# System and Software Architecture Description (SSAD)

## XL2

## Team No. 10

| Name | Primary Role | Secondary Role |
|---|---|---|
| Kevin Crimi | Prototyper, Builder | Feasibility Analyst |
| Sindhu Nachimuthu | Project Manager, Life Cycle Planner | Builder |
| Ritesh Nanda | Operations Concept Engineer | Reviewer, Tester, Trainer |
| Muthukumaran Dhanapal | Software Architect, Builder | Feasibility Analyst |
| Ted Lee | IIV & V, Tester, Quality Manager | Project Manager |

# Version History

| Date | Author | Version | Changes made | Rationale |
|---|---|---|---|---|
| 10/13/12 | Muthukumaran Dhanapal | 1.0 | • Original template | • Initial draft |
| 10/15/12 | Kevin Crimi | 1.0 | • Changed System Context Section<br>• Changed Artifacts and Information Section<br>• Changed Behavior and Processes Section | • Changed to closer match the template<br>• Generated new diagrams using Visual Paradigm |
| 10/20/12 | Muthukumaran Dhanapal | 1.1 | • Template with section 1 updated | • Document for Draft FCP |
| 10/29/12 | Muthukumaran Dhanapal | 1.2 | • Changed System Context Section<br>• Changed Artifacts and Information Section<br>• Changed Behavior and Processes Section | • Generated new diagrams using Visual Paradigm |
| 11/3/12 | Muthukumaran Dhanapal | 2.0 | • Changed the SSAD to architected agile template<br>• Completed section 2, 3 and 4 | • Document for DCP |
| 11/14/12 | Muthukumaran Dhanapal | 2.1 | • Updated Section 2, 3 and 4 | • Updated SSAD with IV&V inputs |
| 11/24/12 | Muthukumaran Dhanapal | 3.0 | • Updated Section 1, 2 and 3 | • Document for Draft TRR |

# Table of Contents

**System and Software Architecture Description (SSAD) .... i**
**Version History .... ii**
**Table of Contents .... ii**
**Table of Tables .... iv**
**Table of Figures .... vii**

**1. Introduction .... 1**
**1.1 Purpose of the SSAD .... 1**
**1.2 Status of the SSAD .... 1**
**2. System Analysis .... 2**
**2.1 System Analysis Overview .... 2**
**2.1.1 System context .... 2**
**2.1.2 Artifacts and information .... 3**
**2.1.3 Behaviour .... 4**
**2.1.3.1 Enter asset information into the model .... 4**
**2.1.3.2 Produce cash flow analysis .... 10**
**2.1.3.3 Track Budgets .... 11**
**2.1.4 Modes of operation .... 11**
**2.2 System Analysis Rationale .... 12**
**3. Technology-Specific System Design .... 13**
**3.1 Design Overview .... 13**
**3.1.1 System structure .... 13**
**3.1.2 Design Classes .... 14**
**3.1.2.1 GUI class .... 14**
**3.1.2.2 Controller class .... 15**
**3.1.2.3 Report generator class .... 16**
**3.1.3 Process realization .... 17**
**3.2 Design Rationale .... 17**
**4. Architectural Styles, Patterns and Frameworks .... 18**

# Table of Tables

*Table 1: Actors summary* .......... *2*
*Table 2: Artifacts and information summary* .......... *3*
*Table 3: Process description (Set Development phases)* .......... *4*
*Table 4: Typical course of action (Set Development phases)* .......... *4*
*Table 5: Exceptional course of action (Set Development phases)* .......... *5*
*Table 6: Process description (Enter revenue assumptions)* .......... *5*
*Table 7: Typical course of action (Enter revenue assumptions)* .......... *5*
*Table 8: Exceptional course of action (Enter revenue assumptions)* .......... *5*
*Table 9: Process description (Adjust the cost distributions)* .......... *6*
*Table 10: Typical course of action (Adjust the cost distribution)* .......... *6*
*Table 11: Exceptional course of action (Adjust the cost distribution)* .......... *6*
*Table 12:  Process description (Allocate sources of funding)* .......... *7*
*Table 13: Typical course of action (Allocate sources of funding)* .......... *7*
*Table 14: Exceptional course of action (Allocate sources of funding)* .......... *7*
*Table 15: Process Description (Sava populated model)* .......... *8*
*Table 16: Typical course of action (Sava populated model)* .......... *8*
*Table 17: Exceptional course of action (Sava populated model)* .......... *8*
*Table 18: Process description (Update a model)* .......... *9*
*Table 19: Typical course of action (Update a model)* .......... *9*
*Table 20: Exceptional course of action (Update a model)* .......... *9*
*Table 21: Process description (Generate cash flow report)* .......... *10*
*Table 22: Typical course of action (Generate cash flow report)* .......... *10*
*Table 23: Exceptional course of action (Generate cash flow report)* .......... *10*
*Table 24: Process description (Generate budget reports)* .......... *11*
*Table 25: Typical Course of Action (Generate Budget Reports – New Model)* .......... *11*
*Table 26: Alternate Course of Action (Generate Budget Reports – Saved Model)* .......... *11*
*Table 27: Software Component Description* .......... *13*
*Table 28: GUI Class Description* .......... *14*
*Table 29: Controller Class Description* .......... *15*

