TABLE OF CONTENT

INTRODUCTION

1 – Purpose

The purpose of this document is to present and document the requirements identified and agreed upon with both customer and developers. This document will go a step further and analyze the system to be developed with the object oriented analysis approach. The approach followed is suggested by the following book "Use Cases Combined with Booch,UML and OMT"Putnam P Texel.
This document should be informative for both customer and developer regarding the capabilities of the system.

2 – Scope

The objective of this report is to define a clear, unambiguous and testable requirement engineering for a software project. This document should clearly communicate to the development team and customer the objectives of this project and how those objectives will be met, while providing a framework of the high level design. This following software requirement is developed under the request of the customer and for his personal use only. The software should be able to establish one-dimensional relationship between objects and components. The main utility would be to use this software to have a good relational over view of certain components of different domain.

3 – References

"Use Cases Combined with Booch, UML and OMT" Putnam P Texel

4 – Overview

This document will give an overview of all the requirements of this project. The object-oriented analysis is the approach that the development team will follow. Each phase will comport am introductory section to explain what the team is developing and on what purpose.

OVERALL DESCRIPTION

The product we are developing must have the capability to function as the primary tool for drawing and identifying logical, mathematical, physical, chemical relationship between different objects. It must contain the necessary functionality to be able to automatically identify the relationships or to accept any relationship that the user may suggest. The application must also have the functionality needed for showing, collecting, analyzing and reporting the data
The product must support different domains It must be flexible enough to support the use of user generated data and analyze calculations. The user must be able to define the standards by which the software will interact with him. The user must have available within the application the ability to define and control the instances of historical data from past projects.
The product to be produced will interface with pre-existing probabilistic, mathematical chemical, physical, science-related relationship and metrics to identify obvious relations.

1 - Product perspective

This product should operate in different platforms offering a multiple user capability. Each user would have his own username and password. In case of successful development the product should change the way the users develop relationship diagrams by offering an abstract overview that can be viewed within different levels of details.

2 - Product functions

The main functionality of the product is to offer relationship detection and drawing capability to a user to put ideas together and try to find relationship diagrams. The user shall have the ability to use the different relationship domain that already exists.

3 - User characteristics

The users are highly educated users. The product is developed for the an unknown company under the request of Dr Drysdale. The development may assume that all the users will have a respectable level of education.

4 - Constraints

The system should run without interruption. A secure copy of each open file will be saved in case of a major power shortage. The secure copy shall be updated every 15 minutes so that the user at the most would loose only 15 minutes of work in the worst case scenario.

The system should offer multi user capability. If one user request use of a certain file that another user is using, the second user access to that specific file would be read only. A user could have the choice not to share his work with other users.

5 - Assumptions and dependencies

There are no assumptions or dependencies at this point

SPECIFIC REQUIREMENT

1 - External interface requirements

The software shall run under a Windows 95 or later version operating system. The system shall be exportable to any Sun Micro-system station using Unix operating system.

User interfaces

The user shall have a graphical user interface with available option. The user interface shall contain short cut buttons and an undo functionality.

2 - Hardware interfaces

Below is the required hardware interface under which the software should run and meet it's expected deadlines.

- At least 120 MHz Processor
- At least 16 M-Ram

3 - Software interfaces

The system should run on a Windows NT environment.

4 - Communications interface

The system shall have the following abilities

- To support a network base installation
- Multiple users
- Exchange of shared data
- Exchange of data between users
- Provide privileged users functionality
- Administrative privileges for administrator
- Network lock out
- Personalized profiles
- Private space for each user

FUNCTIONAL REQUIREMENTS

1 - Object Handling

1.1 - The system shall allow the user to name the selected object.
1.2 - The user shall be able to add attributes to an object.
1.3 - The user shall be able to delete any arguments.
1.4 - The user shall be able to modify the values of the attributes.
1.5 - The user shall be able to breakdown object to different objects by clicking on it.
1.6 - The system shall be able to do error checking for variables consistency (food or Food).
1.7 - The system shall be able to allow the user to select different type of objects.
1.8 - The user shall be able to create objects.
1.9 - The relationship shall be used from general modules
1.91 - The user shall be able to link object manually.
1.92 - The user shall be able to draw arrows between object
1.93 - The objects shall be transferable from one model to another

2 - Relationships Handling

2.1 - The user shall not be able to create a relation if the units are inconsistent.
2.2 - The user shall define the relationship.
2.3 - The user shall be able to define the relationship direction
2.4 - The user shall be able to modify any relationship
2.5 - The user shall be able to add a relationship to a link.
2.6 - The mathematical relationships shall be given according to the names of the objects
2.7 - The user shall be able to provide the relationship formula in both directions
2.8 - The system shall be able to identify relationship between non-directly related objects
2.9 - The user shall be able to save the actual association.

3 – Domain

3.1 - The system shall provide different application domain. (Physics, mathematics, chemical at least).
3.2 - The user shall be able to add new application domains (library).
3.3 - The library shall comport default values.
3.4 - The system shall be able to find the relationship formulas between directly related objects

4 – Interim

4.1 - The system shall warn the user in case of errors.
4.2 - The system shall check the unit.
4.3 - The user shall be able to open an existing file
4.4 - The system shall produce reports: text and graphics.
4.5 - The report shall be alphabetical and/or relational.
4.6 - The system shall have an undo function.
4.7 - The system shall keep a database of all the modifications introduced with the related dates.
4.8 - The system shall be able to show the sequence of all the modifications and display the evolution of the changes.

NON FUNCTIONAL REQUIREMENT

The system shall have a graphical user interface.
The system shall be used in a Windows environment.
The system shall be a multiple user application.
The user shall be able to login on the system
The system shall check if the username and password match.
The system shall capture text, numbers, graphics, and icons.

PERFORMANCE REQUIREMENTS

There is no performance requirement at this time.

DESIGN CONSTRAINTS

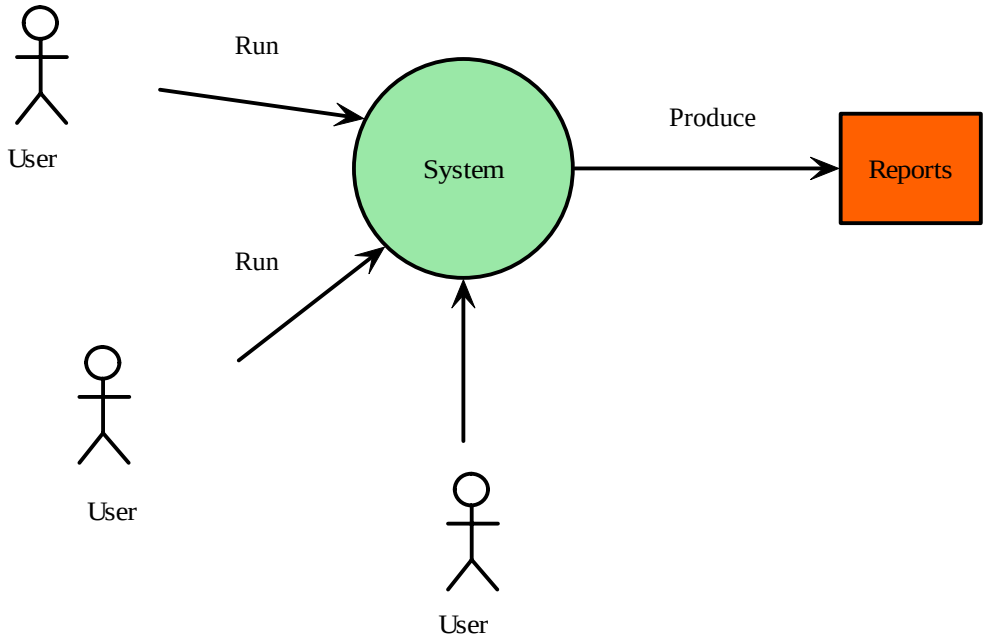
The documentation related to design constraint are in the next pages. A disciplined approach for an Object Oriented Analysis was the objective.

PHASE 1: Requirement Engineering

1 – Context Diagram

Definition:

A Context diagram is a diagram consisting of one circle (or ellipse), which represents the system, that is typically placed in the center of the diagram. Rectangles are used to represent entities, both hardware and software, that are external, finally, a pair of double lines represents any data store that is required.



2 – Capture the shall statements

The purpose of this activity is to produce an initial Requirement Trace Matrix (RTM) that contains the entire set of sentences from the system specification, and any other agreed upon document that contains the world “shall”.

Entry	System Specification Text
1	The system shall have a graphical user interface.
2	The system shall be used in a Windows environment.
3	The system shall support multiple users.
4	The user shall be able to create new objects.
5	The system shall allow the user to add existing objects.
6	The system shall be able to allow the user to select different type of objects.
7	The system shall allow the user to name the selected object.
8	The user shall be able to add attributes to an object.
9	The user shall be able to modify any attributes of any object.
10	The user shall be able to delete any attributes of any object.
11	The user shall be able to add object inside other objects.
12	The user shall be able to delete any object.
13	The user shall be able to breakdown an object into several different objects that shall be considered as attributes of the main object.
14	The system shall provide different application domain. (Physics, mathematics, chemical at least).
15	The user shall be able to add new application domain (Library).
16	The library shall be initialized with default values.
17	The user shall be able to add new relationship to any library.
18	The user shall be able to see the different libraries of the system and their contents.
19	The user shall define the relationship.
20	The user shall be able to define the relationship direction.
21	The user shall be able to modify any relationship.
22	The user shall be able to link object manually.
23	The user shall be able to add a relationship to any graphically linked objects.
24	The user shall be able to draw arrows between object.
25	The user shall be able to provide the relationship formula in both direction.
26	The system shall be able to find the relationship formulas between directly related objects.
27	The mathematical relationships shall be given according to the names of the objects.
28	The system shall be able to identify relationship between non-directly related objects.
29	The user shall be able to use relationship from libraries.
30	The object shall be transferable from one model to another.

31	The user shall be able to open an existing file.
32	The user shall be able to save the actual.
33	The user shall only accede file for which he has write and/or read authorization.
34	The system shall produce reports in textual and graphics format.
35	The report shall be alphabetical and/or relational.
36	The system shall run the simulation before producing the report.
37	The system shall have an undo function.
38	The system shall keep a database of all the modifications introduced with the related dates.
39	The system shall be able to show the sequence of all the modifications.
40	An authorized user shall be able to login in the system.
41	The system shall check if the username and password are in the authorized user list.
42	The system shall be able to do error checking for variable consistency (food or Food).The system shall warn the user in case of errors.
43	The system shall warn the user in case of errors.
44	The user shall be able to simulate and verify the consistency of the relationship.

3 – Allocate and Prioritize requirements

The purpose of allocating the requirements is to begin to define clearly what requirement is to begin to define clearly what requirement are the responsibility of software and what requirements are the responsibility of hardware.
The purpose of prioritizing the requirements is to allocate the Use Case to reasonable development schedule. That is allocating each one to a build. A build is a specification of a software functionality to be developed by a specific date.

Entry	System Specification Text	Type	Build
1	The system shall have a graphical user interface.	SWC	1
2	The system shall be used in a Windows environment.	SWC	1
3	The system shall support multiple users.	SWC	1
4	The user shall be able to create new objects.	SW	1
5	The system shall allow the user to add existing objects.	SW	2
6	The system shall be able to allow the user to select different type of objects.	SW	1
7	The system shall allow the user to name the selected object.	SW	1
8	The user shall be able to add attributes to an object.	SW	2
9	The user shall be able to modify any attributes of any object.	SW	2
10	The user shall be able to delete any attributes of any object.	SW	2
11	The user shall be able to add object inside other objects.	SW	1
12	The user shall be able to delete any object.	SW	1
13	The user shall be able to breakdown an object into several different objects that shall be considered as attributes of the main object.	SW	2
14	The system shall provide different application domain. (Physics, mathematics, chemical at least).	SW	4
15	The user shall be able to add new application domain (Library).	SW	4
16	The library shall be initialized with default values.	SW	4
17	The user shall be able to add new relationship to any library.	SW	4
18	The user shall be able to see the different libraries of the system and their contents.	SW	4
19	The user shall define the relationship.	SW	3
20	The user shall be able to define the relationship direction.	SW	3
21	The user shall be able to modify any relationship.	SW	3
22	The user shall be able to link object manually.	SW	3
23	The user shall be able to add a relationship to any graphically linked objects.	SW	3
24	The user shall be able to draw arrows between object.	SW	3

25	The user shall be able to provide the relationship formula in both direction.	SW	3
26	The system shall be able to find the relationship formulas between directly related objects.	SW	3
27	The mathematical relationships shall be given according to the names of the objects.	SW	3
28	The system shall be able to identify relationship between non-directly related objects.	SW	4
29	The user shall be able to use relationship from libraries.	SW	4
30	The object shall be transferable from one model to another.	SW	4
31	The user shall be able to open an existing file.	SW	1
32	The user shall be able to save the actual.	SW	1
33	The user shall only accede file for which he has write and/or read authorization.	SW	
34	The system shall produce reports in textual and graphics format.	SW	5
35	The report shall be alphabetical and/or relational.	SW	5
36	The system shall run the simulation before producing the report.	SW	5
37	The system shall have an undo function.	SW	5
38	The system shall keep a database of all the modifications introduced with the related dates.	SW	5
39	The system shall be able to show the sequence of all the modifications.	SW	5
40	An authorized user shall be able to login in the system.	SW	1
41	The system shall check if the username and password are in the authorized user list.	SW	1
42	The system shall be able to do error checking for variable consistency (food or Food).The system shall warn the user in case of errors.	SW	4
43	The system shall warn the user in case of errors.	SW	4
44	The user shall be able to simulate and verify the consistency of the relationship.	SW	4

PHASE 2: System Object Oriented Analysis, Static View

1 – Identify Use Cases

The purpose of this activity is to extract the software requirement from the RTM and reformat these requirements to aid in the discovery of potential Categories. The reformatting activity transition a shall sentence into Use Case format.
A Use Case is a statement of functionality required of the software. A use case is written in a specific format. The specific format for a Use Case is

UC_Actor_Action_Subject

Entry	System Specification Text	Type	Build	Use Case Name	Scenario
1	The system shall have a graphical user interface.	SWC	1	N/A	N/A
2	The system shall be used in a Windows environment.	SWC	1	N/A	N/A
3	The system shall support multiple users.	SWC	1	N/A	N/A
4	The user shall be able to create new objects.	SW	1	UC01_User_Manages_Object	1
5	The system shall allow the user to add existing objects.	SW	2	UC01_User_Manages_Object	6
6	The system shall be able to allow the user to select different type of objects.	SW	1	UC01_User_Manages_Object	3,4,5
7	The system shall allow the user to name the selected object.	SW	1	UC01_User_Manages_Object	1
8	The user shall be able to add attributes to an object.	SW	2	UC01_User_Manages_Object	2
9	The user shall be able to modify any attributes of any object.	SW	2	UC01_User_Manages_Object	4
10	The user shall be able to delete any attributes of any object.	SW	2	UC01_User_Manages_Object	3
11	The user shall be able to add object inside other objects.	SW	1	UC01_User_Manages_Object	6
12	The user shall be able to delete any object.	SW	1	UC01_User_Manages_Object	7
13	The user shall be able to breakdown an object into several different objects that shall be considered as attributes of the main object.	SW	2	UC01_User_Manages_Object	5
14	The system shall provide different application domain. (Physics, mathematics, chemical at least).	SW	4	UC02_User_Manages_Libraries	1
15	The user shall be able to add new application domain (Library).	SW	4	UC02_User_Manages_Libraries	2
16	The library shall be initialized with default values.	SW	4	UC02_User_Manages_Libraries	1
17	The user shall be able to add new relationship to any library.	SW	4	UC02_User_Manages_Libraries	3
18	The user shall be able to see the different libraries of the system and their contents.	SW	4	UC02_User_Manages_Libraries	1

19	The user shall define the relationship.	SW	3	UC03_User_Manages_Relationships	2
20	The user shall be able to define the relationship direction.	SW	3	UC03_User_Manages_Relationships	2
21	The user shall be able to modify any relationship.	SW	3	UC03_User_Manages_Relationships	3
22	The user shall be able to link object manually.	SW	3	UC03_User_Manages_Relationships	1
23	The user shall be able to add a relationship to any graphically linked objects.	SW	3	UC03_User_Manages_Relationships	2
24	The user shall be able to draw arrows between object.	SW	3	UC03_User_Manages_Relationships	1
25	The user shall be able to provide the relationship formula in both direction.	SW	3	UC03_User_Manages_Relationships	2
26	The system shall be able to find the relationship formulas between directly related objects.	SW	3	UC03_User_Manages_Relationships	4
27	The mathematical relationships shall be given according to the names of the objects.	SW	3	UC03_User_Manages_Relationships	4
28	The system shall be able to identify relationship between non-directly related objects.	SW	4	UC03_User_Manages_Relationships	4
29	The user shall be able to use relationship from libraries.	SW	4	UC03_User_Manages_Relationships	2
30	The object shall be transferable from one model to another.	SW	4	UC04_System_Manages_Files	3
31	The user shall be able to open an existing file.	SW	1	UC04_System_Manages_Files	1
32	The user shall be able to save the actual.	SW	1	UC04_System_Manages_Files	2
33	The user shall only accede file for which he has write and/or read authorization.	SW		UC04_System_Manages_Files	1,2
34	The system shall produce reports in textual and graphics format.	SW	5	UC05_System_Produces_Report	1,2
35	The report shall be alphabetical and/or relational.	SW	5	UC05_System_Produces_Report	1,2
36	The system shall run the simulation before producing the report.	SW	5	UC05_System_Produces_Report	1,2
37	The system shall have an undo function.	SW	5	UC06_System_Manages_Historical_Data	1
38	The system shall keep a database of all the modifications introduced with the related dates.	SW	5	UC06_System_Manages_Historical_Data	2
39	The system shall be able to show the sequence of all the modifications.	SW	5	UC06_System_Manages_Historical_Data	2
40	An authorized user shall be able to login in the system.	SW	1	UC07_User_Logs_To_System	1
41	The system shall check if the username and password are in the authorized user list.	SW	1	UC07_User_Logs_To_System	1
42	The system shall be able to do error checking for variable consistency (food or Food).The system shall warn the user in case of errors.	SW	4	UC08_User_Runs_Simulation	1
43	The system shall warn the user in case of errors.	SW	4	UC08_User_Runs_Simulation	1
44	The user shall be able to simulate and verify the consistency of the relationship.	SW	4	UC08_User_Runs_Simulation	1

2 – Develop Scenarios

The purpose of a Scenario is to provide the operational concept behind a Use Case.

A Scenario is a formatted description of the steps required for the completion of a use case. A scenario consists of the text that represents the concept of how an operator interacts with the software to achieve the desired result. For a Scenario that does not have operator interaction. The scenario describes the sequence of software actions required to complete the specified functionality.

Use Case 01: UC01_USER_MANAGES_OBJECT– Scenario 1

Overview:

This Use Case demonstrates the user ability to manage objects.
This scenario demonstrates the user ability to create a new object. This object can be a text, a number, a graphic or an icon. This object has a name.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on create object	1. System asks for the name of the object
User types name of object	1. System asks for type of object(text, number, graphic, icon) using preexisting libraries.
User enters type of object	1. System asks for content of object
User type or select object content	1. System creates the object and display it

Scenario Notes:

Object content is name of the file for graphic and icon. Otherwise it is typed directly.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

1. File does not exist when asking for content.
2. File is corrupted.

Use Cases Utilized:

UC02_USER_MANAGES_LIBRARIES

Timing Constraints:

None.

