**Project Specification Document**

Adventium

Ansley Schug

Charles Kanoy

Jeremy Rylicki

Riley Abrahamson

**Table of Contents**

Section Page

[1.](#_heading=h.30j0zll) Project Vision and Objectives 1

[1.1](#_heading=h.1fob9te) Project Scope and Vision 1

[1.2](#_heading=h.3znysh7) Project Goals and Objectives 1

[2.](#_heading=h.2et92p0) Project Planning 2

[2.1](#_heading=h.tyjcwt) Project Lifecycle 2

[2.2](#_heading=h.3dy6vkm) Project Setup 2

[2.3](#_heading=h.4d34og8) Stakeholders 2

[2.4](#_heading=h.2s8eyo1) Project Resources 2

[2.5](#_heading=h.17dp8vu) Assumptions 2

[3.](#_heading=h.26in1rg) Project Tracking 3

[3.1](#_heading=h.lnxbz9) Tracking 3

[3.2](#_heading=h.35nkun2) Communication Plan 3

[3.3](#_heading=h.3j2qqm3) Deliverables 4

[3.4](#_heading=h.1y810tw) Project Metrics 4

[4.](#_heading=h.2xcytpi) Requirements (User Stories) 5

[4.1](#_heading=h.1ci93xb) Overall Description 5

[4.2](#_heading=h.3whwml4) Users and Roles 5

[4.3](#_heading=h.2bn6wsx) Use Case Diagrams 5

[4.4](#_heading=h.qsh70q) User Stories (Requirements) 5

[4.5](#_heading=h.3as4poj) Constraints and Limitations 10

[5.](#_heading=h.1pxezwc) Design 11

[5.1](#_heading=h.49x2ik5) Introduction 11

[5.2](#_heading=h.2p2csry) Scope 11

[5.3](#_heading=h.147n2zr) High-Level Component Design 11

[5.4](#_heading=h.3o7alnk) Class Diagram 11

[5.5](#_heading=h.23ckvvd) Activity Diagrams 11

[5.6](#_heading=h.ihv636) Sequence Diagram 11

[5.7](#_heading=h.32hioqz) Alternative Designs and Design Rationale 11

[5.8](#_heading=h.41mghml) Data Architecture 11

[6.](#_heading=h.vx1227) User Interface 12

[6.1](#_heading=h.3fwokq0) UI Description 12

[6.2](#_heading=h.1v1yuxt) UI Mockup 12

[7.](#_heading=h.4f1mdlm) Project Closure 13

[7.1](#_heading=h.2u6wntf) Goals / Vision 13

[7.2](#_heading=h.19c6y18) Delivered Solution 13

[7.3](#_heading=h.3tbugp1) Remaining Work 13

[8.](#_heading=h.28h4qwu) Definitions and Acronyms 14

# 1. Project Vision and Objectives

## 1.1 Project Scope and Vision

As a small portion of the GUMBO project, our group is responsible for creating an OSATE plug-in that translates AADL models from FACE to AGREE models. The vision for our project is to make a simple user interface plugin for the Eclipse IDE. This interface will have a selection experience to reduce the likelihood of error in the generation of AGREE constraints. The end user, members of the US Army, will not have a lot of experience with actually programming in AADL, so this project is critical to creating an error-free final AGREE model with proper constraints.

## 1.2 Project Goals and Objectives

|  |  |
| --- | --- |
| **#** | **Goal or Objective** |
| 1 | Create an OSATE plugin that: |
| 2 | * Reads an AADL instance models through the OSATE API |
| 3 | * Identifies components translated from FACE |
| 4 | * Create GUI that opens a dialog box that collects behavioral details on these components |
| 5 | * Output results as AGREE annex that defines interface behaviors |

# 2. Project Planning

## 2.1 Project Lifecycle

Our project team plans to use an Agile approach with four sprints. The first sprint will focus on project startup, the second will focus on the first delivery, the third will focus on the second delivery, and the last sprint will be focused on finalizing the project for the final delivery. Project startup includes setting up the development environment and establishing what team member will be working on what project area. The first delivery will focus on establishing a UI prototype and getting a better understanding of project details. The second delivery will focus on getting a working first version of the plug-in. The final delivery will focus on finalizing project deliverables and polishing the project itself.

|  |
| --- |
| Project Schedule   * Sprint 1 1/21/2021 - 2/6/2021 Project startup * Sprint 2 2/7/2021 - 3/10/2021 First delivery * Sprint 3 3/11/2021 - 4/8/2021 Second delivery * Sprint 4 4/9/2021 - 5/8/2021 Final delivery |

## 2.2 Project Setup

|  |  |
| --- | --- |
| **#** | **Decision Description** |
| 1 | Setup tool environments - Eclipse and OSATE |
| 2 | Become familiar with AADL models in Java |
| 3 | Become familiar with AGREE syntax |
| 4 | Become familiar with OSATE environment |
| 5 | Draw up a prototype of the UI |

## 2.3 Stakeholders

|  |  |
| --- | --- |
| **Stakeholder** | **Role** |
| Danielle Stuart | Sponsor |
| U.S. Army | This project is funded by the U.S. Army who will be the end users. |

## 2.4 Project Resources

|  |  |  |
| --- | --- | --- |
| **Resource** | **Resource Description** | **Quantity** |
| Capstone Team | The team primarily responsible for the development of the OSATE plugin | 4 |
| OSATE Eclipse environment | The development environment the capstone team will be working on throughout the course of the project. | 4 |
| OSATE documentation | The primary resource to be referenced throughout development for OSATE specific issues. | 1 |