*Table 30: Report generator Class Description* ........ *16*
*Table 31: Architectural Styles, Patterns, and Frameworks* ........ *18*

# Table of Figures

*Figure 1: System Context diagram* ........ *2*
*Figure 2: Artifacts and information diagram* ........ *3*
*Figure 3: Process diagram* ........ *4*
*Figure 4: Software Component class diagram* ........ *13*
*Figure 5: GUI class diagram* ........ *14*
*Figure 6: Controller class diagram* ........ *15*
*Figure 7: Report generator class diagram* ........ *16*
*Figure 8: Process realization diagram* ........ *17*

# 1. Introduction

## 1.1 Purpose of the SSAD

The purpose of the SSAD is to document the technical architecture, results of the analysis and design of the project XL2 (Team No. 10). This document will serve the developers of the system as a reference.

## 1.2 Status of the SSAD

The current version of the SSAD is 3.0 and is a part of draft TRR package. Therefore this document reflects the current understanding and architecture of the developing system. This document is in architected agile template with all the sections fully completed.

# 2. System Analysis

## 2.1 System Analysis Overview

The main purpose of XL2 project is to provide real estate companies and real estate brokers an easy way of doing complex real estate analysis such as sensitivity analysis, quality analysis, budget analysis, pro forma cash analysis etc. This project will provide a controller implemented in java that will allow input/output and calculation functionality to the client's current real estate models implemented in Excel files. This controller will improve the quality of all the real estate analysis mentioned above, reduce the margin of errors, standardize process, increase learning curve as well as reduce the cost to the company.

### 2.1.1 System Context

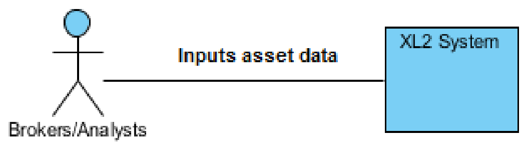

**Figure 1: System Context diagram**

**Table 1: Actors summary**

| Actor | Description | Responsibilities |
|---|---|---|
| Brokers/analysts | Performs the analysis of the proposed property using the XL2 system | Compiles the costs and required development phases from the real estate developers and the funding sources from the investors in order to prepare the analysis and generate reports using the XL2 models |