Use Case 01: UC01_USER_MANAGES_OBJECT– Scenario 2

Overview:

This Use Case demonstrates the user ability to manage objects.
This scenario demonstrates the user ability to add an attribute to an existing object. This object can be a text, a number, a graphic or an icon. This object has a name.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.
4. At least one object must already exist.

Scenario:

ACTION	SOFTWARE REACTION
User right clicks on object(text, number, graphic, or icon)	1. System highlights the object as selected. 2. System popup a menu of a command related to object.
User clicks on add attribute	1. System asks for the name of the attribute.
User entres name of attribute	1. Sytem asks for the content of attribute.
User enters typeof attibute.	1. System creates the object and display it.

Scenario Notes:

None.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

UC02_USER_MANAGES_LIBRARIES

Timing Constraints:

None.

Use Case 01: UC01_USER_MANAGES_OBJECT– Scenario 3

Overview:

This Use Case demonstrates the user ability to manage objects.
This scenario demonstrates the user ability to delete an attribute to an existing object. This object can be a text, a number, a graphic or an icon. This object has a name.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.
4. At least one object has already one attribute.

Scenario:

ACTION	SOFTWARE REACTION
User right clicks on object(text, number, graphic, or icon)	1. System highlights the object as selected 2. System popup a menu of a command related to object
User clicks on delete attribute	1. System asks for attribute to delete.
User selects attribute	1. System deletes attribute and refresh display.

Scenario Notes:

None.

Post Condition:

1. System is ready for input.
2. Object updated.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 01: UC01_USER_MANAGES_OBJECT– Scenario 4

Overview:

This Use Case demonstrates the user ability to manage objects.
This scenario demonstrates the user ability to modify an attribute to an existing object. This object can be a text, a number, a graphic or an icon. This object has a name.

Precondition:
1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User right clicks on object(text, number, graphic, or icon)	1. System highlights the object as selected 2. System popup a menu of commands related to object
User clicks on modify attribute	1. System asks for attribute to modify.
User selects attribute	1. System asks for content of attribute using preexisting libraries.
User type or select object content	1. System modifies the attribute and refresh display.

Scenario Notes:

None.

Post Condition:

1. System is ready for input.
2. Object updated.

Required GUI:

1. Main Display.

Exceptions:

1. File does not exist when asking for content.
2. File is corrupted.

Use Cases Utilized:

UC02_USER_MANAGES_LIBRARIES

Timing Constraints:

None.

Use Case 01: UC01_USER_MANAGES_OBJECT– Scenario 5

Overview:

This Use Case demonstrates the user ability to manage objects.
This scenario demonstrates the user ability to breakdown an object to different objects. This object can be a text, a number, a graphic or an icon. This object has a name.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User double clicks on a object	1. The system breakdowns the selected object into the different objects that compose the selected object.

Scenario Notes:

System does not do anything if the object is not composed of other objects.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 01: UC01_USER_MANAGES_OBJECT– Scenario 6

Overview:

This Use Case demonstrates the user ability to manage objects.
This scenario demonstrates the user ability to add object inside another object.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User right clicks on object(text, number, graphic, or icon)	1. System highlights the object as selected 2. System popup a menu of commands related to object
User clicks on Add Object	1. System asks for object to add.
User selects object	1. System adds object inside the selected object and refresh display

Scenario Notes:

None

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 01: UC01_USER_MANAGES_OBJECT– Scenario 7

Overview:

This Use Case demonstrates the user ability to manage objects.
This scenario demonstrates the user ability to delete an object. This object can be a text, a number, a graphic or an icon. This object has a name.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User right clicks on object(text, number, graphic, or icon)	1. System highlights the object as selected 2. System popup a menu of commands related to object
User clicks on Delete Object	1. System asks for confirmation
User confirms	1. System delete object and refresh display.

Scenario Notes:

None.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 02: UC02_USER_MANAGES_LIBRARIES– Scenario 1

Overview:

This Use Case demonstrates the user ability to manage libraries.
This scenario demonstrates the user ability to see the different libraries installed on the system.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on Show libraries	1. System list the libraries installed.
User clicks on a library listed	1. System display formulas and default values stored in the library

Scenario Notes:

System should at least contained the following libraries: physics, mathematics, chemical.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 02: UC02_USER_MANAGES_LIBRARIES– Scenario 2

Overview:

This Use Case demonstrates the user ability to manage libraries.
This scenario demonstrates the user ability to add a library in the system.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on Add library	1. System asks for library file.
User select library file	1. System install the library

Scenario Notes:

None.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

1. The file selected is not a library.
2. The library is already installed.
3. The file does not exist.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 02: UC02_USER_MANAGES_LIBRARIES– Scenario 3

Overview:

This Use Case demonstrates the user ability to manage libraries.
This scenario demonstrates the user ability to add a new relationship to any library installed on the system.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User right click on a relationship	1. System pops up a contextual menu.
User clicks on Add to Library	1. System asks to select a library.
User selects a library	1. System adds the relationship to the library.

Scenario Notes:

System should at least contain the following libraries: physics, mathematics, chemical.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 03: UC03_USER_MANAGES_RELATIONSHIPS– Scenario 1

Overview:

This Use Case demonstrates the user ability to manage relationships between objects.
This scenario demonstrates the user ability to link two objects.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.
4. At least two objects exist.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on Link	1. System switch to drawing Link mode.
User drags a line from one object to another	1. System draw a line connecting the two objects.

Scenario Notes:

None.

Post Condition:

1. System is ready for input.
2. Objects are linked.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 03: UC03_USER_MANAGES_RELATIONSHIPS– Scenario 2

Overview:

This Use Case demonstrates the user ability to manage relationships between objects.
This scenario demonstrates the user ability to define relationship between two objects.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.
4. The user defined at least one link.

Scenario:

ACTION	SOFTWARE REACTION
User right clicks on a link	1. System popup a command menu related to the link.
User clicks on define relationship	1. System asks user to define relationship.
User types in relationship	1. System adds relationships to the link.

Scenario Notes:

The relationship can be an existing relationship stored in a library or can be a new relationship.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

UC02_USER_MANAGES_LIBRARIES

Timing Constraints:

None.

Use Case 03: UC03_USER_MANAGES_RELATIONSHIPS– Scenario 3

Overview:

This Use Case demonstrates the user ability to manage relationships between objects.
This scenario demonstrates the user ability to modify an existing relationship between two objects.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.
4. The user defined at least one relationship.

Scenario:

ACTION	SOFTWARE REACTION
User right clicks on a link	1. System popup a command menu related to the link.
User clicks on modify relationship	1. System asks user to define the new relationship.
User types in relationship	1. System modifies relationships to the link.

Scenario Notes:

The relationship can be an existing relationship stored in a library or can be a new relationship.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

UC02_USER_MANAGES_LIBRARIES

Timing Constraints:

None.

Use Case 03: UC03 USER MANAGES RELATIONSHIPS– Scenario 4

Overview:

This Use Case demonstrates the user ability to manage relationships between objects.
This scenario demonstrates the system ability to display relationship formula between two objects. These objects can be either directly related or non directly related.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.
4. The user defined at least two objects.

Scenario:

ACTION	SOFTWARE REACTION
User presses CTRL and clicks on two different objects	1. System highlights the objects as selected.
User clicks on See relationship	1. System display relationship formula between two objects.

Scenario Notes:

If there is no relationship between the two selected objects, the system displays no relationships. The mathematical relationship will be given accordingly to the names of the objects.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 04: UC04_SYSTEM_MANAGES_FILES– Scenario 1

Overview:

This use case demonstrates the user ability to manage any projects (new or existing).
This scenario demonstrates the user ability to open an existing file.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on Load file	1. System asks user to select the file to open.
User selects an existing file	1. System loads file to memory and display objects and relationships.

Scenario Notes:

None.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

1. File does not exist.
2. File is corrupted.
3. The user does not have read/write access to the file.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 04: UC04_SYSTEM_MANAGES_FILES– Scenario 2

Overview:

This use case demonstrates the user ability to manage any projects (new or existing).
This scenario demonstrates the user ability to save a file.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on Save file	1. System asks user to type the name of the file.
User types the name of the file	1. System saves file.

Scenario Notes:

If a file already exists with name typed, the user is prompted to overwrites the file.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

1. Not enough disk place.
2. The user does not have write access.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 04: UC04_SYSTEM_MANAGES_FILES– Scenario 3

Overview:

This use case demonstrates the user ability to manage any files (new or existing).
This scenario demonstrates the user ability to transfer an object from one file to another.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on Import object	1. System asks user to type the name of the file.
User types the name of the file	1. System lists the objects contained in the file. 2. System asks to select an object.
User select the object to transfer	1. System transfers the object to the current file.

Scenario Notes:

None.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 05: UC05_SYSTEM_PRODUCES_REPORT– Scenario 1

Overview:

This use case shows the report generation.
This scenario demonstrates the system ability to produce alphabetical report using text and graphics.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on produce alphabetical report	1. For each objects and relationships, system checks consistency and units by running simulation. 2. Systems print out the alphabetical report.

Scenario Notes:

If system find an error, the user is prompted to correct it.

Post Condition:

1. System is ready for input.
2. The report is printed.

Required GUI:

1. Main Display.

Exceptions:

1. Printer is not available.
2. The system has at leat found an error.

Use Cases Utilized:

UC08_USER_RUNS_SIMULATION

Timing Constraints:

None.

Use Case 05: UC05_SYSTEM_PRODUCES_REPORT– Scenario 2

Overview:

This use case shows the report generation.
This scenario demonstrates the system ability to produce relational report using text and graphics.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on produce relational report	1. For each objects and relationships, system checks consistency and units by running simulation. 2. Systems print out the relational report.

Scenario Notes:

If system finds an error, the user is prompted to correct it.

Post Condition:

1. System is ready for input.
2. The report is printed.

Required GUI:

1. Main Display.

Exceptions:

1. Printer is not available.
2. The system has at leat found an error.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 06: UC06_SYSTEM_MANAGES_HISTORAL_DATA– Scenario 1

Overview:

This use case show the system ability to manage the user's set of actions.
This scenario demonstrates the system ability to undo the last modifications.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on undo	1. System undoes the last modification.

Scenario Notes:

This step can be repeated as long as the file is not at the same state when it was last saved.

Post Condition:

1. System is ready for input.
2. System returns to the previous state.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 06: UC06_SYSTEM_MANAGES_HISTORAL_DATA– Scenario 2

Overview:

This use case show the system ability to manage the user's set of actions.
This scenario demonstrates the system ability to display the last modifications of the file.

Precondition:

1. The system is powered up.
2. The software is running.
3. The user opened a file.
4. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on History	1. System displays the entire modification history of the current file.

Scenario Notes:

None.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

None.

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 07: UC07_USER_LOGS_TO_SYSTEM– Scenario 1

Overview:

This use case shows the ability of the system to log an user to the system
This scenario demonstrates the system ability to log a user to the system.

Precondition:

1. The system is powered up.
2. The software is running..
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on Log	1. System asks user for Username and Password
User types Username and password	1. System Verify Username and password.

Scenario Notes:

If username and password don't match, user is asked to check his password and username.

Post Condition:

1. System is ready for input.
2. The user is logged.

Required GUI:

1. Main Display.

Exceptions:

1. Username and password don't match

Use Cases Utilized:

None.

Timing Constraints:

None.

Use Case 08: UC08_USER_RUNS_SIMULATION– Scenario 1

Overview:

This use case demonstrates the user ability to run a simulation.
This scenario demonstrates the user ability to check the file for errors.

Precondition:

1. The system is powered up.
2. The software is running.
3. The system is ready for input.

Scenario:

ACTION	SOFTWARE REACTION
User clicks on Run Simulation	1. System checks for variable consistency. 2. System checks for relationship consistency. 3. System computes new values. 4. System refreshes the display

Scenario Notes:

If an error is detected user is prompted to correct the error.

Post Condition:

1. System is ready for input.

Required GUI:

1. Main Display.

Exceptions:

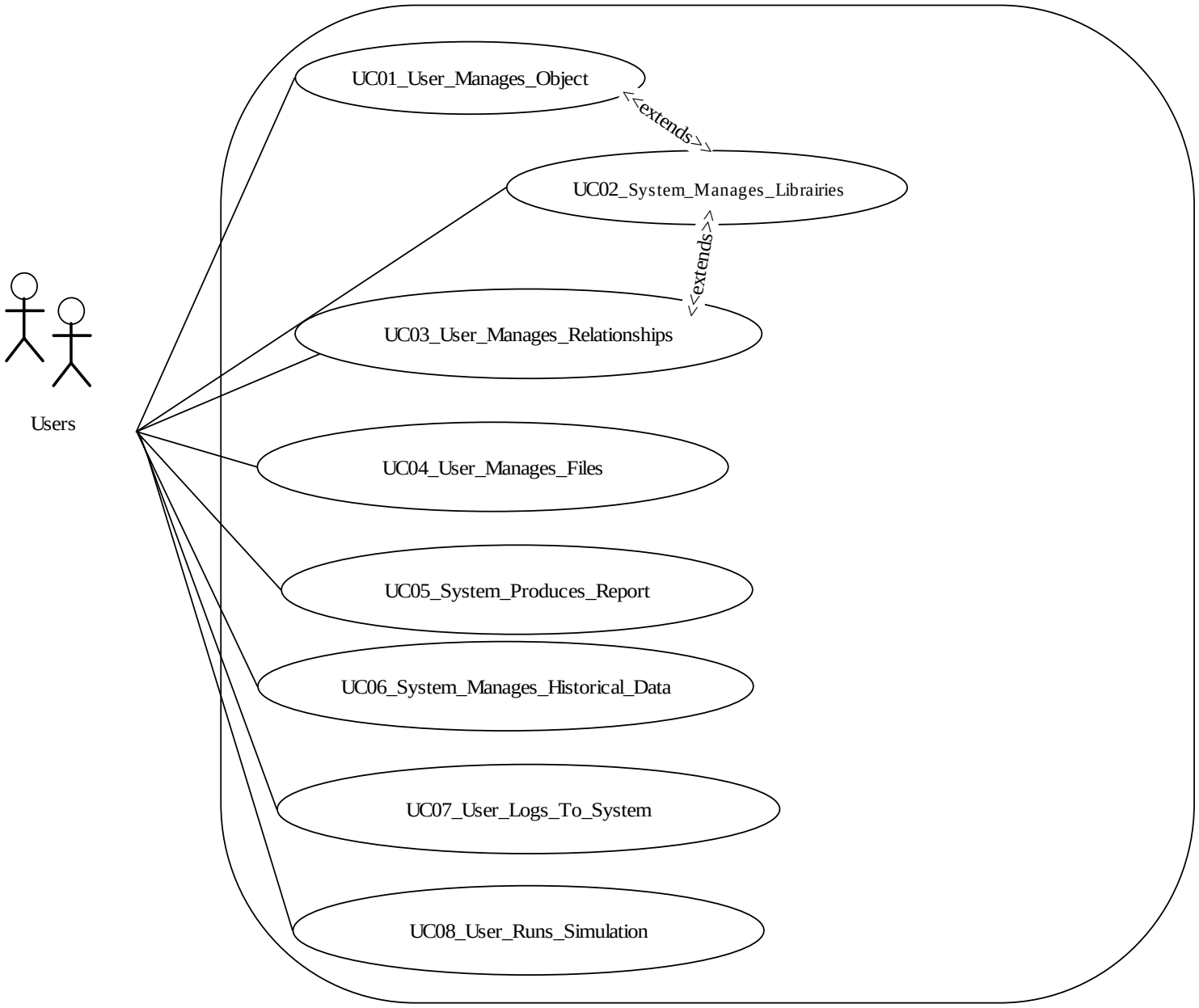
1. Illegal computation when computing results.

Use Cases Utilized:

None.

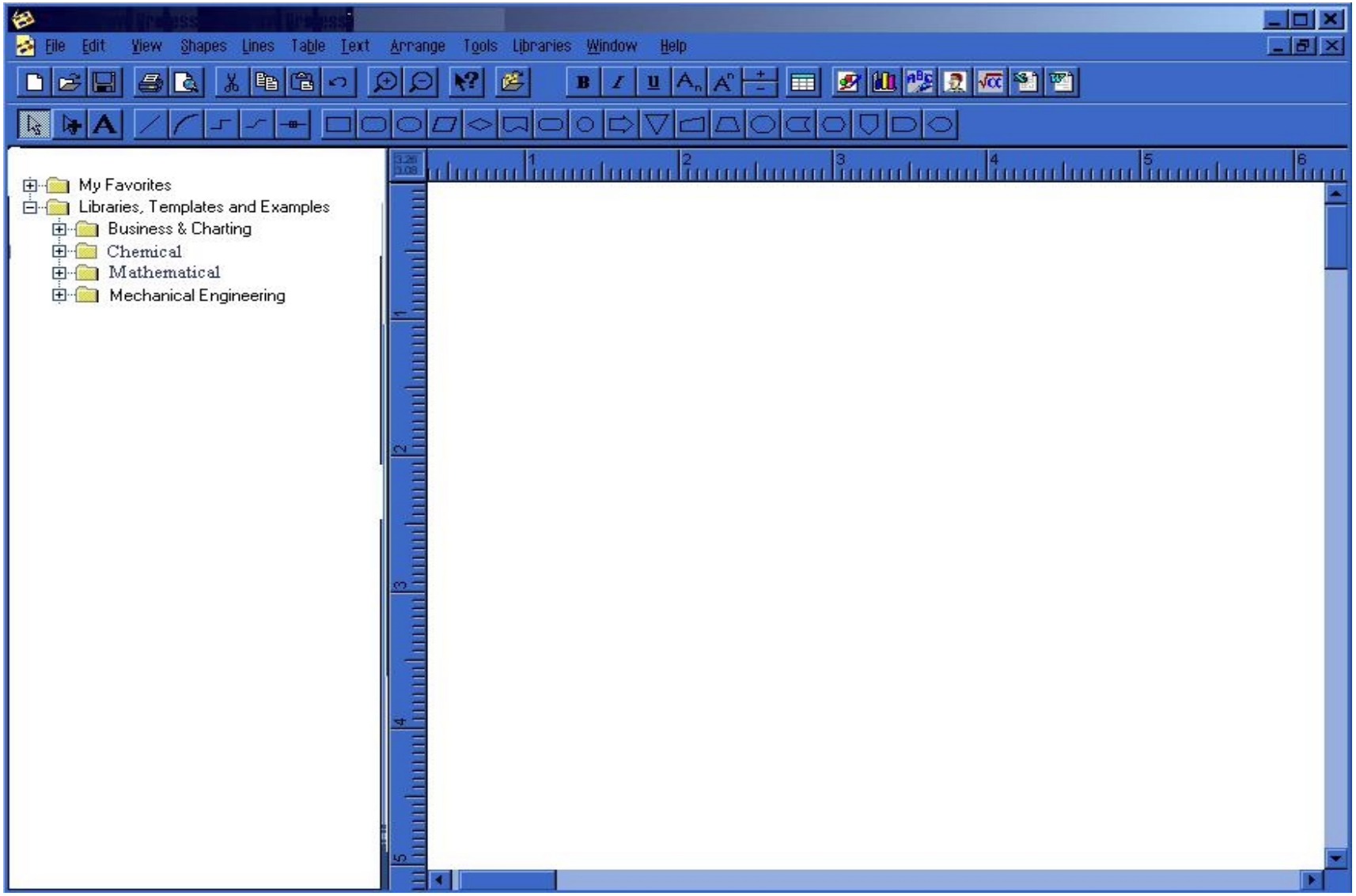
Timing Constraints:

None.