## 2.5 Assumptions

|  |  |
| --- | --- |
| **#** | **Assumption** |
| 1 | Team members will be able to familiarize themselves with OSATE plug-in development |
| 2 | Project examples will be provided as they come available from meetings |
|  |  |

# 3. Project Tracking

## 3.1 Tracking

|  |  |  |
| --- | --- | --- |
| **Information** | **Description** | **Link** |
| Short meeting | Tuesdays after class | Project team |
| Sponsor meeting check-in | Thursdays after class | Project team, mentor, and sponsor |

## 3.2 Communication Plan

**Regularly Scheduled Meetings**

|  |  |  |
| --- | --- | --- |
| **Meeting Type** | **Frequency/Schedule** | **Who Attends** |
| Zoom Meeting | Every Tuesday at 12:45pm | Team (Ansley, Charles, Riley, and Jeremy) |
| Google Teams Meeting | Every Thursday at 12:45pm | Team (Ansley, Charles, Riley, and Jeremy), and Adventium Team (Danielle, Ryan - tbd, and August - tbd) |

**Information To Be Shared Within Our Group**

|  |  |  |  |
| --- | --- | --- | --- |
| **Who?** | **What Information?** | **When?** | **How?** |
| Everyone | Everything | At least weekly | Slack, Google Teams, Zoom, Google Drive |

**Information To Be Provided To Other Groups**

|  |  |  |  |
| --- | --- | --- | --- |
| **Who?** | **What Information?** | **When?** | **How?** |
| Adventium Team | Progress on project | Weekly, bi-weekly, and monthly | Weekly or bi-weekly meetings to report progress, ask questions, etc.  Monthly reports are to be sent to Danielle noting progress on the project. |

**Information Needed From Other Groups**

|  |  |  |  |
| --- | --- | --- | --- |
| **Who?** | **What Information?** | **When?** | **How?** |
| Adventium Team | Updates to project requirements | Weekly, bi-weekly, monthly | Regular meetings will give Danielle an opportunity to inform us of possible requirements updates should they come up. |

## 3.3 Deliverables

|  |  |
| --- | --- |
| **#** | **Deliverable** |
| 1 | Source code (Java) |
| 2 | README including installation and running instructions |
| 3 | Demo video |
| 4 | Project specification |
| 5 | Post-Mortem document |
| 6 | Final report |

## 3.4 Project Metrics

|  |  |  |
| --- | --- | --- |
| **Metric** | **Frequency** | **Location** |
| User Story Points from Monthly Report Feedback | End of each month | At the beginning of Individual Sprint (Section 4.5) |

# 4. Requirements (User Stories)

## 4.1 Overall Description

This project is designed to assist in a small portion of the overarching GUMBO project by Adventium Labs. Our group is responsible for creating an OSATE plug-in that translates AADL models from FACE to AGREE models. OSATE stands for Open Source AADL Tool Environment which is an Eclipse-based IDE for AADL modeling. AADL stands for Architecture Analysis and Design Language. AADL is used for modeling software and hardware systems. Future Airborne Capabilities Environment (FACE) models represent system architectures. Lastly, Assume Guarantee Reasoning Environment (AGREE) models assumptions and guarantees for AADL models. OSATE includes a plug-in to translate FACE models to AADL. Our job is to create an OSATE plugin that reads an AADL instance model, recognizes the components from the FACE model, and then translates it to an AGREE model.

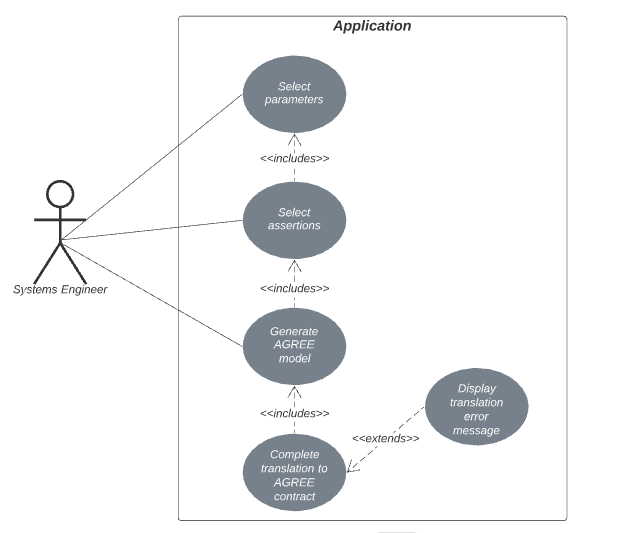
Using Java, we will be creating a user interface that is designed to set specific parameters and limitations to FACE models and accurately translate them into AGREE models. The user interface should be constructed so that user errors are reduced and users can easily learn how to use the application. This could be done using dropdown menus to limit the variability of user input.

To verify that our plug in works properly, we will be using preexisting FACE and AADL files to test that it works according to the requirements elicited from our sponsor. This testing will involve maintaining accurate information in the final AGREE model. Another area we will be testing is the capacity of constraints that a user can enter. If a clear impact on performance appears at a certain boundary, we will implement limiting factors to prevent users from entering more than that limit.