### 2.1.2 Artifacts & Information

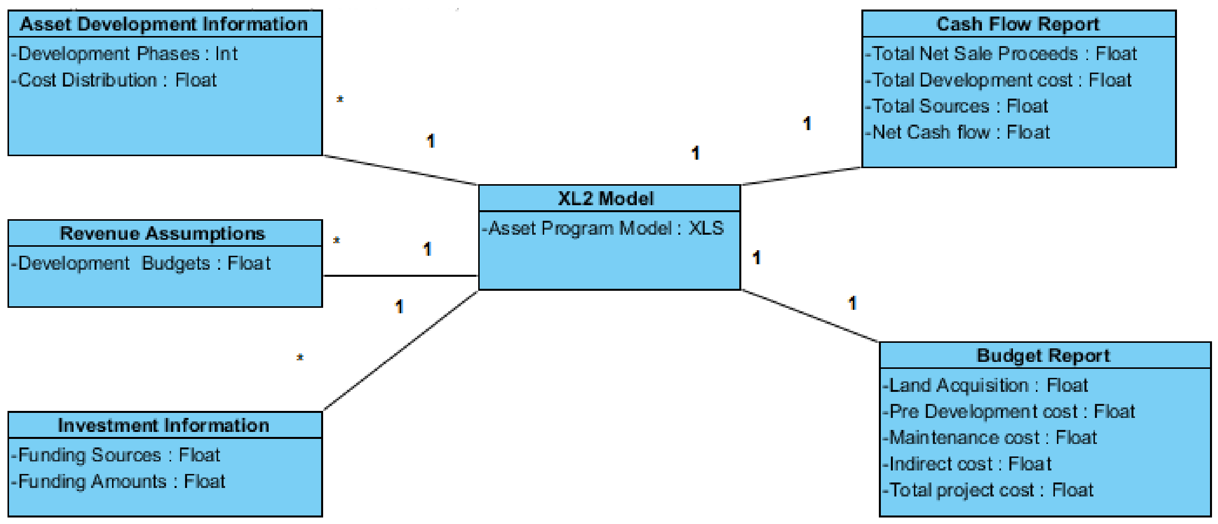
Asset Development Information
-Development Phases : Int
-Cost Distribution : Float
Revenue Assumptions
-Development Budgets : Float
Investment Information
-Funding Sources : Float
-Funding Amounts : Float
XL2 Model
-Asset Program Model : XLS
Cash Flow Report
-Total Net Sale Proceeds : Float
-Total Development cost : Float
-Total Sources : Float
-Net Cash flow : Float
Budget Report
-Land Acquisition : Float
-Pre Development cost : Float
-Maintenance cost : Float
-Indirect cost : Float
-Total project cost : Float
*
1
*
1
*
1
1
1
1
1

**Figure 2: Artifacts and information diagram**

**Table 2: Artifacts and information summary**

| Artifact | Purpose |
|---|---|
| Asset Development Information | Convey to the analyst the costs and efforts required for the development of the asset |
| Revenue assumptions | Adjust budgets across each phase of the development process |
| Investment Information | Convey to the analyst the details of the different funding sources |
| XL2 Model | The populated template .xls file using the above input information to generate the calculated values based on the client's model in order to generate reports |
| Budget report | This will enable the analyst to track the budget of the project over time. |
| Cash flow report | This will be the primary output report from the application. This will display an amount of values generated by the model in an easy to read format. |

### 2.1.3 Behavior

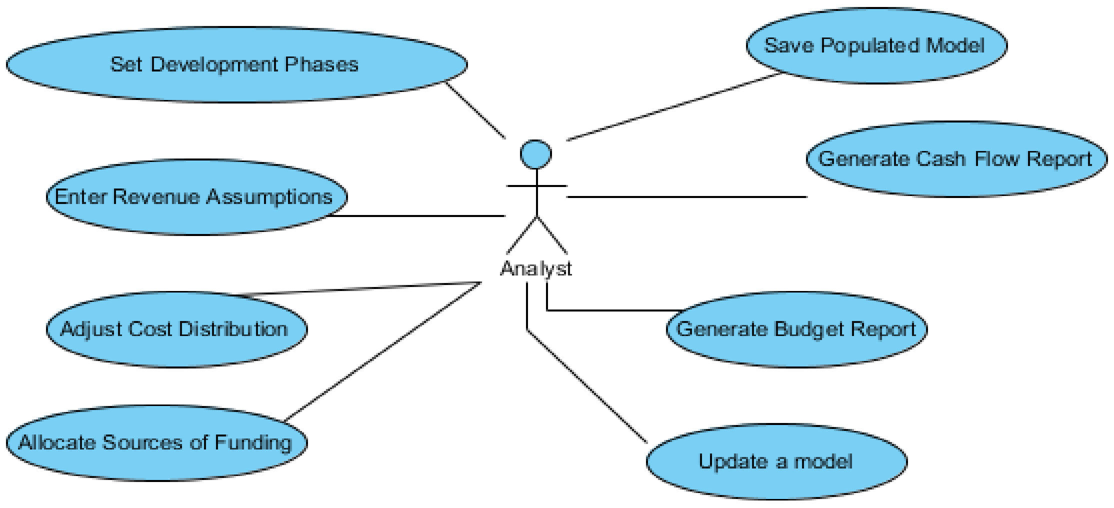

**Figure 3: Process diagram**

#### 2.1.3.1 Enter Asset Information into the Model

##### 2.1.3.1.1 UC-1Set Development Phases

**Table 3: Process description (Set Development phases)**

| | |
|---|---|
| **Identifier** | Set Development Phases |
| **Purpose** | Determine the necessary development phases which will be modeled and how long they will be |
| **Requirements** | WC_1306, WC_1309, WC_1589,  WC_1592, WC_1596 |
| **Development Risks** | None |
| **Pre-conditions** | The analyst is in the Development Phase GUI tab |
| **Post-conditions** | The model will reflect the amount and length of development phases the analyst has chosen |