UC01_User_Manages_Object
<<extends>>
UC02_System_Manages_Librairies
<<extends>>
UC03_User_Manages_Relationships
Users
UC04_User_Manages_Files
UC05_System_Produces_Report
UC06_System_Manages_Historical_Data
UC07_User_Logs_To_System
UC08_User_Runs_Simulation

3 – Develop draft GUI

The purpose of this section is to develop a draft of the GUI for the system, as envisioned during the development of the Scenarios.



4 – Establish project categories

The purpose of this activity is to develop an initial Category List. Categories provide the kernel for all the future software development effort.
A Category is defined as a collection of logically related classes. Logically related classes are classes related through aggregation, composition, inheritance or any meaningful association.

Software Candidate Category:

- User_Interface_CAT
- Project_Manager_CAT
- File_Manager_CAT
- Report_Manager_CAT
- User_Manager_CAT
- Simulation_Manager_CAT

5 – Allocate use cases to categories

The purpose of allocating use cases to categories is to allocate the responsibility for use case development to a Category (and thus to a software Category Lead). This allocation makes it perfectly clear who is responsible for the development of a specific Use Case.

Entry	System Specification Text	Type	Build	Use Case Name	Scenario	Category
1	The system shall have a graphical user interface.	SWC	1	N/A	N/A	N/A
2	The system shall be used in a Windows environment.	SWC	1	N/A	N/A	N/A
3	The system shall support multiple users.	SWC	1	N/A	N/A	N/A
4	The user shall be able to create new objects.	SW	1	UC01_User_Manages_Object	1	Project_Manager_CAT
5	The system shall allow the user to add existing objects.	SW	2	UC01_User_Manages_Object	6	Project_Manager_CAT
6	The system shall be able to allow the user to select different type of objects.	SW	1	UC01_User_Manages_Object	3,4,5	Project_Manager_CAT
7	The system shall allow the user to name the selected object.	SW	1	UC01_User_Manages_Object	1	Project_Manager_CAT
8	The user shall be able to add attributes to an object.	SW	2	UC01_User_Manages_Object	2	Project_Manager_CAT
9	The user shall be able to modify any attributes of any object.	SW	2	UC01_User_Manages_Object	4	Project_Manager_CAT
10	The user shall be able to delete any attributes of any object.	SW	2	UC01_User_Manages_Object	3	Project_Manager_CAT
11	The user shall be able to add object inside other objects.	SW	1	UC01_User_Manages_Object	6	Project_Manager_CAT
12	The user shall be able to delete any object.	SW	1	UC01_User_Manages_Object	7	Project_Manager_CAT
13	The user shall be able to breakdown an object into several different objects that shall be considered as attributes of the	SW	2	UC01_User_Manages_Object	5	Project_Manager_CAT

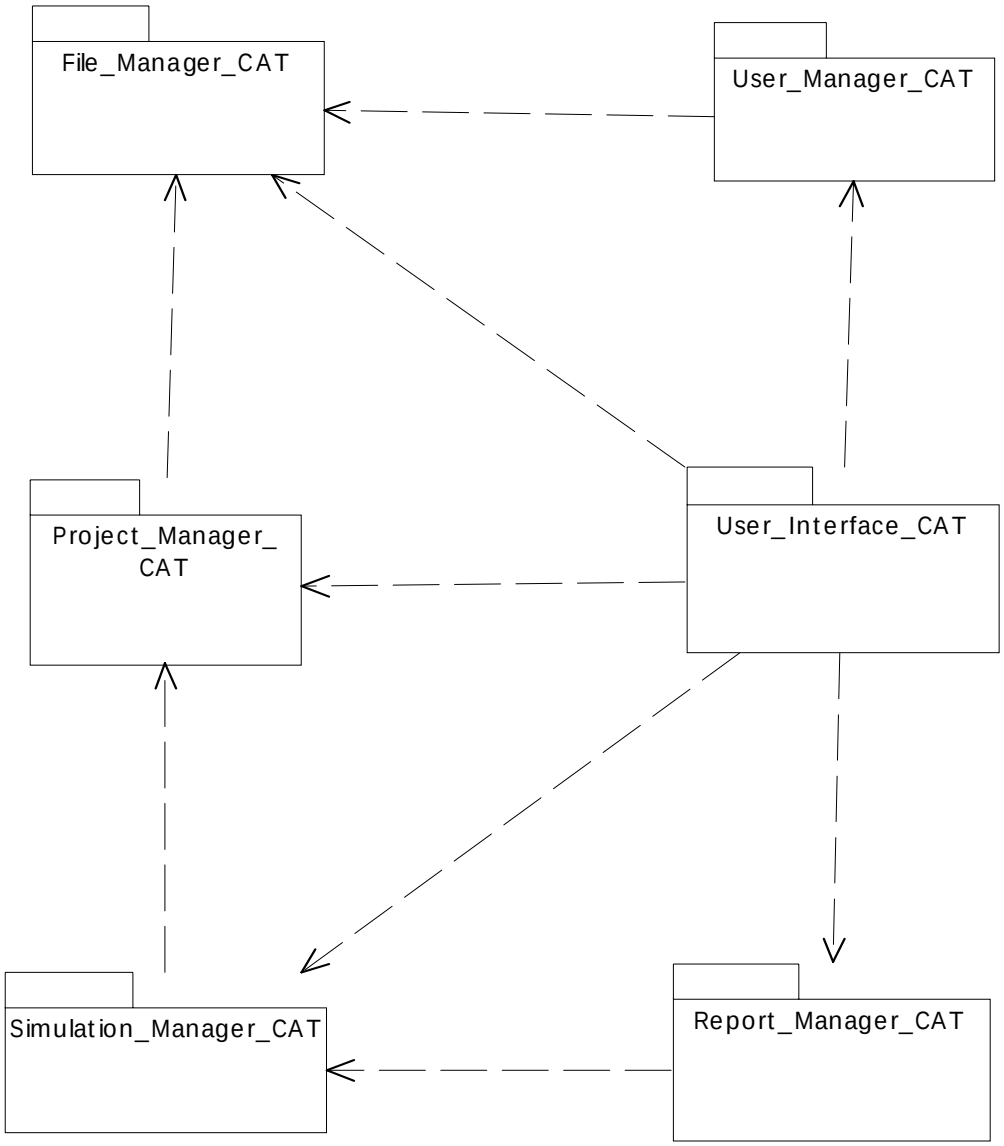
	main object.					
14	The system shall provide different application domain. (Physics, mathematics, chemical at least).	SW	4	UC02_User_Manages_Libraries	1	Project_Manager_CAT
15	The user shall be able to add new application domain (Library).	SW	4	UC02_User_Manages_Libraries	2	Project_Manager_CAT
16	The library shall be initialized with default values.	SW	4	UC02_User_Manages_Libraries	1	Project_Manager_CAT
17	The user shall be able to add new relationship to any library.	SW	4	UC02_User_Manages_Libraries	3	Project_Manager_CAT
18	The user shall be able to see the different libraries of the system and their contents.	SW	4	UC02_User_Manages_Libraries	1	Project_Manager_CAT
19	The user shall define the relationship.	SW	3	UC03_User_Manages_Relationships	2	Project_Manager_CAT
20	The user shall be able to define the relationship direction.	SW	3	UC03_User_Manages_Relationships	2	Project_Manager_CAT
21	The user shall be able to modify any relationship.	SW	3	UC03_User_Manages_Relationships	3	Project_Manager_CAT
22	The user shall be able to link object manually.	SW	3	UC03_User_Manages_Relationships	1	Project_Manager_CAT
23	The user shall be able to add a relationship to any graphically linked objects.	SW	3	UC03_User_Manages_Relationships	2	Project_Manager_CAT
24	The user shall be able to draw arrows between object.	SW	3	UC03_User_Manages_Relationships	1	Project_Manager_CAT
25	The user shall be able to provide the relationship formula in both direction.	SW	3	UC03_User_Manages_Relationships	2	Project_Manager_CAT
26	The system shall be able to find the relationship formulas between directly related objects.	SW	3	UC03_User_Manages_Relationships	4	Project_Manager_CAT
27	The mathematical relationships shall be given according to the names of the objects.	SW	3	UC03_User_Manages_Relationships	4	Project_Manager_CAT
28	The system shall be able to identify relationship between non-directly related objects.	SW	4	UC03_User_Manages_Relationships	4	Project_Manager_CAT
29	The user shall be able to use relationship	SW	4	UC03_User_Manages_Relatio	2	Project_Manager_CAT

	from libraries.			nships		
30	The object shall be transferable from one model to another.	SW	4	UC04_System_Manages_Files	3	File_Manager_CAT
31	The user shall be able to open an existing file.	SW	1	UC04_System_Manages_Files	1	File_Manager_CAT
32	The user shall be able to save the actual.	SW	1	UC04_System_Manages_Files	2	File_Manager_CAT
33	The user shall only accede file for which he has write and/or read authorization.	SW		UC04_System_Manages_Files	1,2	File_Manager_CAT
34	The system shall produce reports in textual and graphics format.	SW	5	UC05_System_Produces_Report	1,2	Report_Manager_CAT
35	The report shall be alphabetical and/or relational.	SW	5	UC05_System_Produces_Report	1,2	Report_Manager_CAT
36	The system shall run the simulation before producing the report.	SW	5	UC05_System_Produces_Report	1,2	Report_Manager_CAT
37	The system shall have an undo function.	SW	5	UC06_System_Manages_Historical_Data	1	File_Manager_CAT
38	The system shall keep a database of all the modifications introduced with the related dates.	SW	5	UC06_System_Manages_Historical_Data	2	File_Manager_CAT
39	The system shall be able to show the sequence of all the modifications.	SW	5	UC06_System_Manages_Historical_Data	2	File_Manager_CAT
40	An authorized user shall be able to login in the system.	SW	1	UC07_User_Logs_To_System	1	User_Manager_CAT
41	The system shall check if the username and password are in the authorized user list.	SW	1	UC07_User_Logs_To_System	1	User_Manager_CAT
42	The system shall be able to do error checking for variable consistency (food or Food).The system shall warn the user in case of errors.	SW	4	UC08_User_Runs_Simulation	1	Simulation_Manager_CAT
43	The system shall warn the user in case of errors.	SW	4	UC08_User_Runs_Simulation	1	Simulation_Manager_CAT
44	The user shall be able to simulate and verify the consistency of the relationship.	SW	4	UC08_User_Runs_Simulation	1	Simulation_Manager_CAT

6 – Develop System Category Diagram (SCD)

The purpose of developing a System Category Diagram (SCD) is to depict the high-level Association that exist between the Categories and thus validate the Categories.

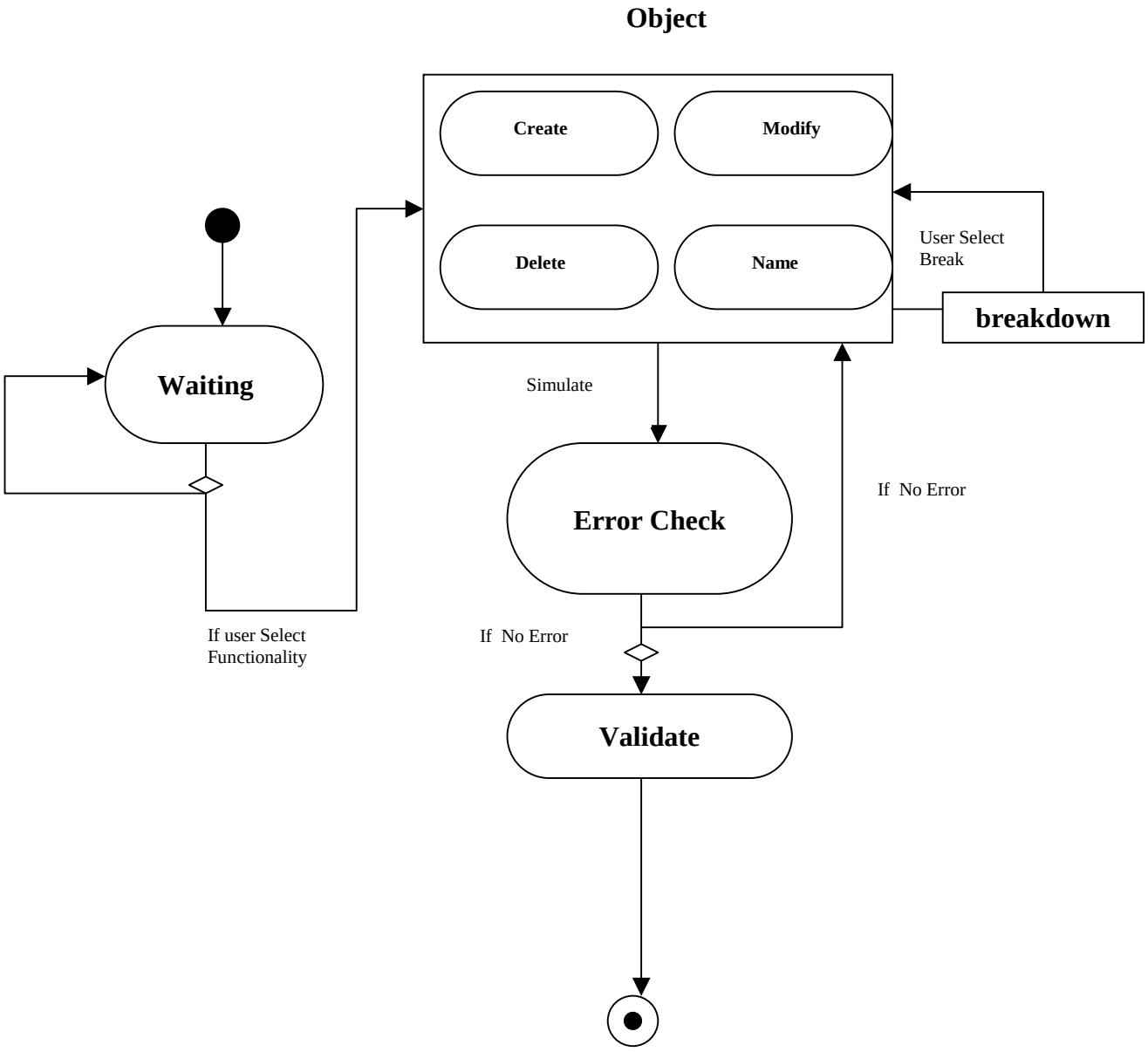
System Category Diagram



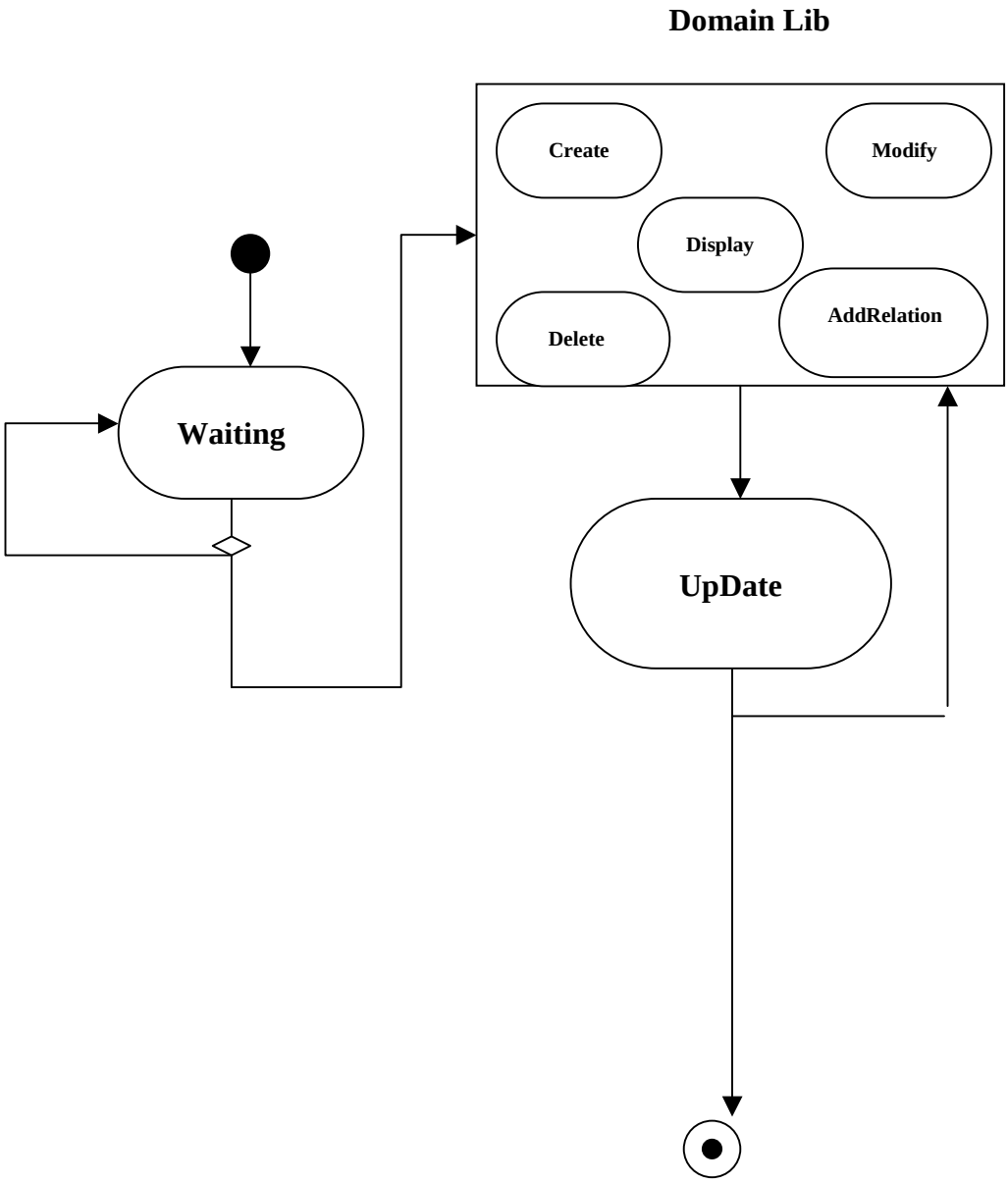
7 – Activity Diagram

An Activity Diagram is an alternative to a State-Chart. It illustrates the state machine that is algorithmic rather than event driven. The way a state chart models an event driven driven system, An Activity Diagram models a system where the actions are sequential. In such a system, once the actions are completed, the state machine automatically transition to the next state, much the way you would see it in a flowchart. The activity diagram can be used to describe the behavior of a class or use case. Although it primarily describes algorithmic behavior, it can also include some event driven behavior.

