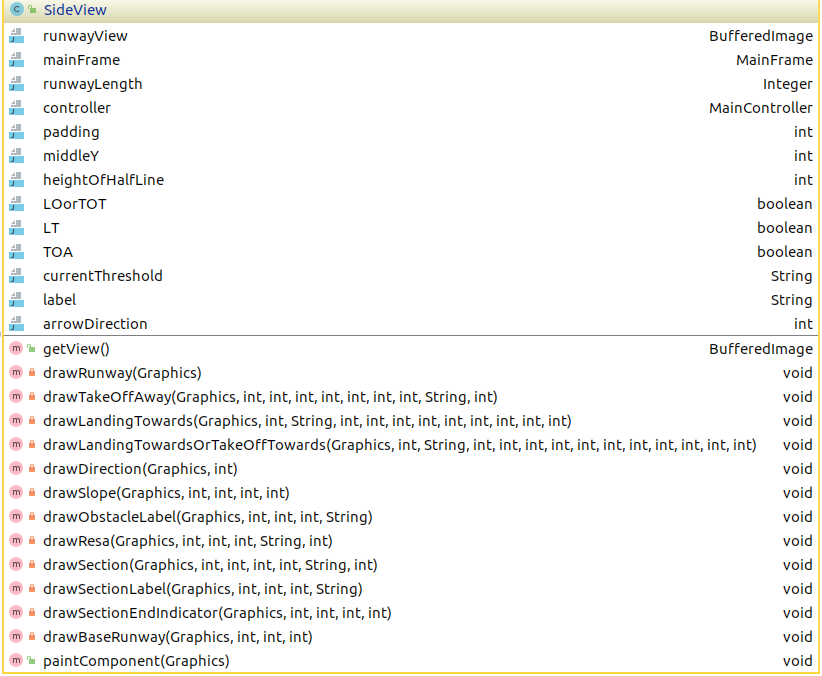
* Top-down visualisation and further testing and refinement of Side-on visualisation
  + Further testing and integration will be needed for the side-on visualisation
  + Create a panel that provides a top view of the runway and the relative values of the runway parameters
  + Show recalculated TORA, TODA, ASDA, LDA etc on side view when there is an obstacle closer to the right threshold
  + Make side-on and top-down views more appealing to the user.
  + Testing of correctness of output (acceptance testing).
* Graphic Runway Strip Rotation
  + Create a button which when clicked rotates the orientation of the runway to its compass heading, when looking at the top view.
  + Acceptance testing to check that the stip is rotating to the correct position and updates according to changes in the model.
* Graphic Threshold orientation
  + Design the top-down and side-on views such that the lower threshold is always located on the left.
  + Testing to make sure that all visualisations are correct with this.
* GUI Notifications
  + Provide notifications to the user when changes occur in the system - making them meaningful.
  + Provide meaningful error messages where necessary and make the errors uniform in style.
  + Test that notifications and error/warning messages are presented in the correct places under the correct conditions.
* Ability to zoom and pan the views
  + Create buttons for zooming and panning the views
  + Implement the panning and zooming features.
  + Acceptance testing of this GUI element.
* Accessibility for people with colorblindness
  + Research the different types of colour blindness and the colours that affect these types. Also, consider the colours which can be easily distinguished by people with the various types of colour-blindness.
  + Create colour schemes for the various components to take into considerations these issues.
  + Create a menu which allows users to choose different colour schemes for the two views in the program.

**Increment 3 - Day 0 burndown chart**

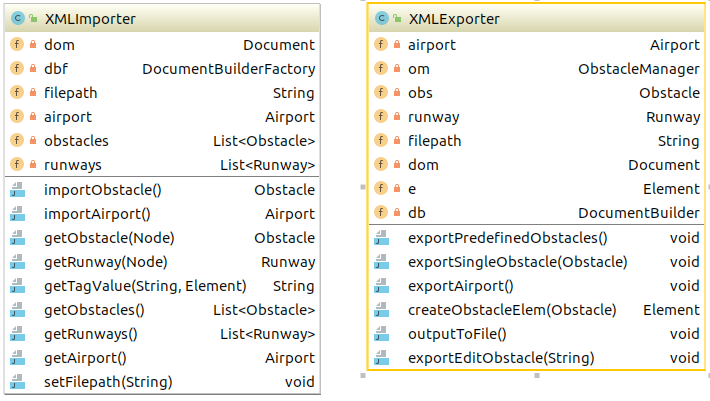
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tasks | 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 18 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Day | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

**UML Diagrams**

**1. Side View**

****

**2. XML import/export**

****