**Table 4: Typical course of action (Set Development phases)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Analyst enters which phases are required for this project into the GUI | |
| **2** | | The system populates the phase names onto the client's Excel templates |

**Table 5: Exceptional course of action (Set Development phases)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Analyst enters exceptional phases (like undefined phase names) into the GUI | |
| **2** | | A window pops up stating that there was an error |

#### 2.1.3.1.2 UC-2Enter Revenue Assumptions

**Table 6: Process description (Enter revenue assumptions)**

| | |
|---|---|
| **Identifier** | Enter Revenue Assumptions |
| **Purpose** | Input the revenue assumptions into the model in the appropriate phases |
| **Requirements** | WC_1306, WC_1309, WC_1589, WC_1593 |
| **Development Risks** | None |
| **Pre-conditions** | The analyst is in the Revenue assumption GUI tab |
| **Post-conditions** | The system model is populated with the revenue assumptions entered by the analyst |

**Table 7: Typical course of action (Enter revenue assumptions)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Analyst enters Revenue Assumptions in the respective phases into the GUI | |
| **2** | | System model is populated with the revenue assumptions |

**Table 8: Exceptional course of action (Enter revenue assumptions)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Analyst enters exceptional Revenue Assumptions (undefined type) into the GUI | |
| **2** | | A window pops up stating that there was an error |

##### 2.1.3.1.3 UC-3 Adjust the Costs Distributions

**Table 9: Process description (Adjust the cost distributions)**

| | |
|---|---|
| **Identifier** | Adjust the Costs Distribution |
| **Purpose** | Populate the model with the appropriately distributed costs in the appropriate phases |
| **Requirements** | WC_1306, WC_1309, WC_1589, WC_1591, WC_1594, WC_1595 |
| **Development Risks** | None |
| **Pre-conditions** | The analyst is in the Costs Distributions GUI tab |
| **Post-conditions** | The model has been populated with the costs distribution in the appropriate phases |

**Table 10: Typical course of action (Adjust the cost distribution)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst enters the amount of the cost in the desired phase | |
| **2** | | The model is populated with the costs distributed in the appropriate development phases |

**Table 11: Exceptional course of action (Adjust the cost distribution)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst enters the exceptional amount of the cost (undefined type) in the desired phase | |
| **2** | | A window pops up stating that there was an error |

#### 2.1.3.1.4 UC-4 Allocate Sources of Funding

**Table 12: Process description (Allocate sources of funding)**

| | |
|---|---|
| **Identifier** | Allocate Sources of Funding |
| **Purpose** | To generate different reports through the program model |
| **Requirements** | WC_1306, WC_1309, WC_1589, WC_1597, WC_1598, WC_1600, |
| **Development Risks** | None |
| **Pre-conditions** | The analyst is in the Sources of Funding GUI tab |
| **Post-conditions** | The system model is populated with the sources of funding |

**Table 13: Typical course of action (Allocate sources of funding)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst specifies the sources of funding and the amount | |
| **2** | | The model is populated with the sources of funding |

**Table 14: Exceptional course of action (Allocate sources of funding)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst specifies the exceptional sources of funding and the amount (undefined type) | |
| **2** | | A window pops up stating that there was an error |

#### 2.1.3.1.5 UC-5 Save Populated Model

**Table 15: Process Description (Sava populated model)**

| | |
|---|---|
| **Identifier** | Save Populated Model |
| **Purpose** | Save the populated model for recall later |
| **Requirements** | WC_1315, WC_1586, WC_1590, WC_1601 |
| **Development Risks** | None |
| **Pre-conditions** | The analyst is in the File menu tab |
| **Post-conditions** | The system model is capable of being recalled since it is stored in a complete .xls Excel file |

**Table 16: Typical course of action (Sava populated model)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst saves the completed model | |
| **2** | | An .xls Excel file is generated from the model template to save the populated data |

**Table 17: Exceptional course of action (Sava populated model)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst saves the completed model | |
| **2** | | If there is an error in saving the file then a window pops up stating that there was an error in saving the file. |