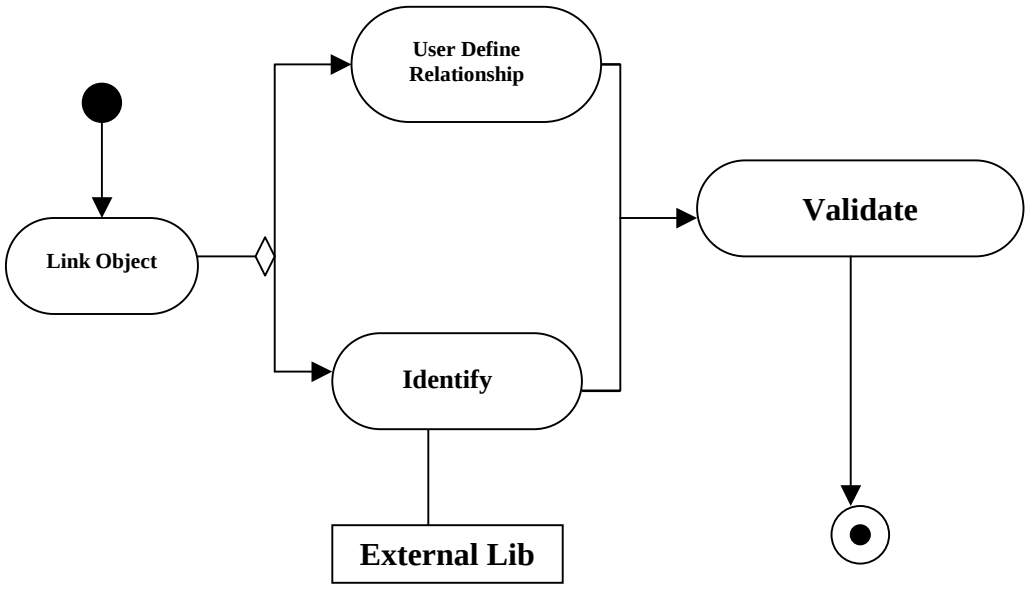
UC01_User_Manage_Object_Activity_Diagram



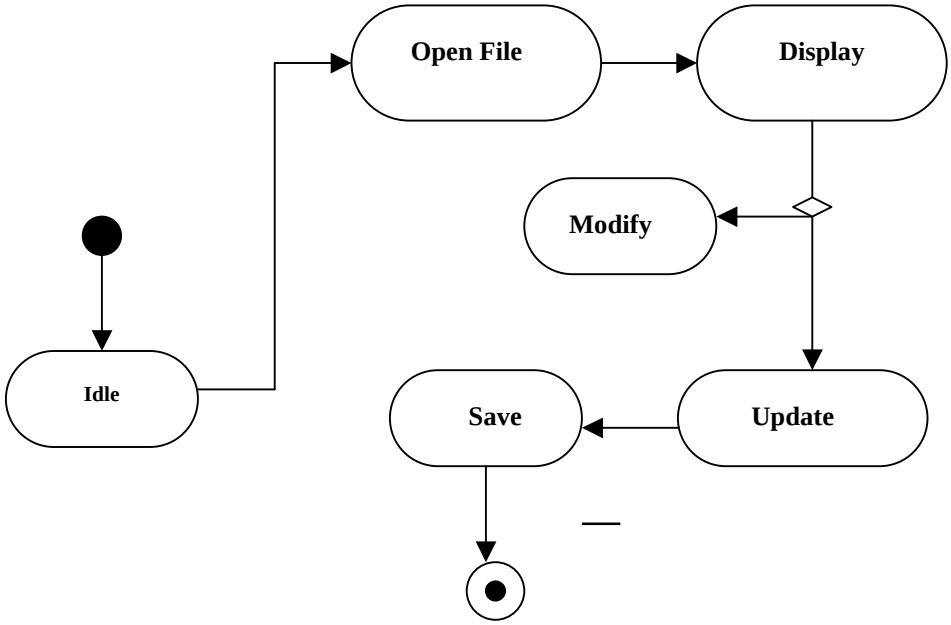
UC02_User_Manage_Library Activity_Diagram



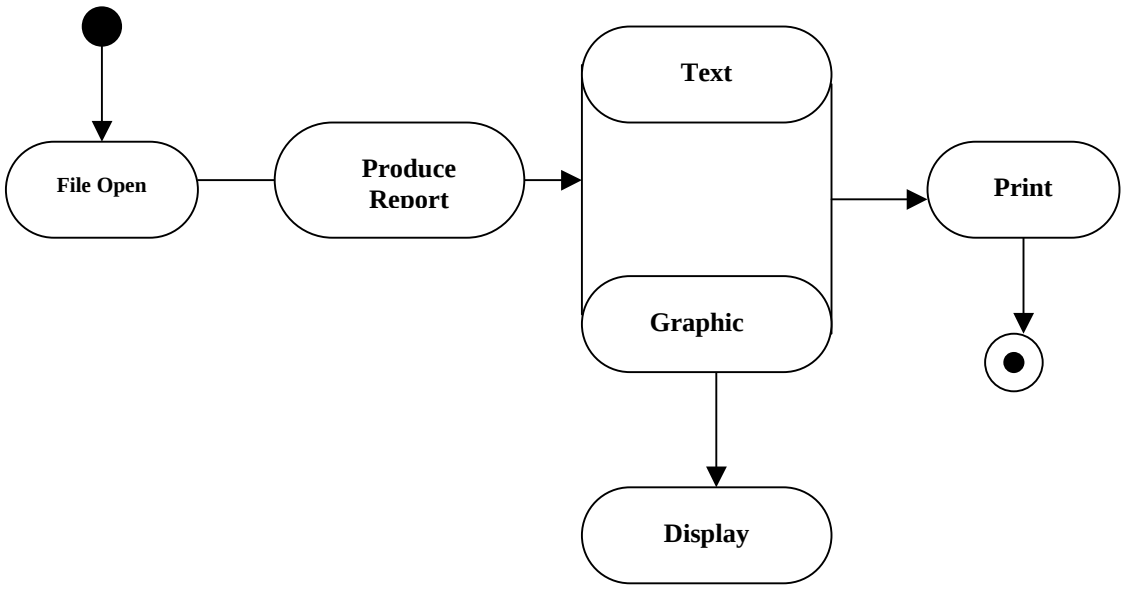
UC03_User_Manage_Relationship_Activity_Diagram



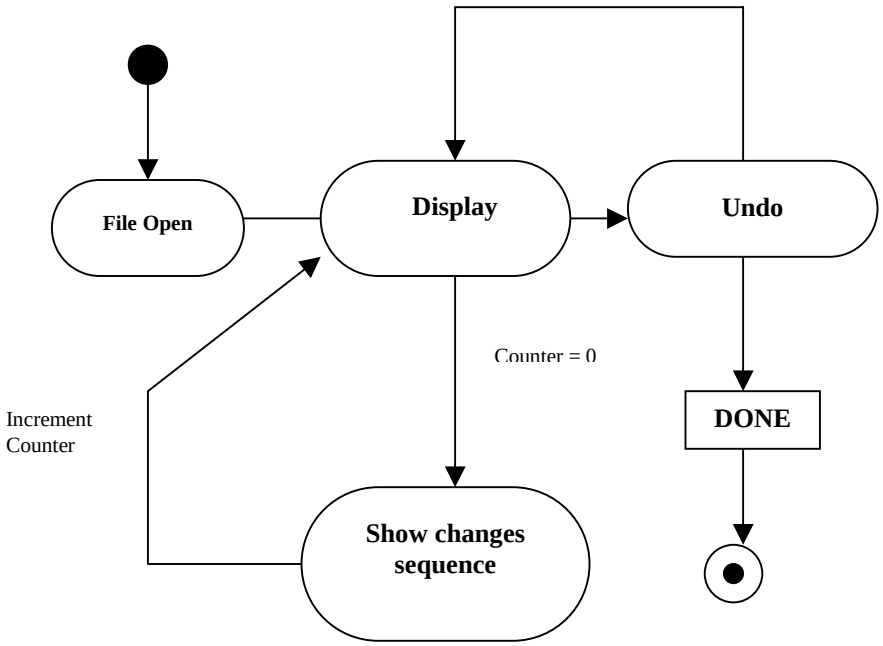
UC04_System_Manage_File_Activity_Diagram



UC05_System_Produces_Report_Activity_Diagram



UC06_System_Manage_Historical_Data_Diagram



PHASE 3: System Object Oriented Analysis, Dynamic View

1 – Allocate Category to category manager/Leads

The purpose of this activity is to assign responsibility for the development of one category to one individual. During this phase a Category Development Responsibilities will be established in which there will be a Category Lead and a Category Manager. A Category Manager is an individual who is assigned both administrative and technical responsibility for the development of one or more categories. In the other hand a Category Lead is an individual who is responsible for either the hardware or software portion of one or more categories.

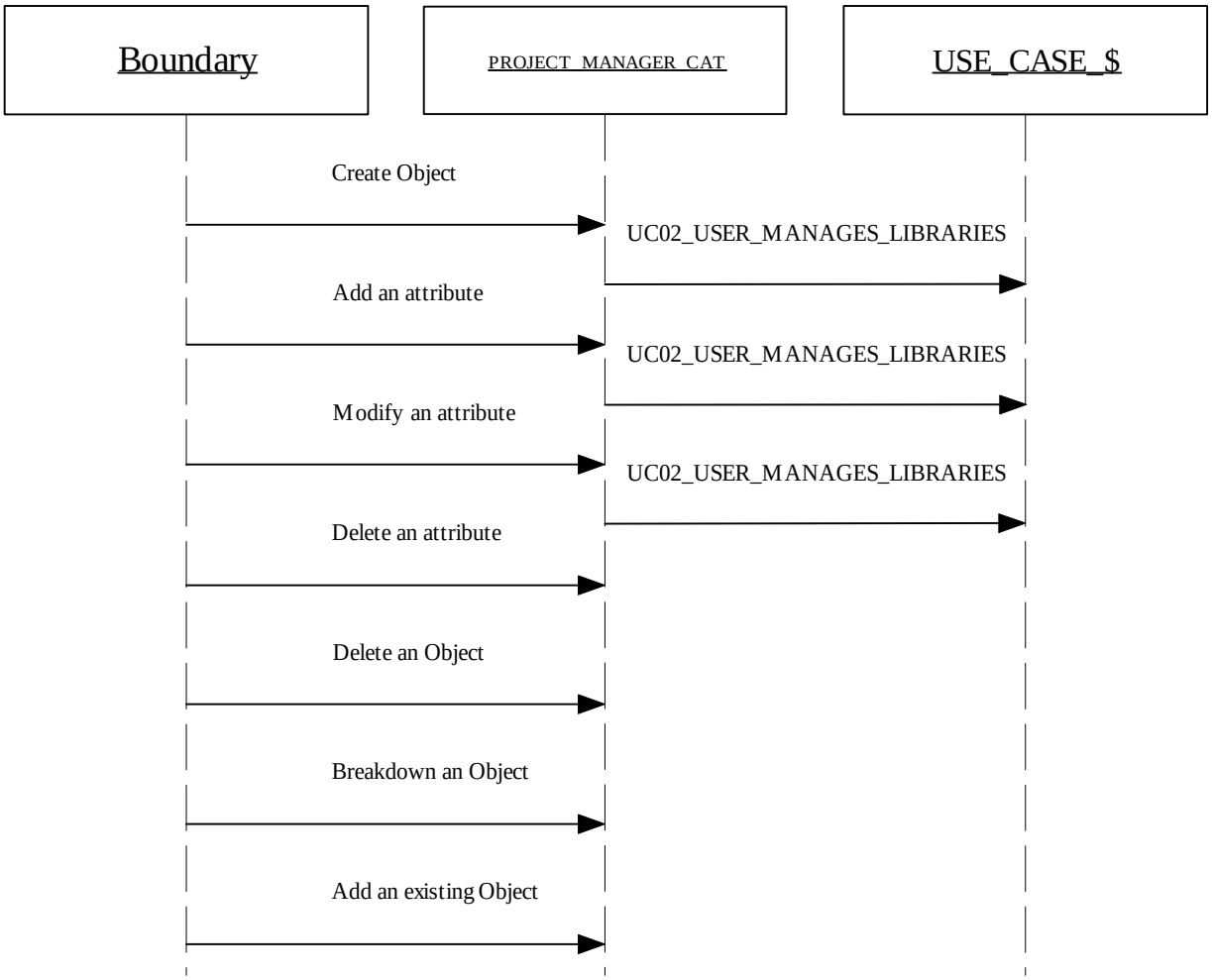
Categories	Category Lead
User_Interface_CAT	Chokri Oueslati
User_Manager_CAT	Chokri Oueslati
Project_Manager_CAT	Thomas Pombourg
File_Manager_CAT	Thomas Pombourg
Report_Manager_CAT	Francoise Boudigou
Simulation_Manager_CAT	Francoise Boudigou

2 – Develop Category Interaction Diagram (CID)

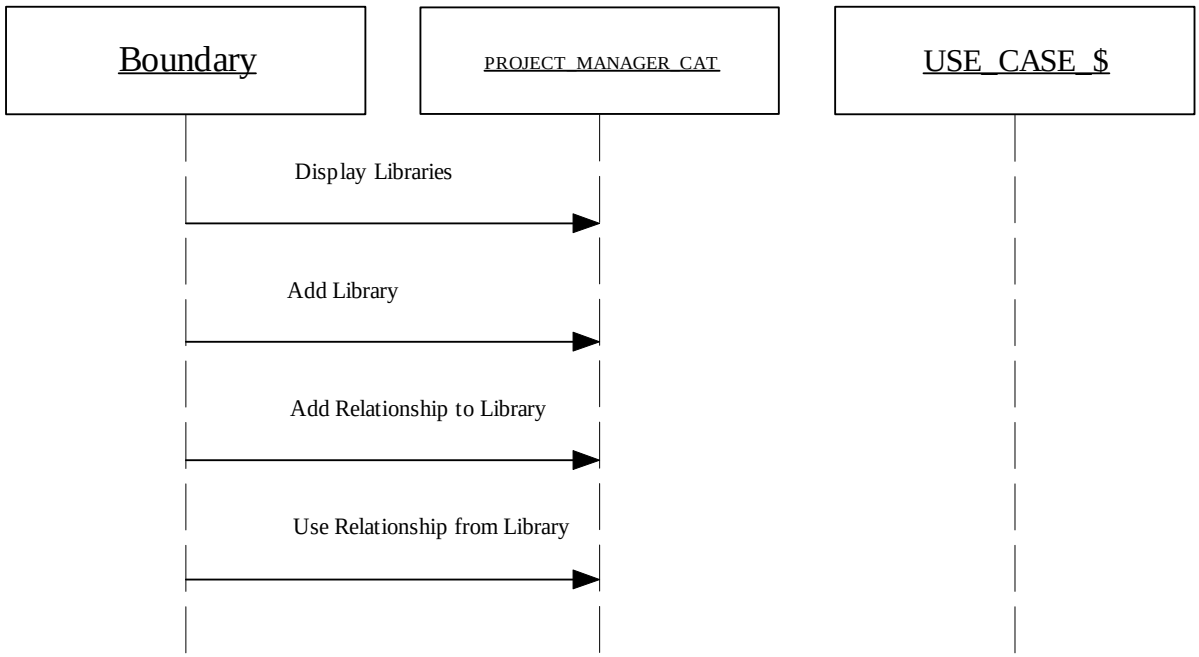
The purpose of this activity is to identify how the Categories collaborate to implement a use case.

Category Interaction Diagram is a graphical representation of the interaction between categories required to satisfy, or implement, a Use Case using the notation of a case tool. In the UML 1.0 terminology, an interaction diagram is any diagram that depicts the interaction between classes at the instance level, and can be either a Sequence Diagram or Collaboration Diagram.