## 4.2 Users and Roles

|  |  |
| --- | --- |
| **User** | **Description** |
| Systems Engineer | There are systems engineers who will be using our plugin in their critical system development using model based engineering. |
| Project Managers | There are project managers on the Adventium team who will keep track of different objectives for the projects they work on and motivate their respective teams. |
| Stakeholder | The stakeholder is the US Army. Stakeholders have an interest in a project or an idea and work with a company to create that project. They are generally the group affected by the project. |

## 4.3 Use Case Diagrams



## 4.4 User Stories (Requirements)

|  |  |  |
| --- | --- | --- |
| **ID** | **Feature** | **Story Points** |
| F1 | Install OSATE environment | 3 |
| F2 | Develop a UI prototype | 3 |
| F3 | As a system engineer, I want to put specifications and constraints on component inputs and outputs, so that the FACE model that has been translated to AADL can be properly extended with AGREE contracts. | 3 |
| F4 | As a project manager, I need translated AGREE contracts to be accurate with all ports, connections, features, and systems from the FACE model accounted for in the AADL model. | 2 |
| F5 | As a project manager, I would like the user interface to be simple and intuitive, so that the rest of the team can use it efficiently. | 2 |
| F6 | As a project manager, I would like the user interface to constrain user input so that it allows for easier translation to an AGREE model and reduces syntax errors. | 2 |
| F7 | As a project manager, I would like the finished product to create and show a syntax error free AGREE model that can extend the AADL model. | 2 |
| F8 | As a project manager, I want the interface to be easily understood and well organized, so that the new users and the stakeholders can pick up on how to use the project quickly. | 1 |
| F9 | As a project manager, I want the interface to both be functional and good looking so that the stakeholders can be satisfied with the outcome of the project. | 1 |
| F10 | As a system engineer, I want the process of generating AGREE contracts to occur within a reasonable amount of time. | 1 |
| F11 | As a system engineer, I want to view documentation of how to install and use the plug-in. | 3 |
| F12 | As a developer of the plug-in, I want to have a repository to collaborate with the other developers. | 3 |

**SPRINT 1**

**Total Estimated User Story Points for Sprint 1: 6**

**Actual Completed User Story Points for Sprint 1:** 6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F1 | Onset | Install OSATE environment with AADL, FACE, and AGREE support plug-ins | C | 3 | 3 | | 100 |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 1.1 | Eclipse environment runs with OSATE dependencies | | “Install Additional OSATE Plugins” appears in “Help” menu for all members | | | | |
| 1.2 | FACE to AADL plugin available | | Option to run this plugin when FACE model is selected for all members | | | | |
| 1.3 | OSATE verification application works on sample models | | Separate OSATE verification application runs for all members | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | Install Eclipse 2020-06 with OSATE | | | | | Eclipse Installer | |
| B | Install FACE to AADL translator | | | | | Eclipse IDE | |
| C | Install OSATE Application | | | | | Danielle Stewart | |
| D | Install Sample Plugin | | | | | Danielle Stewart | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F2 | Onset | Develop a web prototype of the plug-in for demonstration purposes | **C** | **3** | **3** | | **100** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 2.1 | Assumptions page is functional | | Sponsor approves of the features available in web prototype | | | | |
| 2.2 | Guarantees page is functional | | Sponsor approves of the features available in web prototype | | | | |
| 2.3 | Output page is functional | | Sponsor approves of the output available in web prototype | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | Make initial outline identical to Java pop ups | | | | | Riley | |
| B | Make pop up prototype have multiple pages | | | | | Riley | |
| C | Add dropdowns for variables used for constraints | | | | | Riley | |
| D | Add dropdowns for comparators used for constraints | | | | | Riley | |
| E | Aggregate information from assumptions page into output | | | | | Riley | |
| F | Aggregate information from guarantees page into output | | | | | Riley | |
| G | Display output | | | | | Riley | |

**SPRINT 2**

**Total Estimated User Story Points for Sprint 2: 7**

**Actual Completed User Story Points for Sprint 2:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F3 | Onset | Develop an initial version of the plug-in with full constraint adding functionality | **NC** | **3** | **3** | | **85** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 3.1 | UI implemented in Java | | UI satisfies requirements elicited from the prototype demos | | | | |
| 3.2 | Successful iterations through AADL file objects | | Ability to list and aggregate components of an AADL file | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | Translate web UI prototype components into JFrame | | | | | Riley | |
| B | Move plug-in activation to Eclipse Toolbar | | | | | Riley | |
| C | Decide how to iterate through AADL file object | | | | | Charles | |
| D | Display and categorize components in an AADL file object | | | | | Charles | |
| E | Store components in iteration result object | | | | | Charles | |
| F | Wire UI to iteration result object | | | | | Riley & Charles | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F4 | Onset | Store previous AGREE assertions in JSON file | **NC** | **2** | **2** | | **20** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 4.1 | JSON file is written | | Output of the plugin is written to JSON file | | | | |
| 4.2 | AGREE assertion is broken down and described | | JSON file writing separates the AGREE assertion | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | Write to JSON file | | | | | Ansley | |
| B | Write output AGREE annex to JSON file | | | | | Ansley | |
| C | Partition output AGREE annex | | | | | Ansley | |
| D | Write partitioned output to JSON file | | | | | Ansley | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F5 | Onset | Simplify the UI | **NC** | **2** | **2** | | **100** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 5.1 | Layout of components fits flow of plugin | | Sponsor approves of the layout | | | | |
| 5.2 | Labels for inputs of plugin use easy to understand words | | Sponsor approves of label verbiage | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | Determine what needs to be moved | | | | | Team | |
| B | Update the position of inputs | | | | | Riley | |
| C | Determine what verbiage could be clearer | | | | | Team | |
| D | Update the language of labels | | | | | Riley | |