#### 2.1.3.1.6 UC-6 Update a Model

**Table 18: Process description (Update a model)**

| | |
|---|---|
| **Identifier** | Update a Model |
| **Purpose** | Modify and update a currently existing file |
| **Requirements** | WC_1315, WC_1585, WC_1586, WC_1587, WC_1590, WC_1601 |
| **Development Risks** | None |
| **Pre-conditions** | There is an already existing populated program model |
| **Post-conditions** | The system model is capable of being recalled since it is stored in a complete .xls Excel file |

**Table 19: Typical course of action (Update a model)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst opens the already existing file and updates it either my manually tweaking the file or modify the values through the GUI | |
| **2** | | The .xls Excel file updates itself based on the modification that the user does |

**Table 20: Exceptional course of action (Update a model)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst saves the completed model | |
| **2** | | If there is an error then a window pops up stating that there was an error in saving the file. |

### 2.1.3.2 Produce Cash Flow Analysis

#### 2.1.3.2.1 UC-7 Generate Cash Flow Report

**Table 21: Process description (Generate cash flow report)**

| | |
|---|---|
| **Identifier** | Generate Cash Flow Report |
| **Purpose** | To provide investment details as an input to the program model |
| **Requirements** | WC_1312 |
| **Development Risks** | None |
| **Pre-conditions** | The XL2 model must be completely populated |
| **Post-conditions** | A report is generated for a reader-friendly view of the analysis |

**Table 22: Typical course of action (Generate cash flow report)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst populates the model | |
| **2** | The analyst requests a cash flow analysis report | |
| **3** | | A PDF of the cash flow analysis report is generated in the form specified by the client |

**Table 23: Exceptional course of action (Generate cash flow report)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst populates the model | |
| **2** | The analyst requests a cash flow analysis report | |
| **3** | | If there is a problem in PDF generation then an error message will pop up |

#### 2.1.3.3 Track Budgets

##### 2.1.3.3.1 UC-8 Generate Budget Reports

**Table 24: Process description (Generate budget reports)**

| | |
|---|---|
| **Identifier** | Generate Budget Reports |
| **Purpose** | This will enable the analyst, investor or real estate developer to track their asset budgets over time |
| **Requirements** | WC_1308, WC_1312 |
| **Development Risks** | None |
| **Pre-conditions** | The model must be completely populated |
| **Post-conditions** | A report in the form specified by the client is generated in order to display the budget. This can be revisited and updated over time |

**Table 25: Typical Course of Action (Generate Budget Reports – New Model)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst requests a budget report for a newly populated model | |
| **2** | | A budget report is generated in the form specified by the client |

**Table 26: Alternate Course of Action (Generate Budget Reports – Saved Model)**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | The analyst loads a previously saved model | |
| **2** | The analyst updates the model based on the time and resources which have changed | |
| **3** | | An updated budget report is generated in the form specified by the client |

### 2.1.4 Modes of Operation

XL2 works in only one mode of operation.

## 2.2 System Analysis Rationale

XL2 project is comprised of two subsystems:
1. Java swing GUI implementation which populates the client's Excel file
2. Generation of different reports based on client's populated Excel file

The first subsystem provides the user with a GUI that takes all the inputs that the user enters and populates the appropriate cells in the client's Excel file. This GUI is implemented using Java swing library. The entered values are linked to the Excel file through apache POI library which enables us to represent the file as a "workbook" object which can be written and read to different cells and can finally be written as an .xls Excel file

The second subsystem generates different reports like cash flow report, budget report etc. using the populated Excel file (from subsystem 1) and produces these reports in the native Excel form.

Another important requirement of this project is that the entire XL2 project should be in the form of .exe file. So this is done by using Excelsior Jet.

# 3. Technology-Specific System Design

## 3.1 Design Overview

### 3.1.1 System Structure

Since XL2 project's end result is going to be an executable file, there is no hardware component involved in it. Hence there is no need for a hardware component diagram.