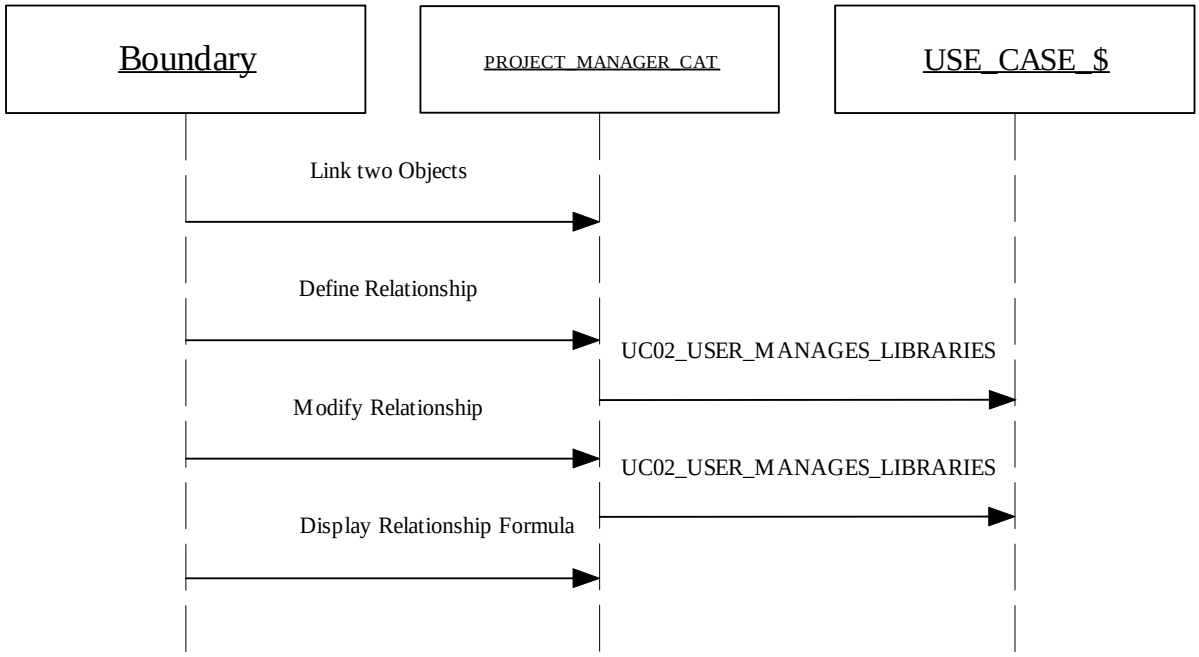
UC01_USER_MANAGES_OBJECT_CID



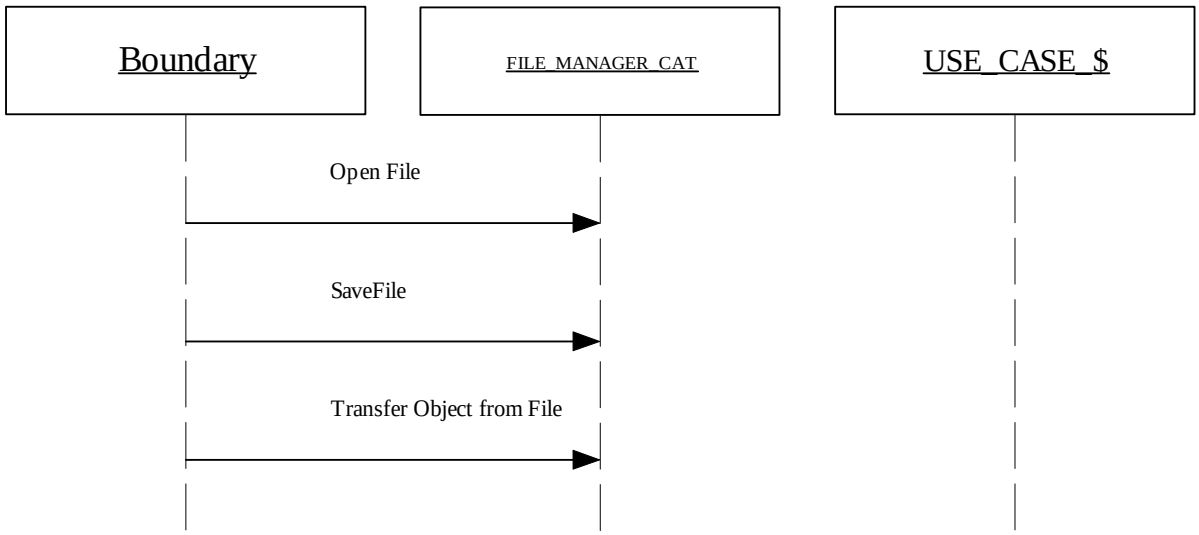
UC02_USER_MANAGES_LIBRARIES_CID



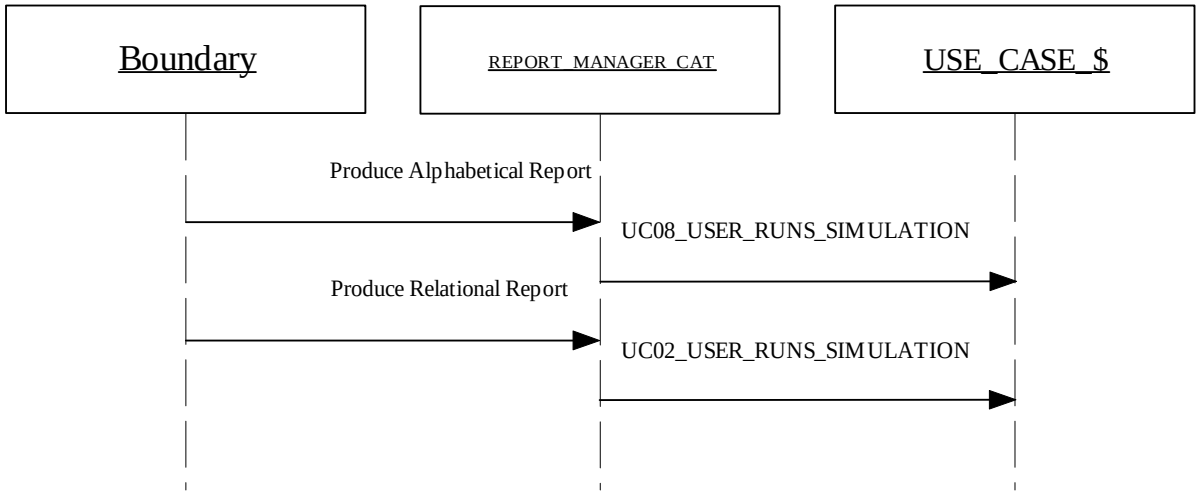
UC03_USER_MANAGES_RELATIONSHIPS_CID



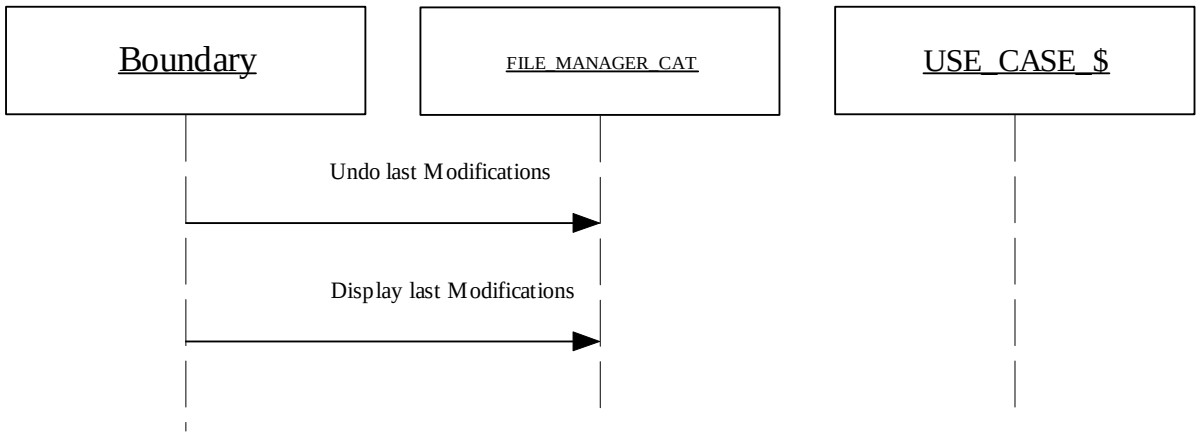
UC04_SYSTEM_MANAGES_FILES_CID



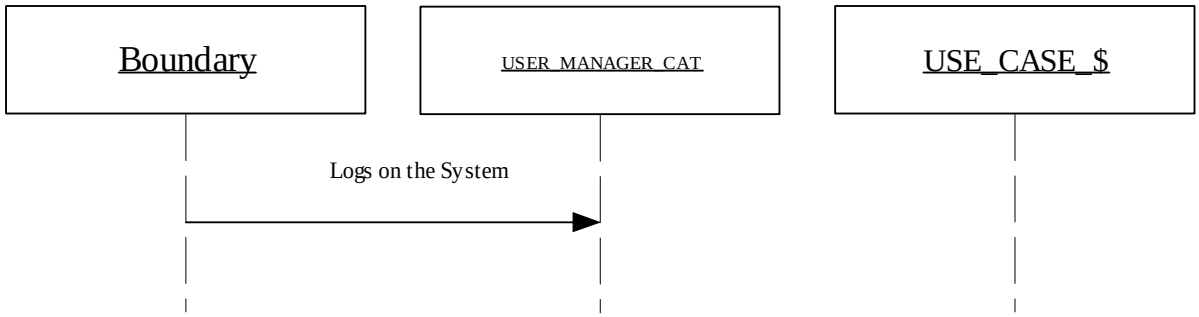
UC05_SYSTEM_PRODUCES_REPORT_CID



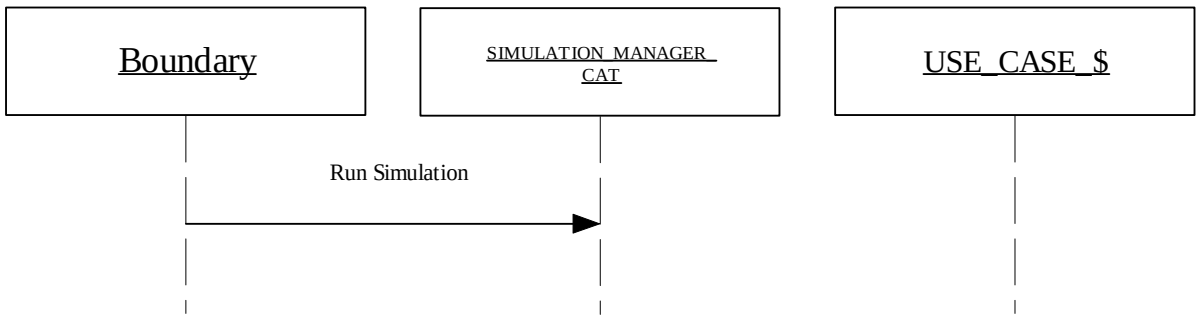
UC06_SYSTEM_MANAGES_HISTORICAL_DATA_CID



UC07_USER_LOGS_TO_SYSTEM_CID



UC08_USER_RUNS_SIMULATION_CID



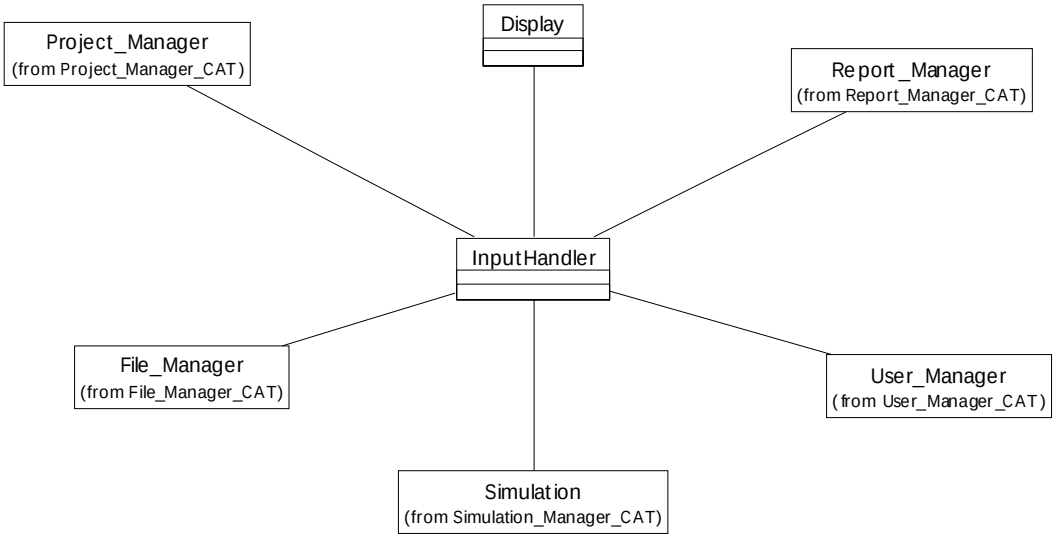
PHASE 4&5: Software Object Oriented Analysis, Static View

1 – Initiate Category Class Diagram

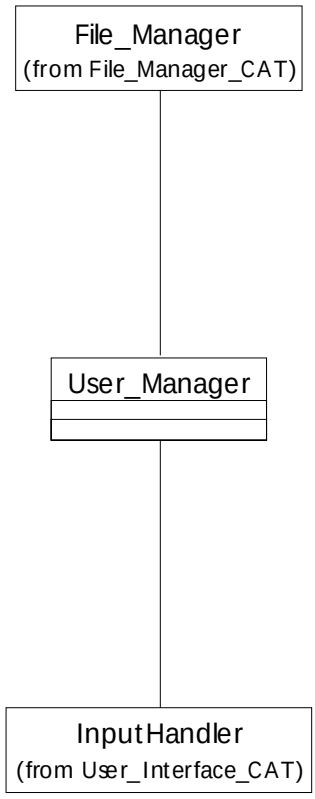
The purpose of this phase is simply to produce a "canvas" for a Category Class Diagram (CCD) for any Category whose CCD was not initiated during the previous phase.

Category Class Diagram (CCD) for one category is a class diagram that depicts all classes owned by the category, all classes required by the classes in the category that are owned by other categories, (imported classes), and all the association between these classes. There is one CCD per category.