**SPRINT 3**

**Total Estimated User Story Points for Sprint 3: 4**

**Actual Completed User Story Points for Sprint 3:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F6 | Onset | Constrain user input to reduce syntax errors | **NC** | **2** | **2** | | **0** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 6.1 | Assumptions are limited based on selected variable | | Front-end applies comparison filters for assumptions depending on data type | | | | |
| 6.2 | Output of guarantees depends on selected variable | | Front-end applies output limits for the guarantees depending on the data type | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | Data type dependent drop down values are added to the current options for assumptions | | | | | Team | |
| B | Drop down values are updated according to selected variables | | | | | Team | |
| C | Data type dependent drop down values are added to the current options for guarantees | | | | | Team | |
| D | Drop down values are updated according to selected variables | | | | | Team | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F8 | Onset | Polishing UI field instructions are clear | **NC** | **2** | **2** | | **0** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 8.1 | Labels for UI inputs are clear and flow is updated | | Approval from sponsor and those who tested the current UI | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | Testers are selected to assess the current UI | | | | | Team and Adventium | |
| B | Feedback is received from testers | | | | | Adventium and peers | |
| C | UI is updated according to feedback | | | | | Team | |
| D | Approval is received from sponsor | | | | | Sponsor | |

**SPRINT 4**

**Total Estimated User Story Points for Sprint 4: 3**

**Actual Completed User Story Points for Sprint 4:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F9 | Onset | Packaging plug in | **NC** | **1** | **1** | | **0** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 9.1 | Plug-in can be downloaded | | A member of the Adventium team can download the plug-in | | | | |
| 9.2 | Plug-in can be used | | A member of the Adventium team can use the plug-in after downloading | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | Plug-in is formatted for packaging | | | | | Jeremy | |
| B | Plug-in is aggregated into one package | | | | | Jeremy | |
| C | Plug-in is made available for download/made a part of the GUMBO package | | | | | Jeremy | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F10 | Onset | Improve time efficiency of the algorithm. | **NC** | **1** | **1** | | **0** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 10.1 | AADL file iteration and processing occurs within a reasonable amount of time | | Sponsor and testers of plug-in have no qualms with the time of the iteration algorithm | | | | |
| 10.2 | AADL file iteration algorithm is capacity tested | | No extreme time delays result from file iteration | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | File iteration algorithm is capacity tested | | | | | Charles | |
| B | File iteration is updated to be faster if necessary | | | | | Charles | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Added** | **Description** | **Status** | **Story Points** | **Actual Equivalent Story Points** | | **% Completed** |
| F11 | Onset | Create documentation to assist future users | **NC** | **1** | **1** | | **25** |
| **Acceptance Criteria** | | | **Verification** | | | | |
| 11.1 | README document is created | | Instructions for installation and use are included in the README document | | | | |
| 11.2 | Project Specification is finalized | | Project Specification document is finalized | | | | |
| **ID** | **Tasks** | | | | | **Resource** | |
| A | Project Specification is completed | | | | | Team | |
| B | Installation instructions are added to README | | | | | Team | |
| C | Use instructions are added to README | | | | | Team | |

## 4.5 Constraints and Limitations

|  |  |
| --- | --- |
| **Constraint** | **ID** |
| The development environment for working on the Eclipse plugin has proven to be unstable, so it is possible that any member on the team might need to take time to repair their Java and IDE install. | F1 |
| The user interface is limited by the customization offered within Eclipse, so making a polished and intuitive design may be made more difficult. | F2 |
| Once we get through our initial run of sample models, we will need to request additional via email in order to run more sample tests. | F3 |

# 5. Design

## 5.1 Introduction

The goal of the project is to create an Eclipse Plug-in to make the extension of existing AADL files with AGREE assertions easier. Our design accomplishes this by providing a user interface tool with drop down selection of features. There are four areas of focus for development of our implementation: development of front-end user experience, iteration through AADL files, storage of previously created AGREE assertions, and packaging and finalization of the project.

The front-end is built in Java using the JFrame library. It consists of three forms for assumptions, guarantees, and AGREE output. Assumptions are the constraints the variables of each module and its dependencies are expected to have. Guarantees are the expected outputs that come from different behaviors of each module and its dependencies.

The iteration is built in Java using an event listener method as the main method to trigger iteration over an AADL file. This method invokes a recursive method that breaks down a file into its dependent implementation objects. These implementation objects are then stored to a custom Iteration Result Object for the front end to use.

The storage system for previously created AGREE assertions is also built in Java. It saves the assertions to a file that can be used for faster extensions. This will prevent the users from having to remake assertions everytime and will allow them to update previous assertions.

The packaging and finalization of the project will consist of making the plug-in easily accessible and downloadable for members working with the GUMBO project.

## 5.2 Scope

Our project fits into a larger project called GUMBO. This narrows our specialization to just extending AADL models with AGREE assertions. Knowing where we fit into the grand scheme of the project and the purpose of the overall project makes it easy for our team to create a plug-in that meets the requirements necessary.