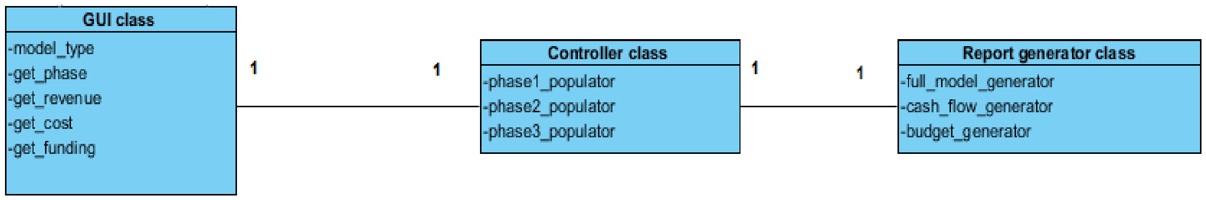

**Figure 4: Software Component class diagram**

**Table 27: Software Component Description**

| Software Component | Description |
|---|---|
| GUI class | Interacts with the user and gets different data like model type, phase, revenue, cost and sources of funding. This uses java swing library in java for implementing the GUI |
| Controller class | Populates each and every entry of the client's Excel file with the assimilated data. This uses apache POI library in java. A phase populator attribute populates that particular phase with revenue, cost and funding details |
| Report generator class | Generates cash flow report, budget report etc. |

## 3.1.2 Design Classes

### 3.1.2.1 GUI class

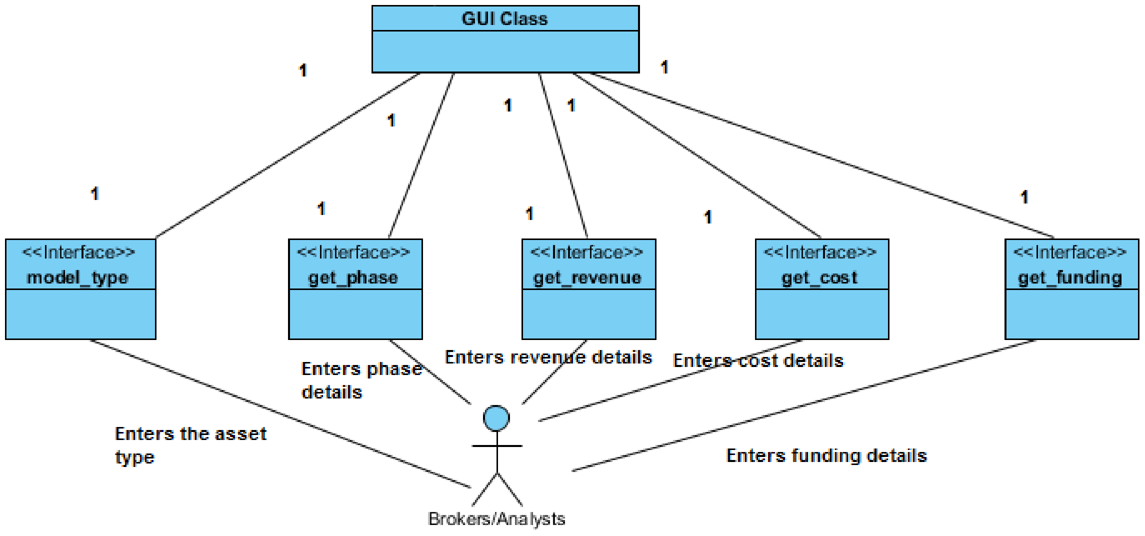

**Figure 5: GUI class diagram**

**Table 28: GUI Class Description**

| Entity | Type | Description |
|---|---|---|
| GUI Class | Class | Assimilates all the data entered by the broker/analyst |
| model_type | Interface | Gets asset/model type from the analyst |
| get_phase | Interface | Gets phase detail from the analyst |
| get_revenue | Interface | Gets revenue details from the analyst |
| get_cost | Interface | Gets cost details from the analyst |
| get_funding | Interface | Gets sources of funding from the analyst |
| Brokers/Analysts | Actor | Enters the asset details into the XL2 system |