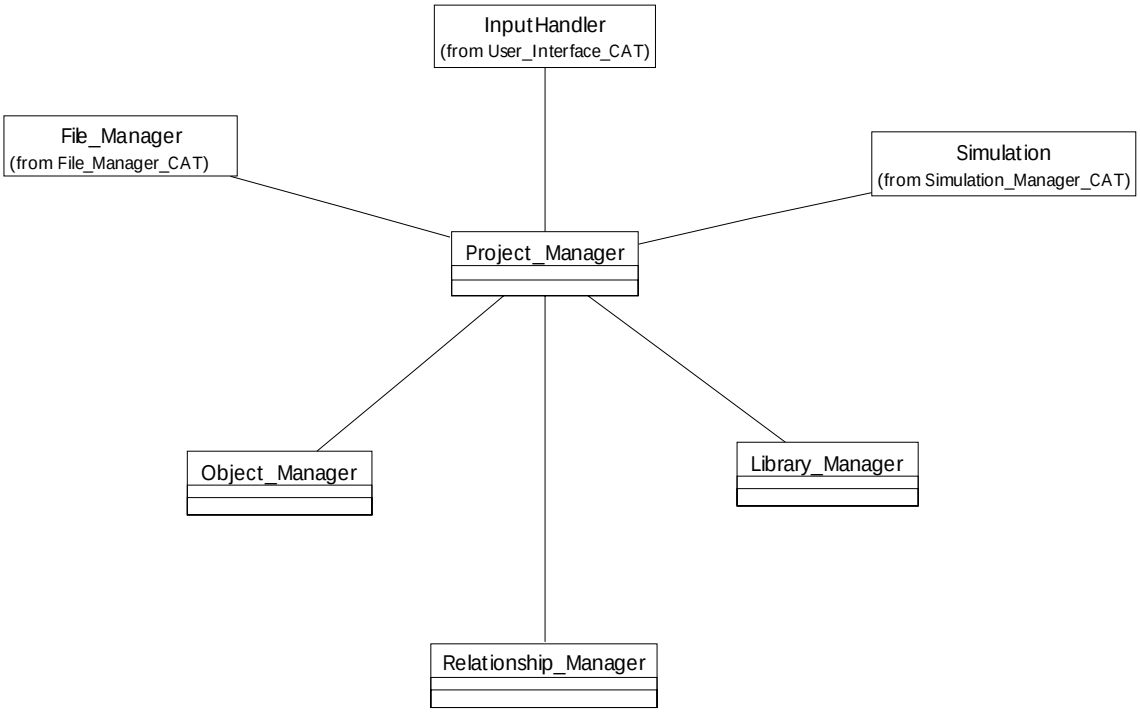
User_Interface_CAT_CCD



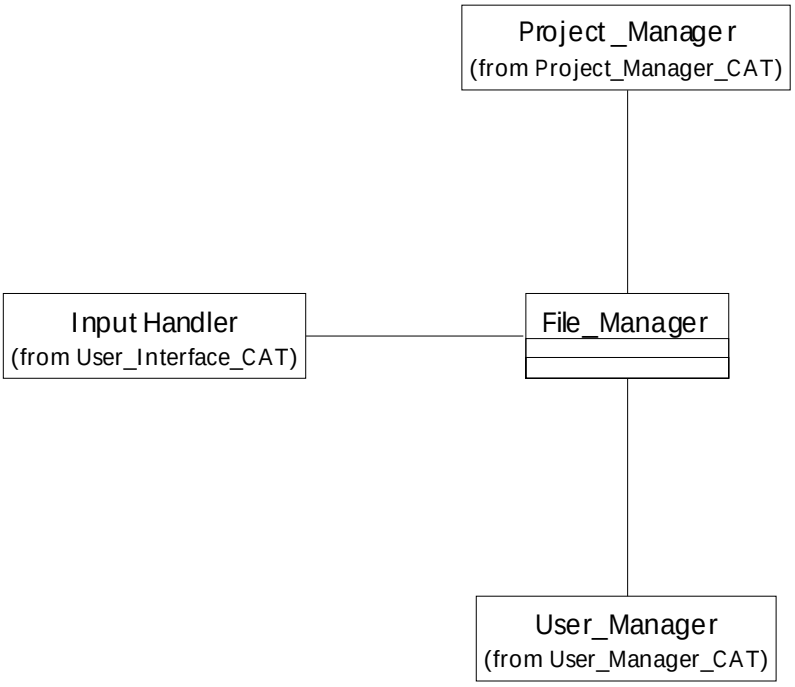
User_Manager_CAT_CCD



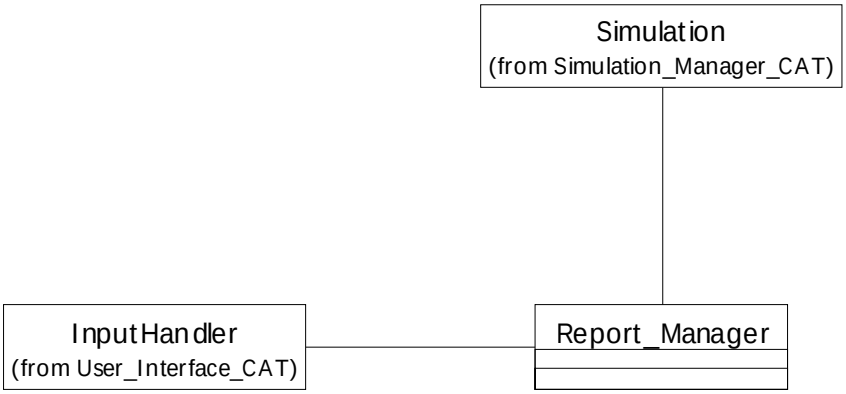
Project_Manager_CAT_CCD



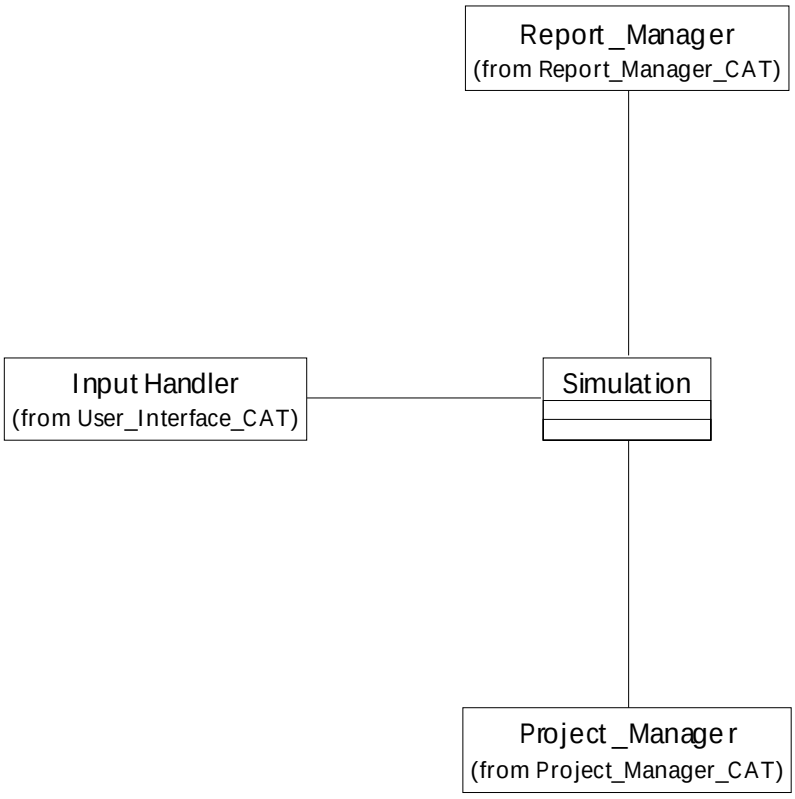
File_Manager_CAT_CCD



Report_Manager_CAT_CCD



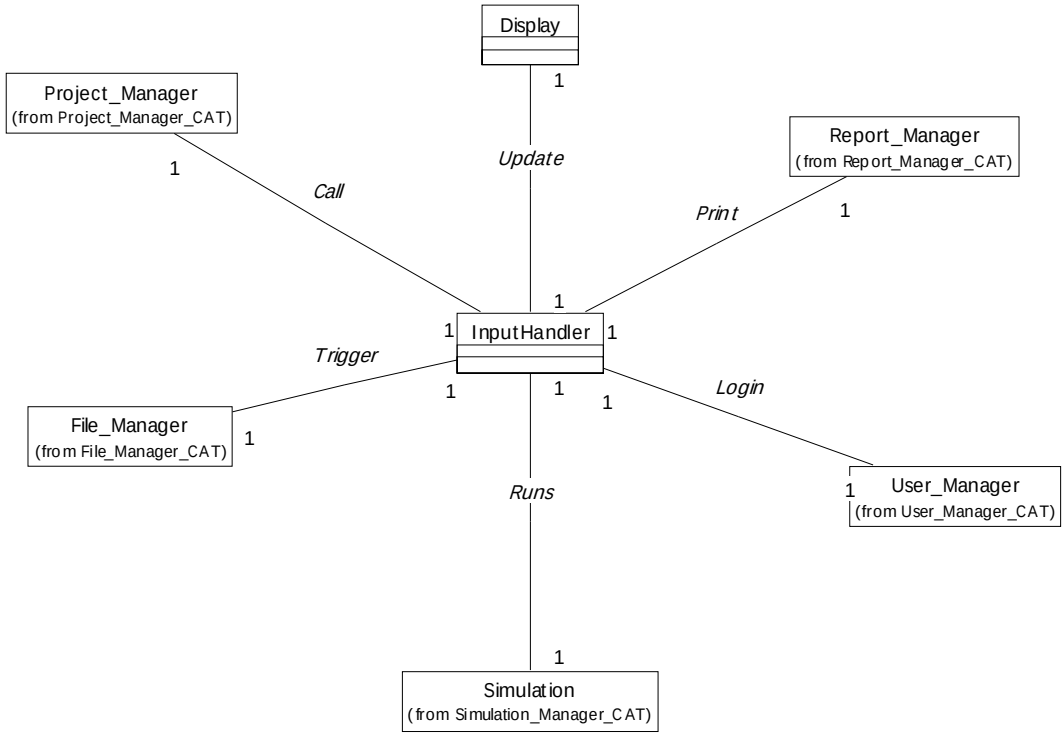
Simulation_Manager_CAT_CCD



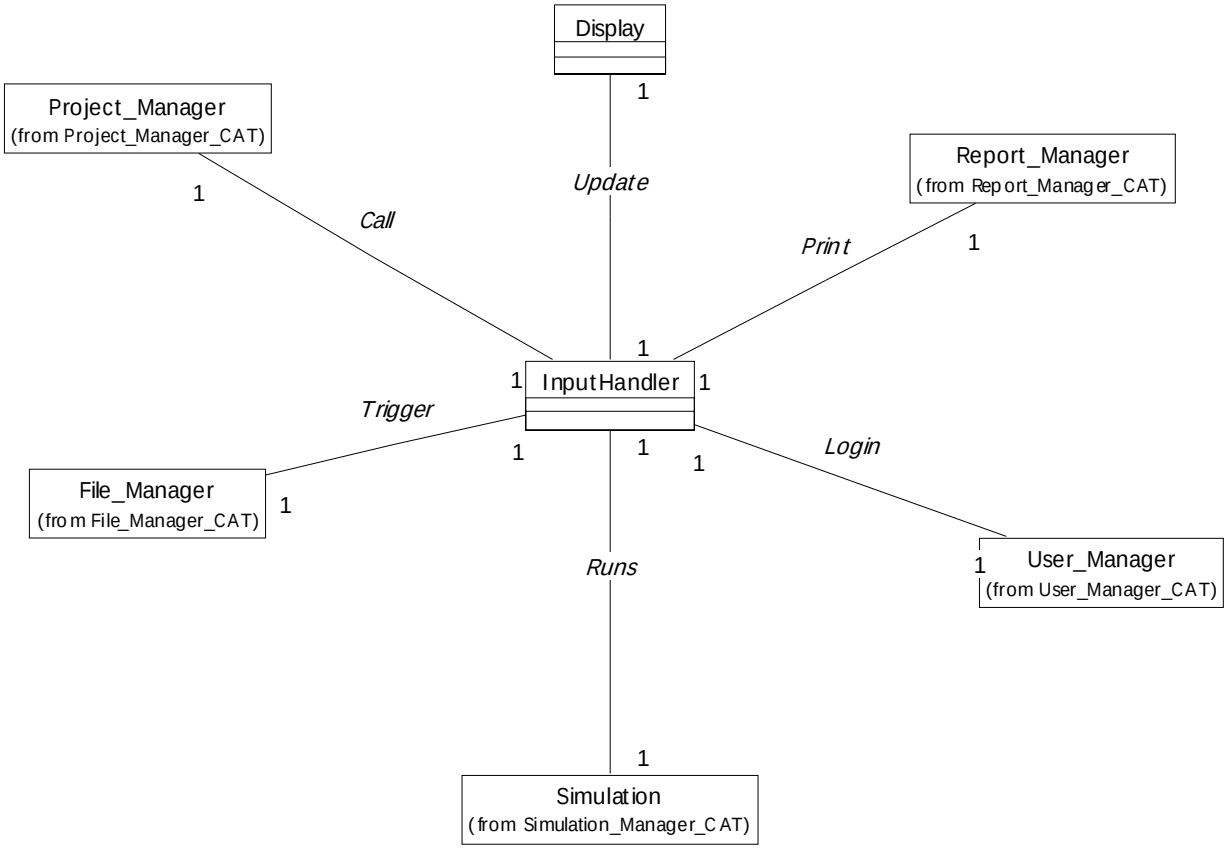
2 – Refine Inheritance, Aggregation, hierarchies

The purpose of this activity is to focus on developing and representing the inheritance hierarchy and or aggregation hierarchy for each category in a CCD. Additionally, this activity continues to justify and validate the category list.
Aggregation hierarchy is a set of classes related through an aggregation relationship. An aggregation relationship is an association between the classes that focus on one class being made up of another class. It is the relationship that exist between whole class and its part class.
Inheritance is defined as being an association relationship that focus on similarities dissimilarities between classes. It is a relationship that exist between super class and sub class.