## 5.3 High-Level Component Design

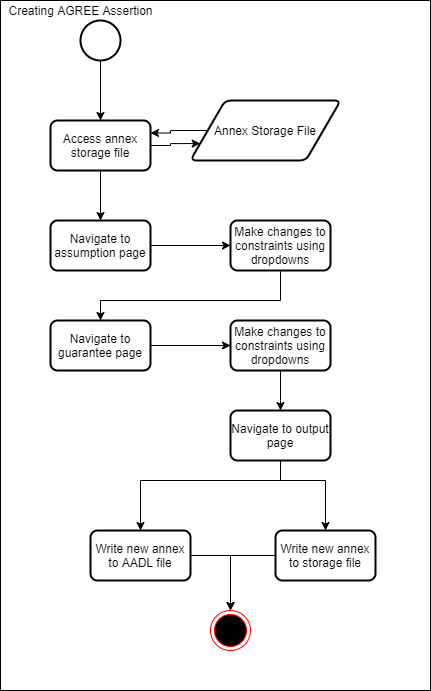
|  |  |  |
| --- | --- | --- |
| **Component** | **Related Requirements** | **Description** |
| GitHub Repo | F12 | Our team has our own repository for storing the necessary documents and files. |
| Eclipse OSATE | F1, F3 | Since our plug-in is an OSATE extension, we work in an Eclipse OSATE environment. |
|  |  |  |
|  |  |  |

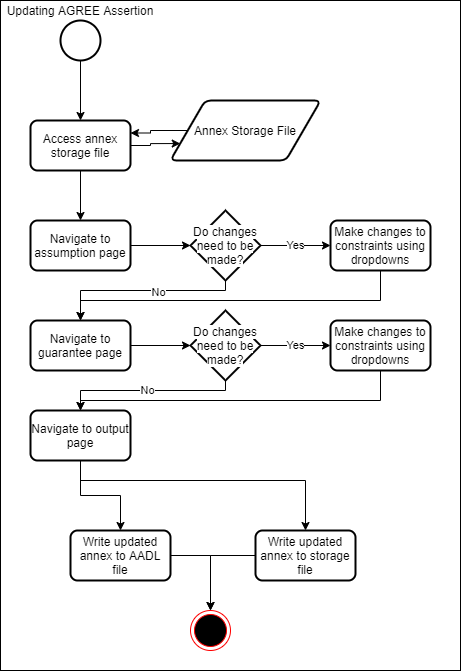
## 

## 5.4 Class Diagram

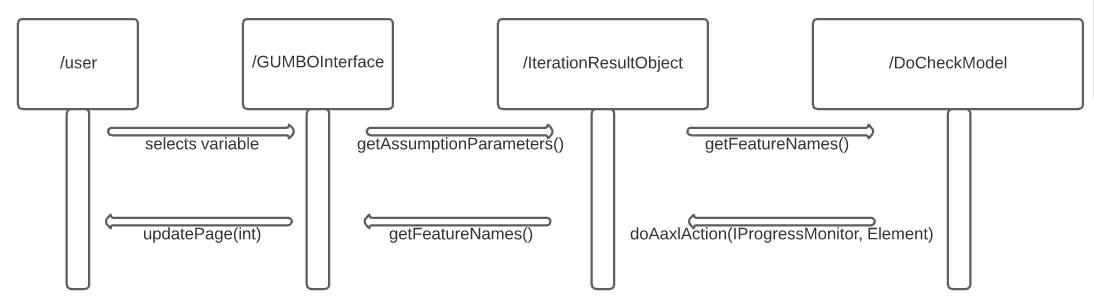
## 

## 5.5 Activity Diagrams





## 5.6 Sequence Diagram



## 5.7 Alternative Designs and Design Rationale

The alternate designs that were considered were different GUI designs. Our goal with the GUI was to make sure that the users would prefer to use our plugin rather than just writing the code themselves. With this goal in mind, we tried multiple GUI drafts which eventually led to our current design. Our current GUI design is simple, clear, and concise. With only three pages, the user can easily add and delete assumptions and guarantees, and copy the AGREE annex shown on the last page.

## 5.8 Data Architecture

Our project does not require access to a database, however we do plan on storing some user data in a JSON file. The JSON file will take the finished assumptions and guarantees from the user interface. The JSON file will save the assumptions and guarantees so that the UI, backend, and users can access them later. A stretch goal is to have the JSON file also save constants, so users can implement constants into their argument easier.

# 6. User Interface

## 6.1 UI Description

The plugin is designed as a pop-up, single-window interface application. It is triggered via a pre-set launch button in the toolbar of Eclipse--set up by Adventium Labs in our starter project. Once this button is fired, the interface that we have created appears and is interactable.

The UI is composed of three separate panels within the pop-up window. They are the Assumptions, the Guarantees, and the Output. Each of these panels correspond to the steps in creating a list of assertions on an AADL model with the AGREE syntax. The first two panels are for statement creation, whereas the last panel is for gathering the generated statements.

For the Assumptions and Guarantees, they are each headed by a large selection box that holds any prior generated statements of their corresponding type. Underneath the selection box are a set of inputs--both free-write text boxes and drop-down boxes--for the user to compose their statements. These inputs provide mechanisms with which users can put in their values freely, but are constrained to only the available inputs. These two panels mirror each other closely, with the Guarantees panel simply having a few extra inputs to accommodate the longer syntax style.