### 3.1.2.2 Controller Class

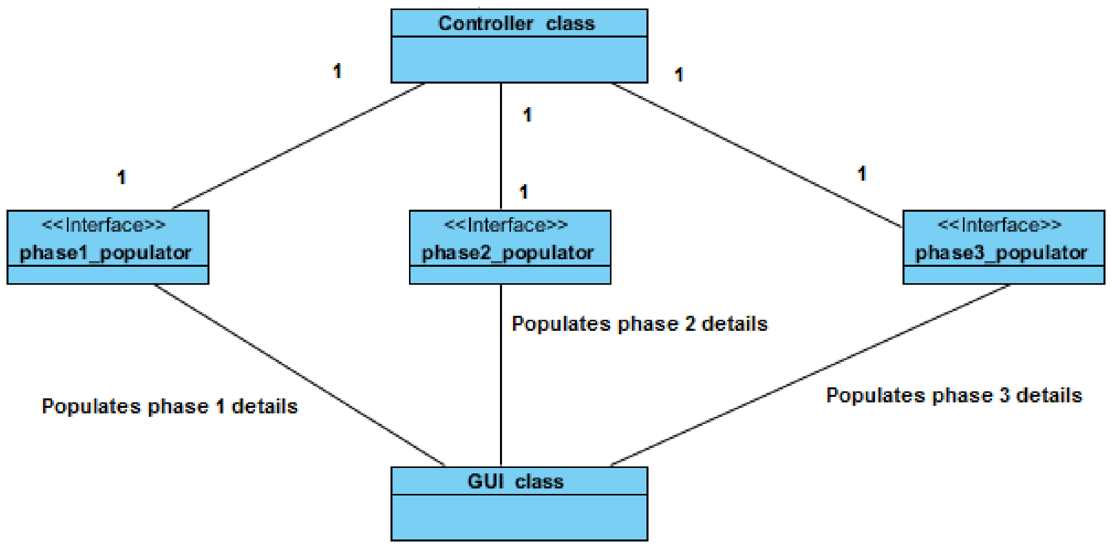

**Figure 6: Controller class diagram**

**Table 29: Controller Class Description**

| Entity | Type | Description |
|---|---|---|
| Controller class | Class | Populates asset input data entered by the analyst into the Client’s Excel file template |
| phase1_populator | Interface | Gets phase 1 asset details |
| phase2_populator | Interface | Gets phase 2 asset details |
| phase3_populator | Interface | Gets phase 3 asset details |
| GUI class | Class | Assimilates all the asset details entered by the analyst |

### 3.1.2.3 Report generator class

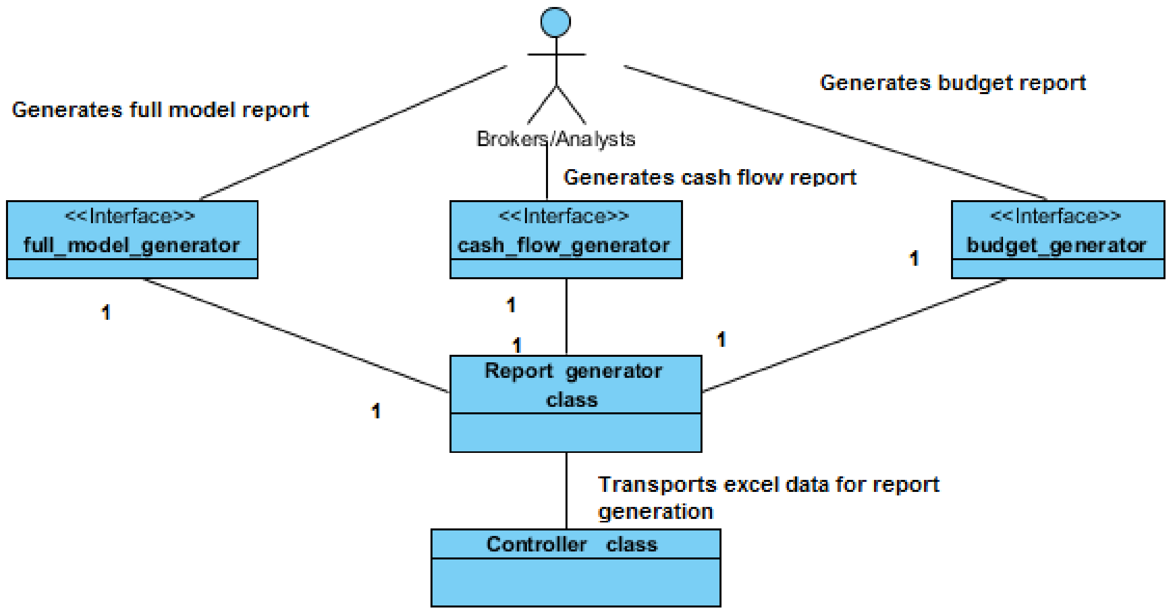

**Figure 7: Report generator class diagram**

**Table 30: Report generator Class Description**

| Entity | Type | Description |
|---|---|---|
| Controller class | Class | Populates client's Excel file with the input asset data |
| Report generator class | Class | Generates different types of reports |
| full_model_generator | Interface | Generates full model report |
| cash_flow_generator | Interface | Generates Cash flow report |
| budget_generator | Interface | Generates budget report |
| Broker/Analysts | Actor | End beneficiary of all the report generation |