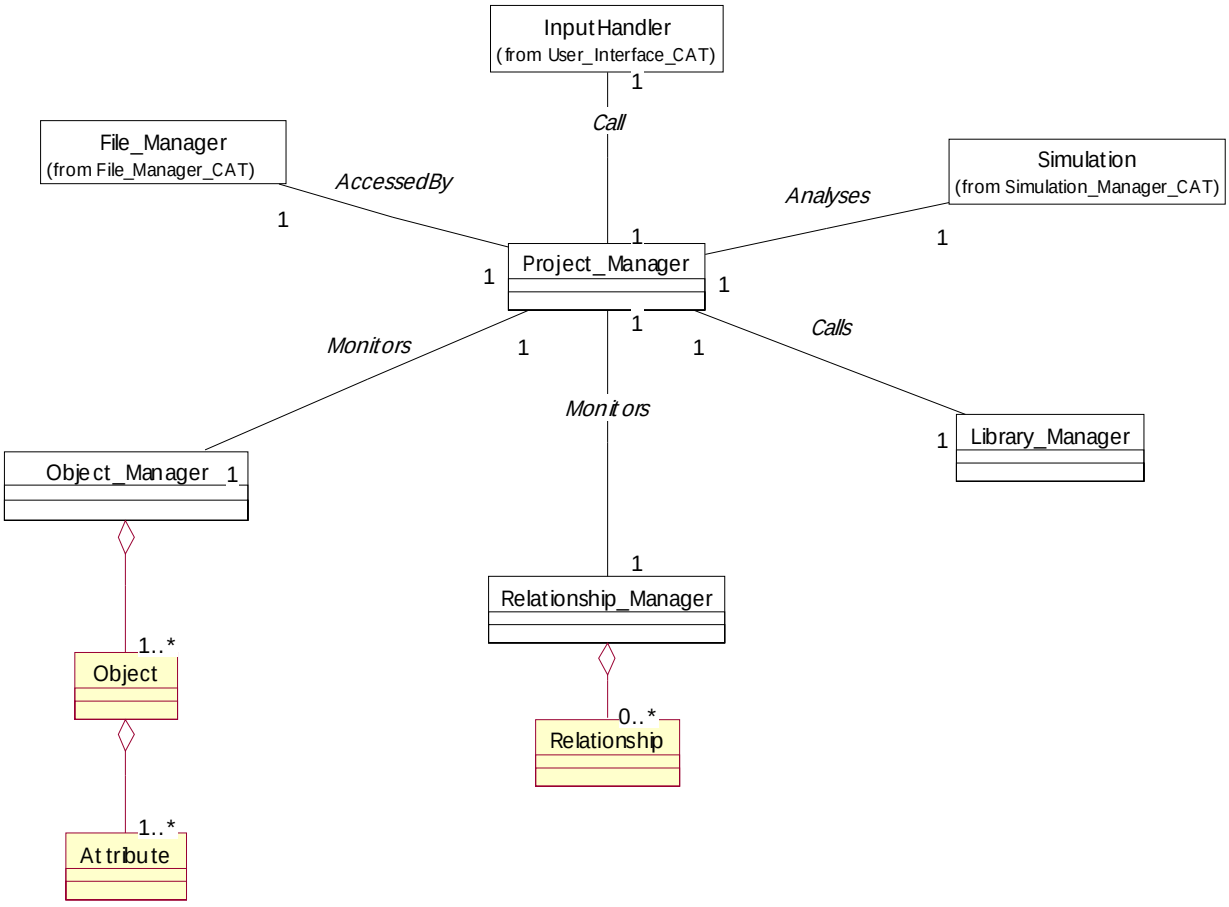
User_Interface_CAT_CCD



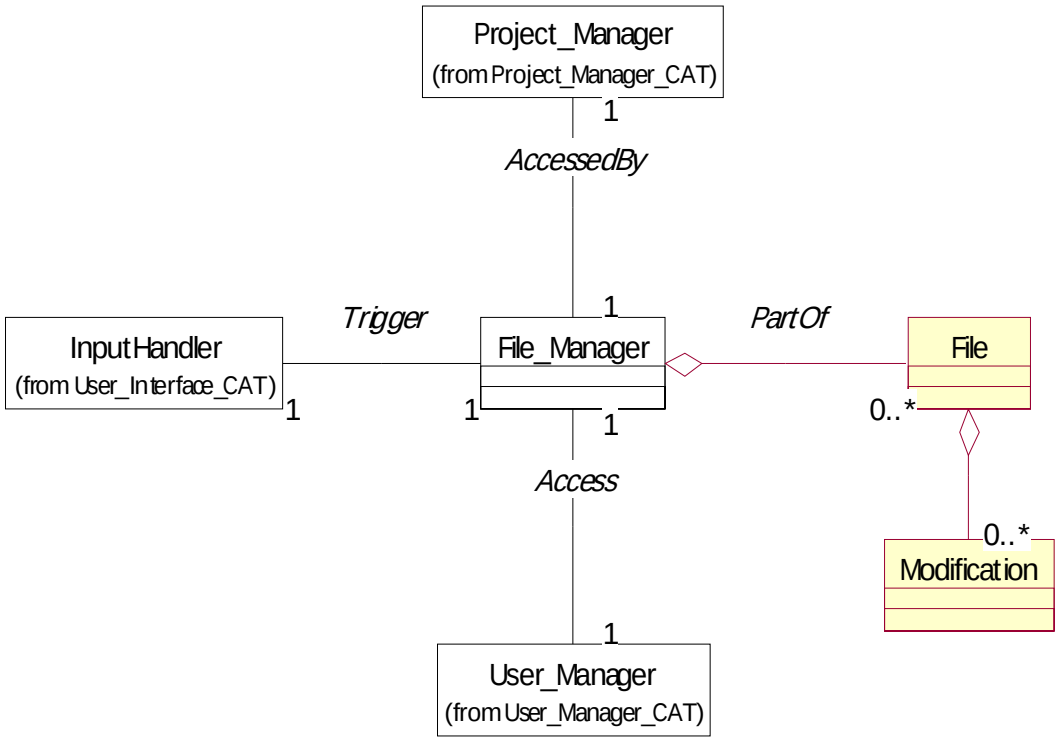
User_Manager_CCD_CAT



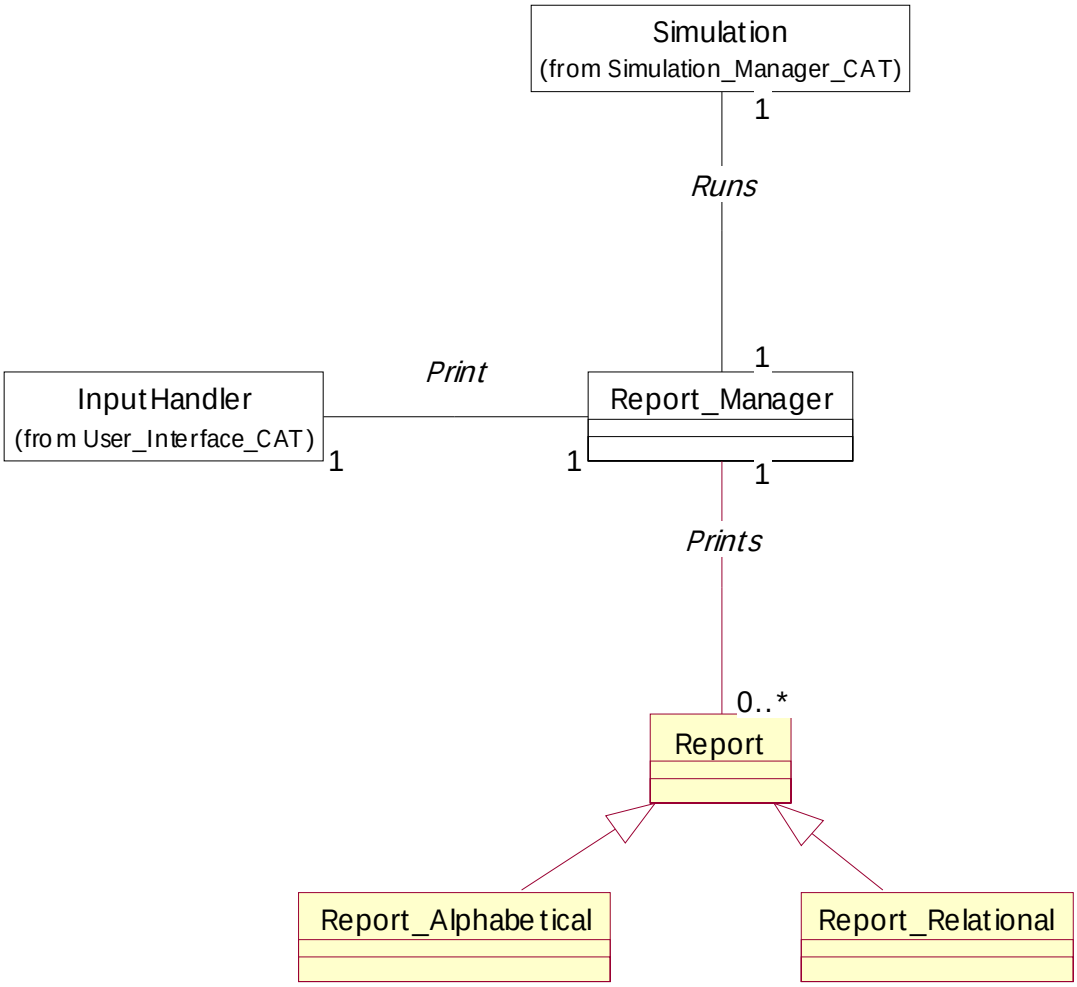
Project_Manager_CAT_CCD



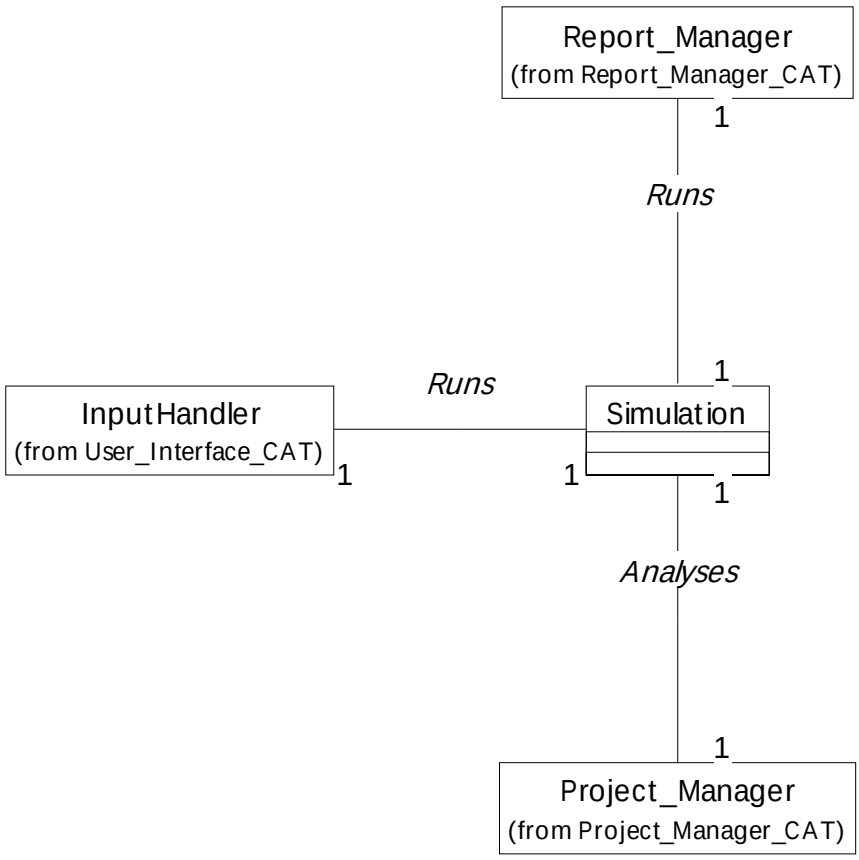
File_Manager_CAT_CCD



Report_Manager_CCD_CAT



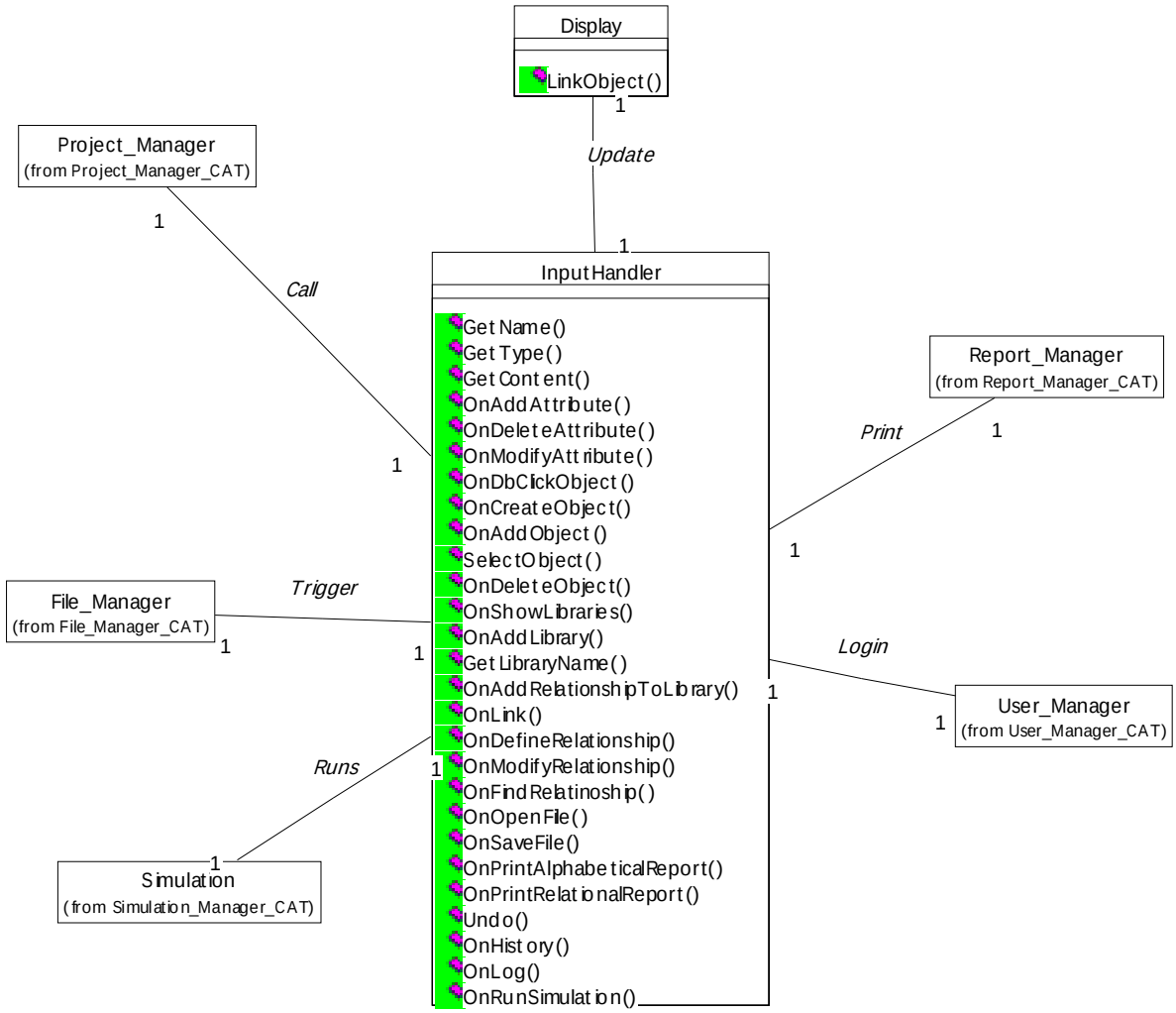
Simulation_Manager_CAT_CCD



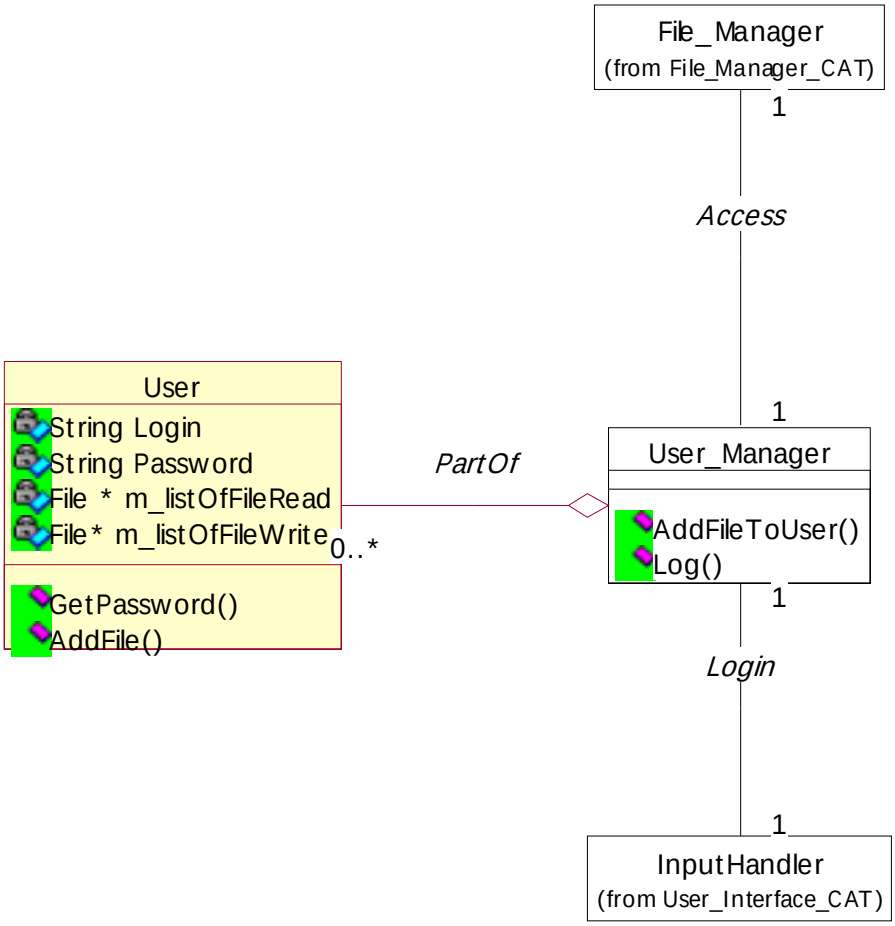
3 – Update Category Class Diagram

The purpose of this activity is to argument the existing CCD with the classes discovered in the previous activity.

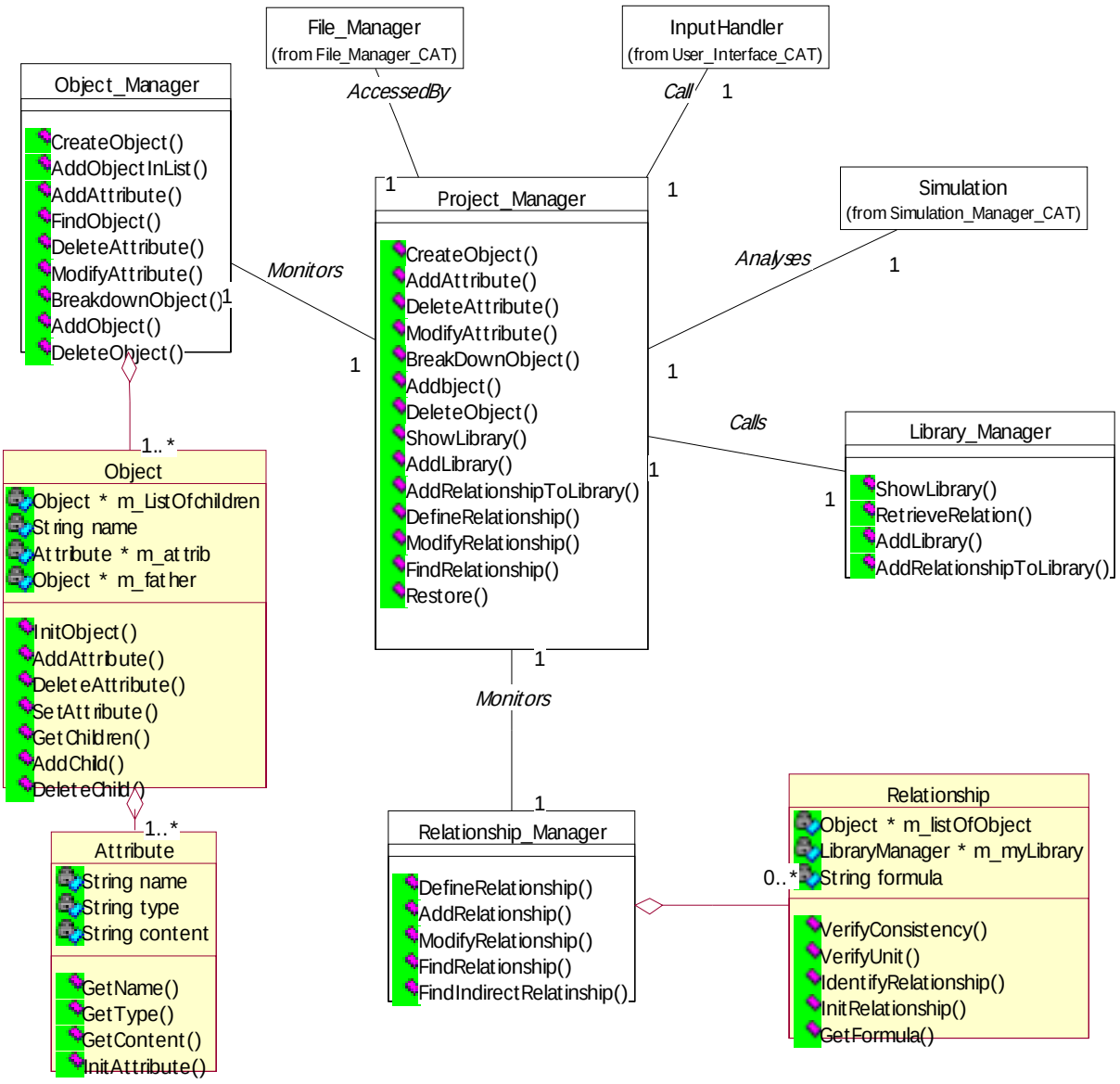
User_Interface_CAT_CCD



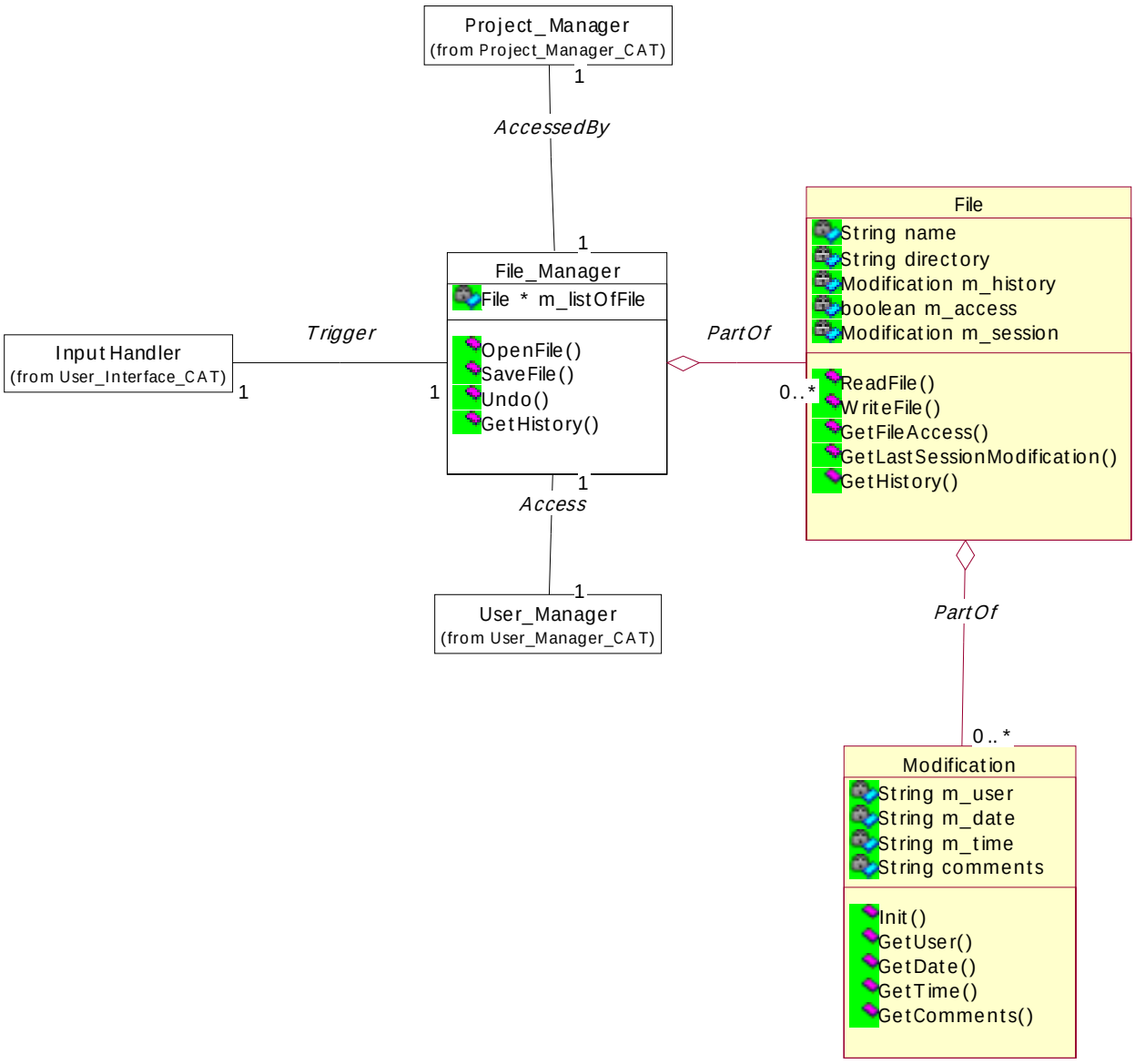
User_Manager_CAT_CCD



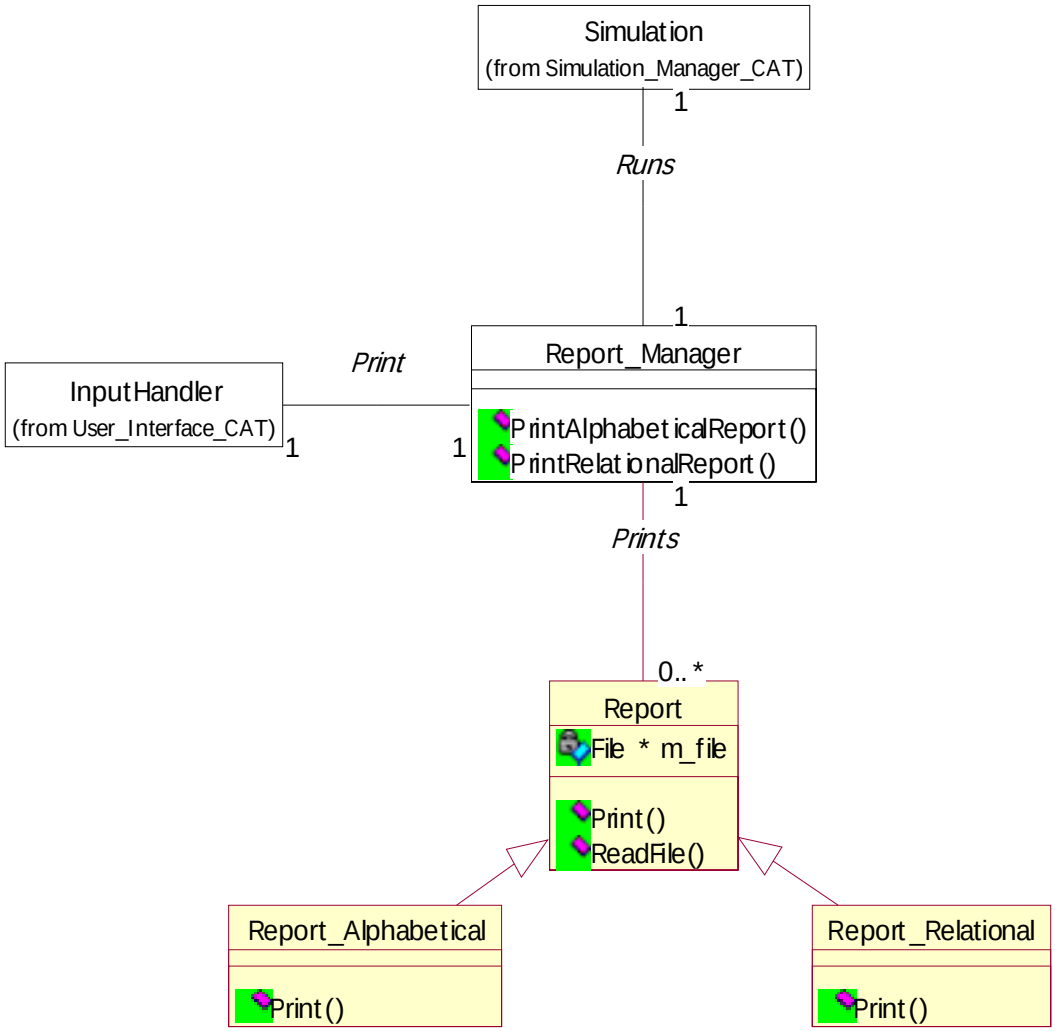
Project_Manager_CAT_CCD



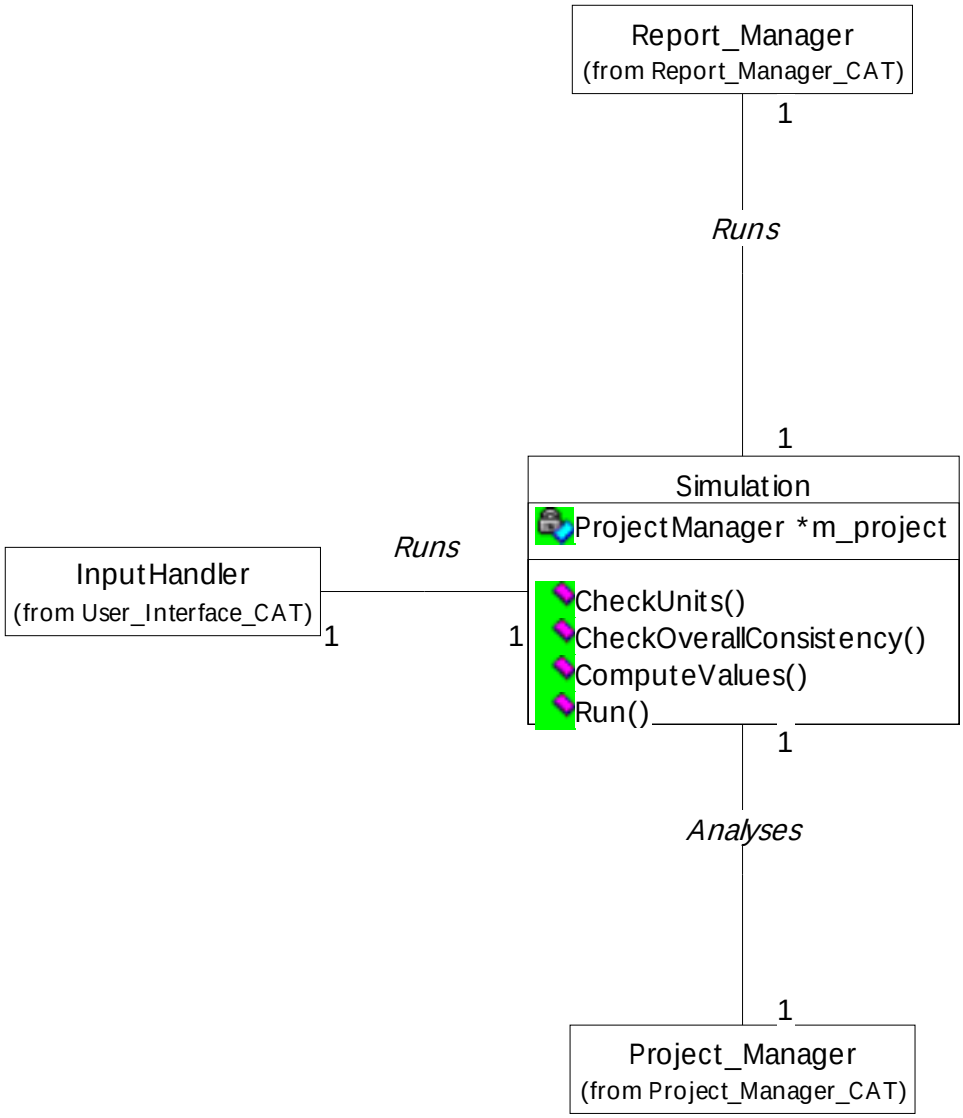
File_Manager_CAT_CCD



Report_Manager_CAT_CCD



Simulation_Manager_CAT_CCD



4 – Complete Class Specification (CCS)

Classes and their description, as well as attributes, methods, association and their descriptions are now documented.

A Class Specification is a description of a class that includes, at a minimum, a textual description of the class, the list of a class attributes and methods.