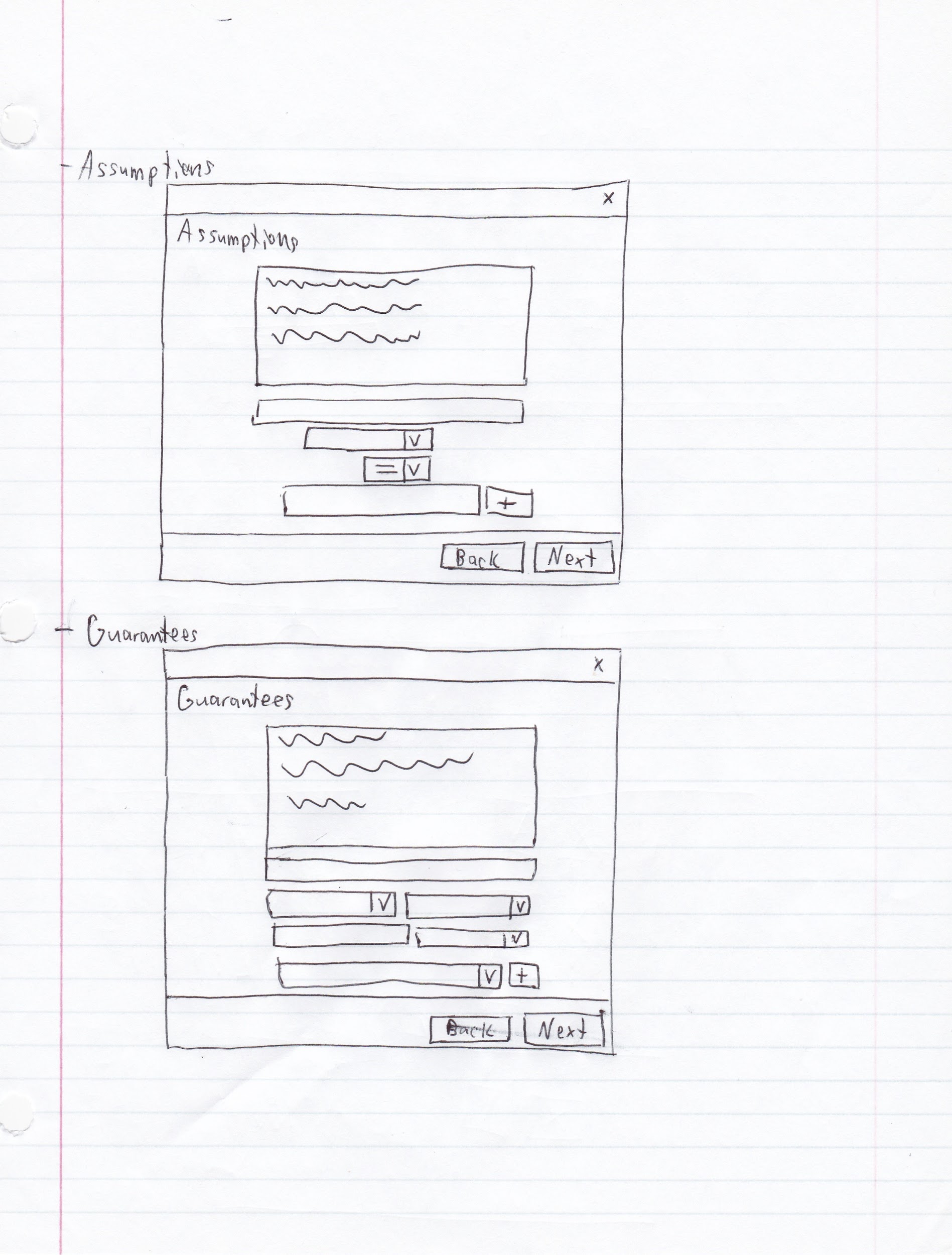
For the Output panel, it is a very simple implementation. Within the panel is a large text area element that holds the entire generated AGREE annex. It is read-only, so users can copy the code manually or click a “Copy to Clipboard” button below for easier use.