### 3.1.3 Process Realization

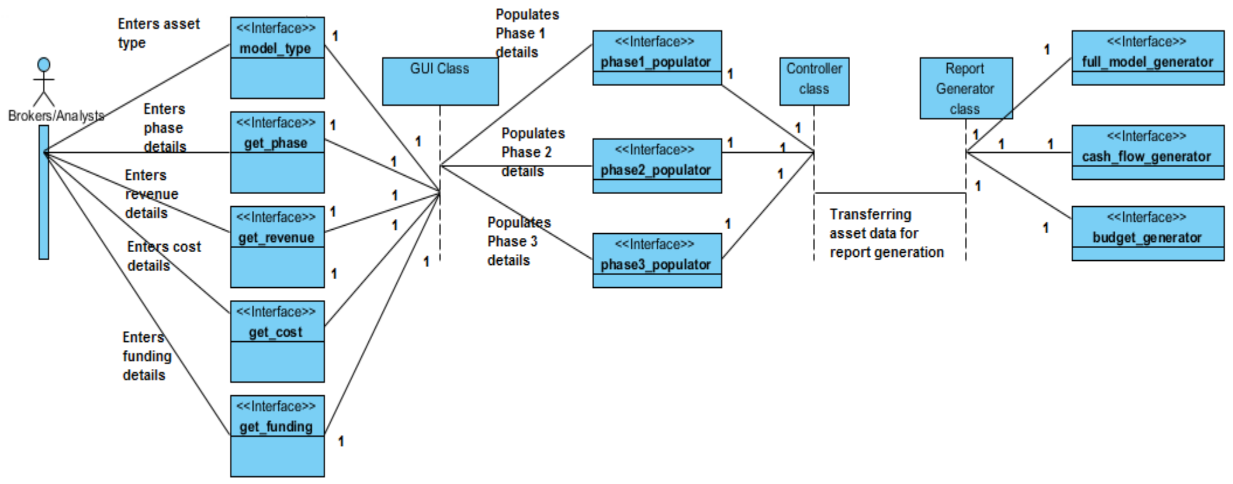
Brokers/Analysts
Enters asset type
Enters phase details
Enters revenue details
Enters cost details
Enters funding details
<<Interface>> model_type
<<Interface>> get_phase
<<Interface>> get_revenue
<<Interface>> get_cost
<<Interface>> get_funding
GUI Class
Populates Phase 1 details
Populates Phase 2 details
Populates Phase 3 details
<<Interface>> phase1_populator
<<Interface>> phase2_populator
<<Interface>> phase3_populator
Controller class
Report Generator class
Transferring asset data for report generation
<<Interface>> full_model_generator
<<Interface>> cash_flow_generator
<<Interface>> budget_generator
1

**Figure 8: Process realization diagram**

## 3.2 Design Rationale

As depicted in the software component diagram, the XL2 project has 3-tier architecture. This architectural style makes future changes to the system easier. The three tiers are listed below:

- GUI Class
- Controller Class
- Report Generator Class

The GUI Class allows the user to enter the asset data through a GUI implemented through java swing library. It takes model type, cost details, sources of funding, revenue details and phase details as input from the user.

The Controller Class uses the asset data obtained from the previous stage and populates the different cells of client's Excel file template, which is used to generate different reports based on client's program model. This uses apache POI library. Apache POI library has classes that represents the Excel files as workbooks. In workbook we have different sheets and there are rows and cells in each and every sheet which can be modified (either read or written).

The Report generator class produces different reports such as cash flow report, budget report, full build report etc. based on the populated Client's Excel file template and produces these reports in .xls format.

# 4. Architectural Styles, Patterns and Frameworks

**Table 31: Architectural Styles, Patterns, and Frameworks**

| Name | Description | Benefits, Costs, and Limitations |
|---|---|---|
| Three tier Architecture | The 3-tier architecture separates the application into 3 different layers: GUI Class, Controller Class, and Report generator Class. Each layer can only communicate with the layer immediately above or below itself (as explained in the design rationale). | This architecture is an Excellent example of separation of concerns principle: each layer is independent of the other. Therefore, any internal changes the layer do not affect other layers. As a result, a layer can be completely replaced with a different set of classes if required.<br>The main disadvantage of this architecture is the increase in the size of the project. |