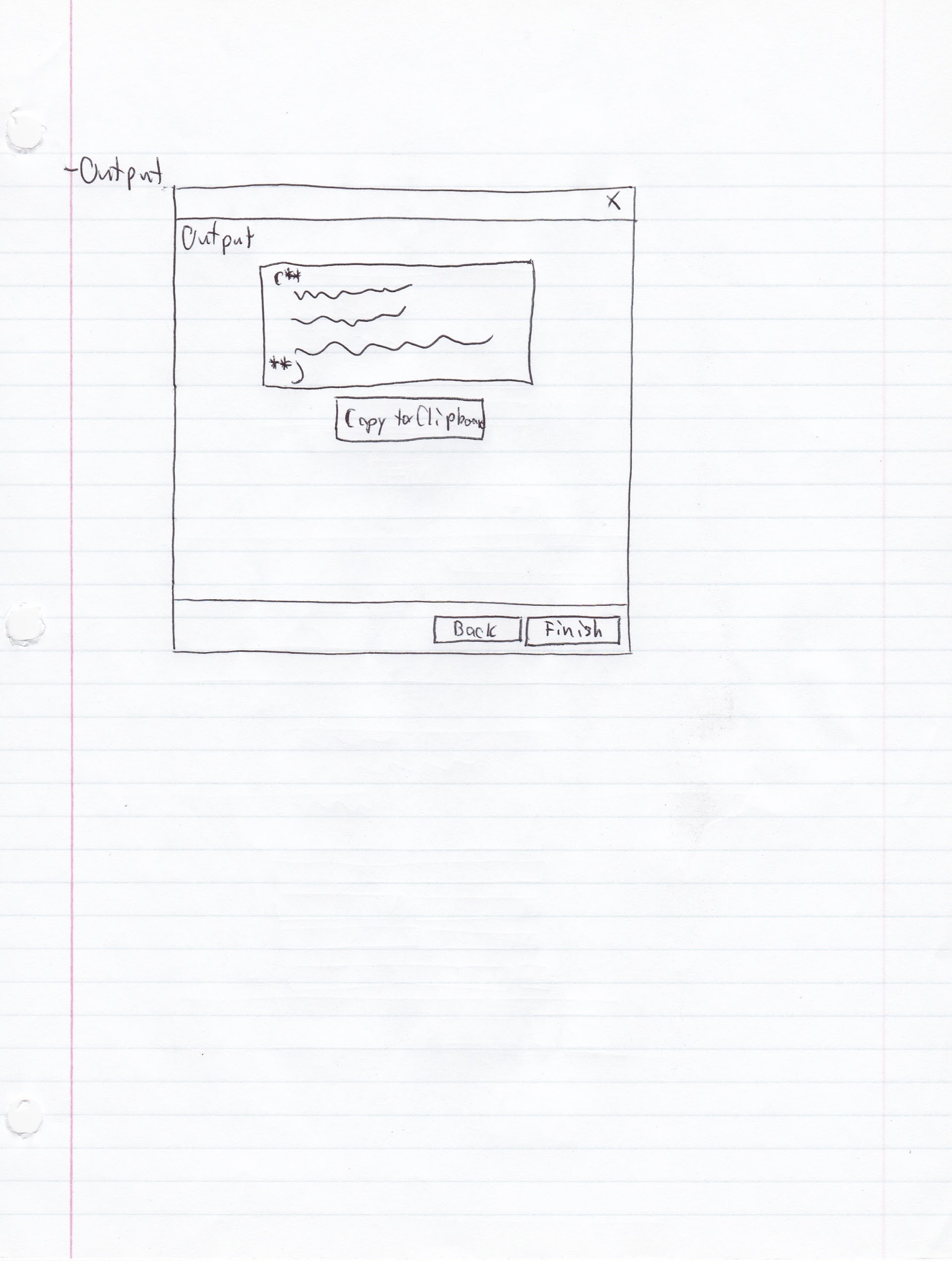
All of the panels contain a footer that holds the navigation buttons. These buttons permit the user to go “Back” or “Next” in the order of panels as they are working. Once the user is on the final panel, the “Next” button transforms into a “Finish” button, and upon click it will close the window.

## 6.2 UI Mockup

The UI was mocked-up in several stages. First, it was sketched in lo-fi paper mockups. After the lo-fi mockups, the design was transferred directly into code for an HTML/CSS prototype. This prototype was . It’s visual style mimics the Eclipse IDE default styling for pop-up windows and plugins, so the visual design could be quickly copied with Java code for the second half of the project.

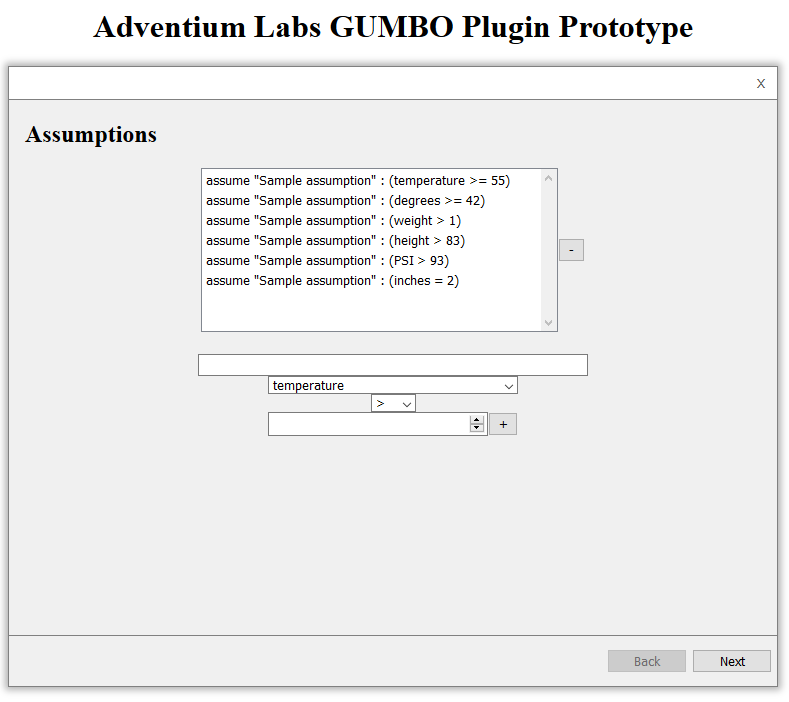
### 6.2.1 Lo-fi Mockups

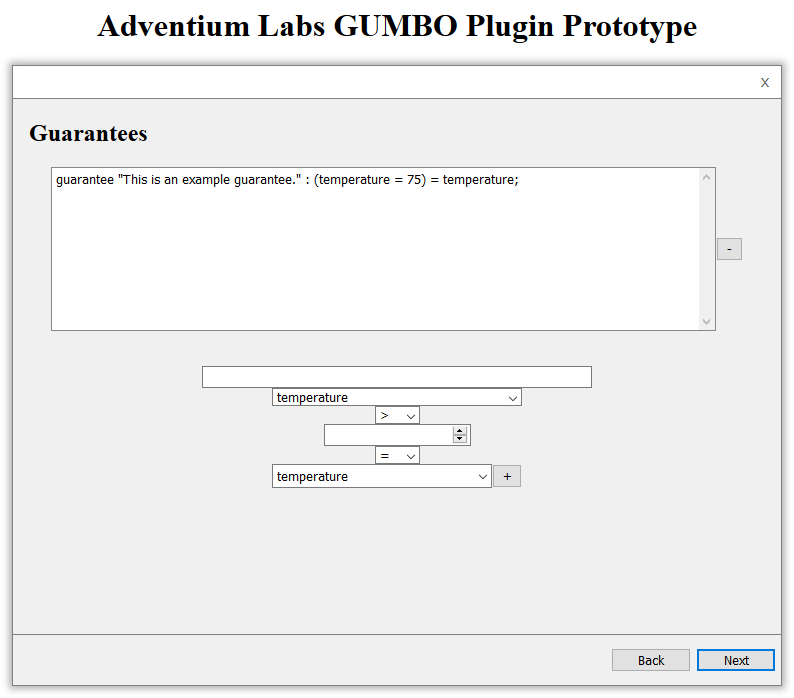


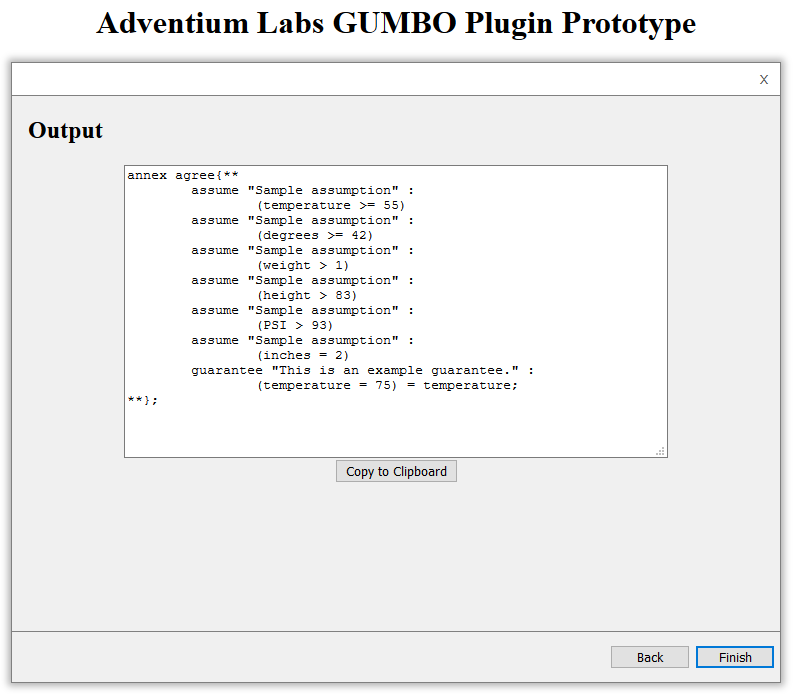


**6.2.2 Hi-fi Mockups/Prototype**

The prototype of the Hi-fi mockups is live at: <https://adventium-gumbo-ui-prototype.netlify.app/>







# 7. Project Closure

## 7.1 Goals / Vision

This primary goal of this project was to create an OSATE plugin for the Eclipse IDE that enables system engineers to speed up their workflow. This plugin would be installed in every developer’s Eclipse environment and cover several actions regarding AADL models. Within the plugin, the user can: read AADL models for their inputs and outputs, display the collected parameters in an Eclipse GUI window, allow the user to input new, constrained assumptions onto the model, and finally generate a copy-pastable AGREE statement for the developer to put into the model they are working on. The plugin’s vision included: being easy to learn, low maintenance with high stability, reduces user cognitive load, and--most importantly--as fast or faster than writing statements manually.

## 7.2 Delivered Solution

The UI for the frontend is satisfactory and fulfills the requirements specified from the beginning of the project. The user can create new assumptions and guarantees and generate a copy-pastable output to extend their models. The backend correctly iterates through a selected model and grabs nearly all of the necessary data to create extensions in the front end. The extension also saves generated statements to local storage for each user to make it easier to go back and make edits as necessary. The entire solution has been packaged up as an independent Eclipse extension, and can be installed locally via Eclipse’s own internal plugin wizard.

## 7.3 Remaining Work

This project has satisfied all of Adventium Labs’ goals set from the beginning of the semester. Adventium and the team worked closely to make sure the project wouldn’t suffer from scope creep and everything would be achievable given the constraints for remote work.

With this in mind, if this project went longer than a single semester, Adventium does have some “stretch goals” they would’ve liked to have seen implemented. The next step for the plugin would be for it to edit the AADL files directly--skipping the copy-paste step from the developer. Adventium has recognized that this would vastly increase the developer hours needed to make it work and chose to pursue performance and stability instead. Therefore, this feature remains as a potential update in the future.

# 8. Definitions and Acronyms

|  |  |
| --- | --- |
| **Term** | **Definition** |
| FACE | **Future Airborne Capabilities Environment:** Model used to describe system architectures |
| AGREE | **Assume Guarantee Reasoning Environment:** Used to model assumptions and guarantees for AADL models |
| OSATE | **Open Source AADL Tool Environment:** A Eclipse-based IDE for AADL modeling |
| AADL | **Architecture Analysis and Design Language:** Used for modeling software and hardware systems |
| GUMBO | **Grand Unified Modeling of Behavioral Operators:** Overarching project aiming to unify cross-section of behavior representations in AADL to support unified behavioral analysis. |