Category: PROJECT_MANAGER_CAT

Class: Object_Manager

Attributes:

Name	Type	Description
M_ptrListOfObjects	List of Object	Contains a list of Object

Methods:

Name	CreateObject
Description	Create a new object in memory

Name	AddObjectInList
Description	Add an existing object into the list of objects m_ptrListOfObjects

Name	AddAttribute
Description	Add an attibute to an existing object

Name	FindObject
Description	Find a specific object into the list of objects m_ptrListOfObjects

Name	DeleteAttibute
Description	Delete an attribute from an existing object

Name	ModifyAttribute
Description	Modify an attribute from an existing object

Name	BreakdownObject
Description	Breakdown an object into its different child objects

Name	DeleteObject
Description	Delete an existing object

Name	AddObject
Description	Add an existing object as a child to an another existing object

Class: Relationship_Manager

Attributes:

Name	Type	Description
M_ptrListOfRelationships	List of Relationship	Contains a list of Relationship

Methods:

Name	DefineRelationship
Description	Create a new relationship in memory

Name	AddRelationship
Description	Add an existing relationship into the list of realtionships m_ptrListOfRelationships

Name	ModifyRelationship
Description	Modify an existing Relationship in memory

Name	FindRelationship
Description	Find a specific relationship into the list of relationships m_ptrListOfRelationships

Name	FindIndirectRelationship
Description	Find the relationship formula between two objects

Class: Library_Manager

Attributes:

Name	Type	Description

Methods:

Name	ShowLibrary
Description	Display the different libraries stored in the database

Name	RetrieveRelation
Description	It retrieves a specific Relationship from a library

Name	AddLibrary
Description	Add a new library in the database

Name	AddRelationshipToLibrary
Description	Add a new relationship to a library

Class: Object

Attributes:

Name	Type	Description
M_ListOfChildren	List of Object	Contains all the Object's children
M_father	Pointer to an object	Points to the parent Object
Name	String	Name of the object
M_attrib	List of Attribute	Contains all the Object's attributes

Methods:

Name	InitObject
Description	Initialize the name of the object

Name	AddAttribute
Description	Add an attribute to an object

Name	DeleteAttribute
Description	Delete an attribute of an Object

Name	SetAttribute
Description	Modify an attribute of the Object

Name	GetChildren
Description	Retrieve the list of children

Name	AddChild
Description	Add a new child Object to the list of children m_listOfChildren

Name	DeleteChild
Description	Delete a child Object from the list of children m_listOfChildren

Class: Attribute

Attributes:

Name	Type	Description
Name	String	Name of the Attribute
Type	String	Type of the Attibute
Content	String	Content of the Attribute

Methods:

Name	InitAttribute
Description	Initialize the name, type an content of the Attribute

Name	GetName
Description	Get name of the Attribute

Name	GetType
Description	Get type of the Attribute

Name	GetContent
Description	Get content of the Attribute

Class: Relationship

Attributes:

Name	Type	Description
M_listOfObject	List of Object	List of objects linked by the relationship formula
M_myLibrary	Pointer to Library manager	Contains a pointer to the library manager
formula	String	Formula linking the two objects

Methods:

Name	VerifyConsistency
Description	Verify consistency of the relationship
Input	None
Output	Return 1 if no error return 0 otherwise

Name	VerifyUnit
Description	Verify the differents units used in the relationships

Name	IdentifyRelationship
Description	Return the type of the relationship

Name	InitRelationship
Description	Initialize the relationship

Name	GetFormula
Description	Get the relationship formula

Category: USER_MANAGER_CAT

Class: User_Manager

Attributes:

Name	Type	Description
M_ptrListOfUsers	List of User	Contains a list of users

Methods:

Name	AddFileToUser
Description	Add an existing file into a list of file currently used by a specific user

Name	Log
Description	Check password of a user and log him to the system (add him to the list of users m_ptrListOfUsers)

Class: User

Attributes:

Name	Type	Description
Login	String	Username of the user
Password	String	Password of the user
M_listOfFileRead	File*	List of the files the user can only read. (read access only)
M_listOfFileWrite	File*	List of the files the user can read and write. (read/write access)

Methods:

Name	GetPassword
Description	Retrieve the password

Name	AddFile
Description	Add a new file to the list of read or write access

Category: USER_INTERFACE_CAT

Class: InputHandler

Attributes:

Name	Type	Description

Methods:

Name	OnAddObject
Description	Triggered when clicking on Add object

Name	GetName
Description	Popup a dialog box and ask for a name

Name	GetType
Description	Popup a dialog box and ask for a type

Name	GetContent
Description	Popop a dialog box and ask fo content

Name	OnAddAttribute
Description	Triggered when clicking on Add attribute

Name	OnDeleteAttribute
Description	Triggered when clicking on Delete attribute

Name	OnModifyAttribute
Description	Triggered when clicking on Modify attribute

Name	OnDbClickObject
Description	Triggered when double clicking on an object

Name	OnCreateObject
Description	Triggered when clicking on Create object

Name	OnAddObject
Description	Triggered when clicking on Add object

Name	SelectObject
Description	Popup a dialog box and ask for selecting an object

Name	OnDeleteObject
Description	Triggered when clicking on Delete object

Name	OnShowLibraries
Description	Triggered when clicking on Show Libraries

Name	OnAddLibrary
Description	Triggered when clicking on Add Library

Name	GetLibraryName
Description	Popup a dialog box and ask for library name

Name	OnAddRelationshipToLibrary
Description	Triggered when clicking on Add Relationship to Library

Name	OnLink
Description	Triggered when dragging the mouse between two objects

Name	OnDefineRelationship
Description	Triggered when clicking on Define Relationship

Name	OnModifyRelationship
Description	Triggered when clicking on Modify Relationship

Name	OnFindRelationship
Description	Triggered when clicking on Find Relationship

Name	OnOpenFile
Description	Triggered when clicking on Open File

Name	OnSaveFile
Description	Triggered when clicking on Save File

Name	OnPrintAlphabeticalReport
Description	Triggered when clicking on Print Alphabetical Report

Name	OnPrintRelationalReport
Description	Triggered when clicking on Print Relational Report

Name	Undo
Description	Triggered when clicking on Undo

Name	OnHistory
Description	Triggered when clicking on Show History

Name	OnLog
Description	Triggered when clicking on Login the system

Name	OnRunSimulation
Description	Triggered when clicking on Run Simulation

Class: Display

Attributes:

Name	Type	Description

Methods:

Name	LinkObjects
Description	Display a line linking two objects

Category: FILE_MANAGER_CAT

Class: File_Manager

Attributes:

Name	Type	Description
M_listOfFile	List of File	Contains a list of files

Methods:

Name	OpenFile
Description	Open a file specified by its location(directory) and name.

Name	SaveFile
Description	Save the current opened file, save all the objects and relationships defined in the file.

Name	Undo
Description	Restore the system to its previous state

Name	GetHistory
Description	Retrieve the sets of action the user perform on each file.

Class: File

Attributes:

Name	Type	Description
Name	String	Name of the file.
Directory	String	Location of the file.
m_history	List of modification	List of all modifications performed on the file since its creation.
m_session	List of modification	List of modification performed on a file since the beginning of the current session.
m_access	boolean	=1, if the file is not opened by any other user. =0, if the file is opened by another user.

Methods:

Name	ReadFile
Description	Open a file with read access

Name	WriteFile
Description	Open a file with write access

Name	GetFileAccess
Description	Retrieve the type of access the user can open the fil

Name	GetLastSessionModification
Description	Retrieve the last modification of the list of modification performed in the session (from the list m_session)

Name	GetHistory
Description	Retrieve the list of modification performed on the file since it creation (the list m_history)

Class: Modification

Attributes:

Name	Type	Description
m_user	String	Name of the user who performed the modification
m_date	String	Date when the modification occured
m_time	String	Time when the modification occured
comments	String	Modification details

Methods:

Name	Init
Description	Initialize a new modification

Name	GetUser
Description	Retrieve the name of the user who performed the modification

Name	GetDate
Description	Retrieve the date when the modification occurred

Name	GetTime
Description	Retrieve the time when the modification occurred

Name	GetComments
Description	Retrieve the Modification details

Category: REPORT_MANAGER_CAT

Class: Report_Manager

Attributes:

Name	Type	Description

Methods:

Name	PrintAlphabeticalReport ( )
Description	This method print in an alphabetical order a report related to the name of the objects, the relationship between the objects as it appear in the file related to the project

Name	PrintRelationalReport ( )
Description	This method print in the report related to the relationship between the objects as it appear in the file related to the project

Class: Report

Attributes:

Name	Type	Description
m_file	FILE	Pointer to file

Methods:

Name	Print
Description	This method activate the print job in either one of the sub classes listed below

Name	ReadFile
Description	This method read entirely the file pointed to by the file pointer

Class: Report_Alphabetical

Attributes:

Name	Type	Description

Methods:

Name	Print
Description	This method activate the print the content of the file system in alphabetical order. Print the names, relationship and formula relating the object

Class: Report_Relational

Attributes:

Name	Type	Description

Methods:

Name	Print
Description	This method print the content of the file system without any order. Print the names, relationship and formula relating the objects.

Category: SIMULATION_MANAGER_CAT

Class: Simulation

Attributes:

Name	Type	Description
M_ProjectManager	ProjectManager	Pointer to the class project manager

Methods:

Name	CheckUnits
Description	Check the consistency of the units that would be used in the relationship

Name	CheckOverallConsistency
Description	Check the consistency of the relationship

Name	Compute Values
Description	Apply the relationship to compute certain values of the relationship

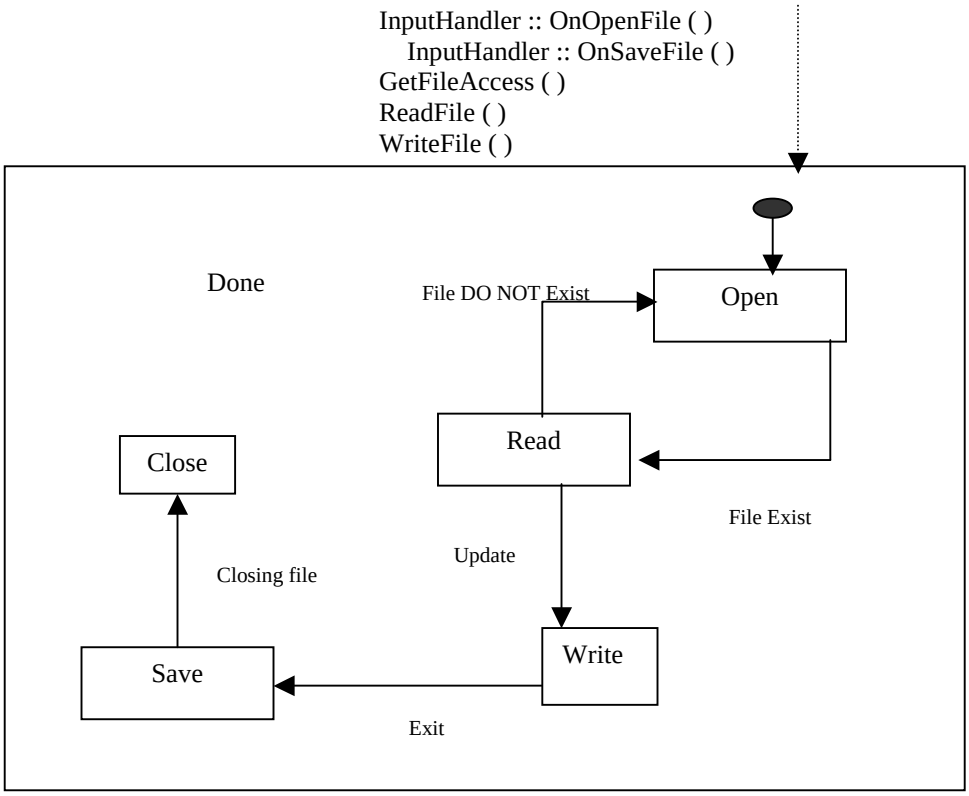
Name	Run
Description	Run in order the simulation for consistency. Execute in order CheckOverallConsistency ( ) , CheckUnits ( )

PHASE 6: Software Object Oriented Analysis, Dynamic View

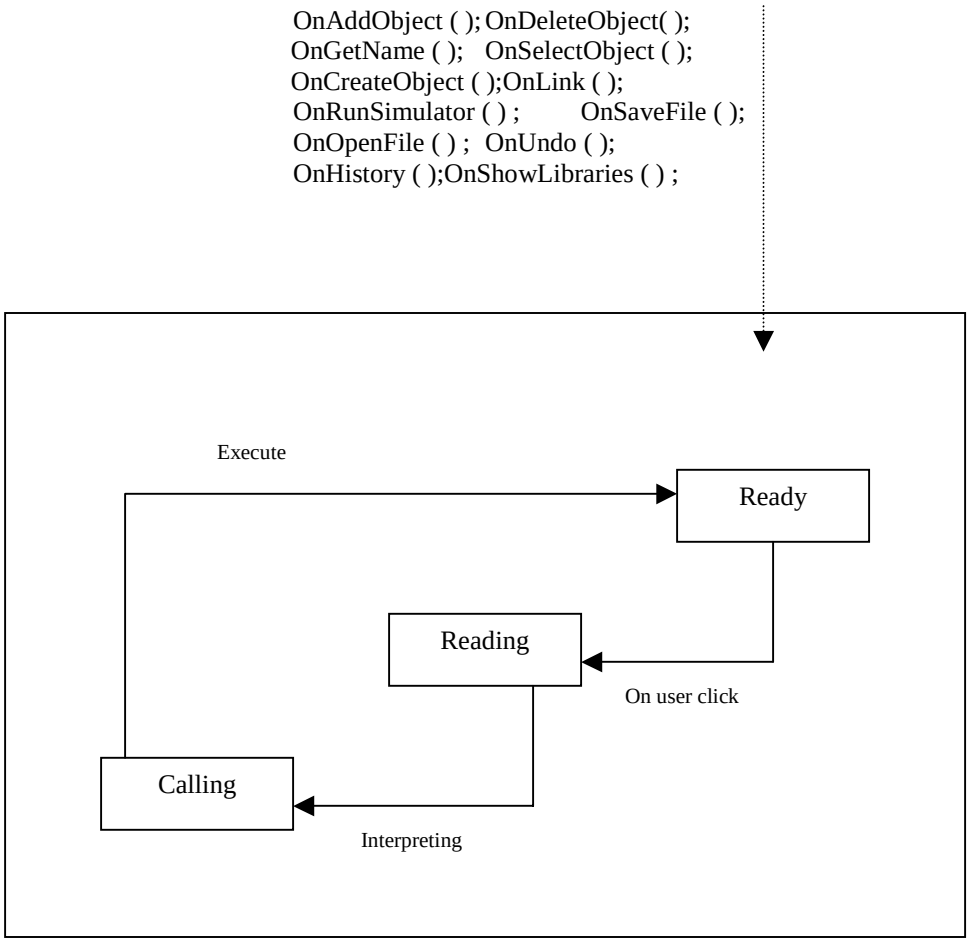
1 – Develop State Transition Diagram

A State Transition Diagram is a graphical representation of the life cycle of the instances of a class. STD's focus on the life cycle phases of a class, whose values represent the domain of values for an attribute of a class, typically named state.

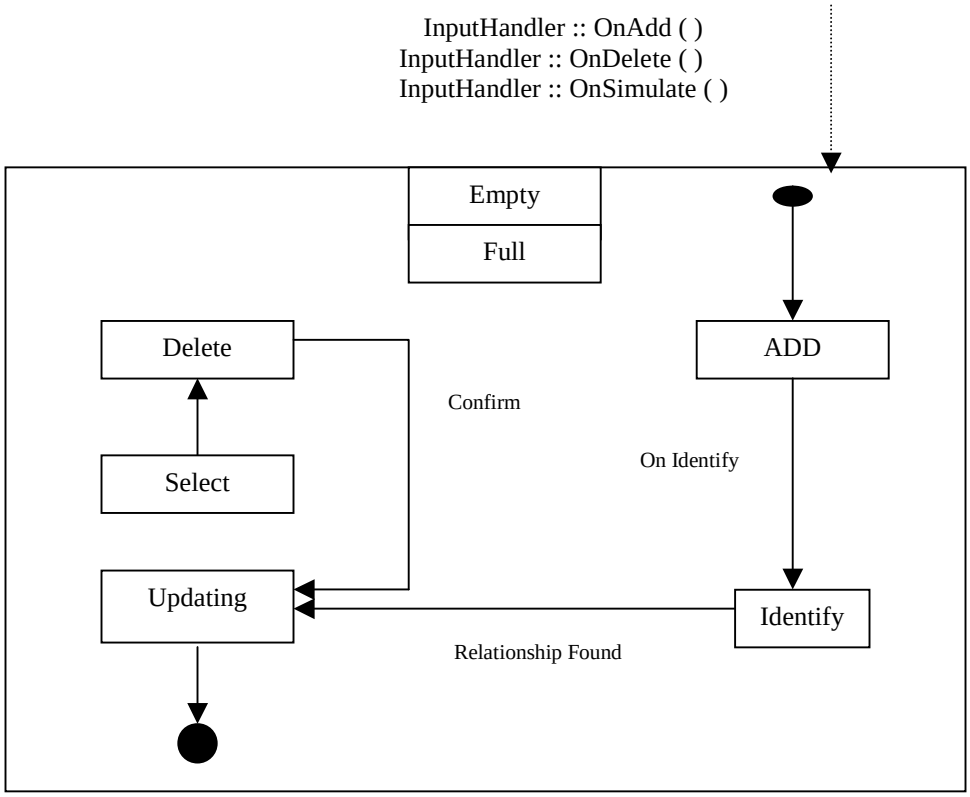
The purpose of this activity is to identify whether or not an STD is warranted. Not all classes warrant an STD.



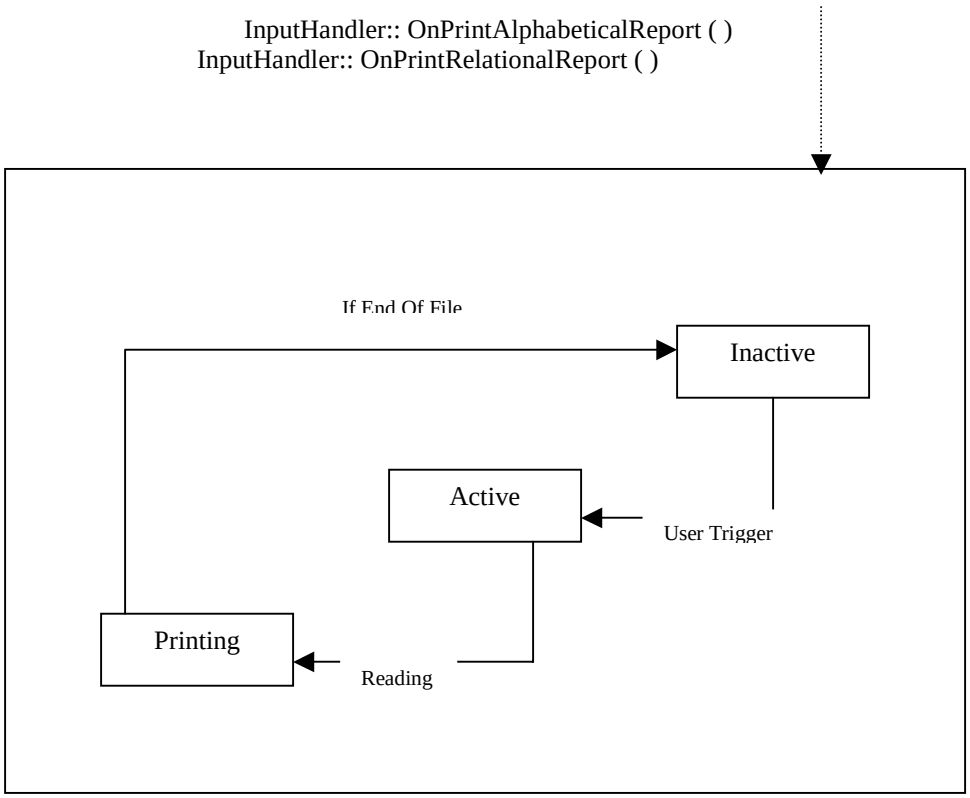
File_Manager_Class – State Transition Diagram



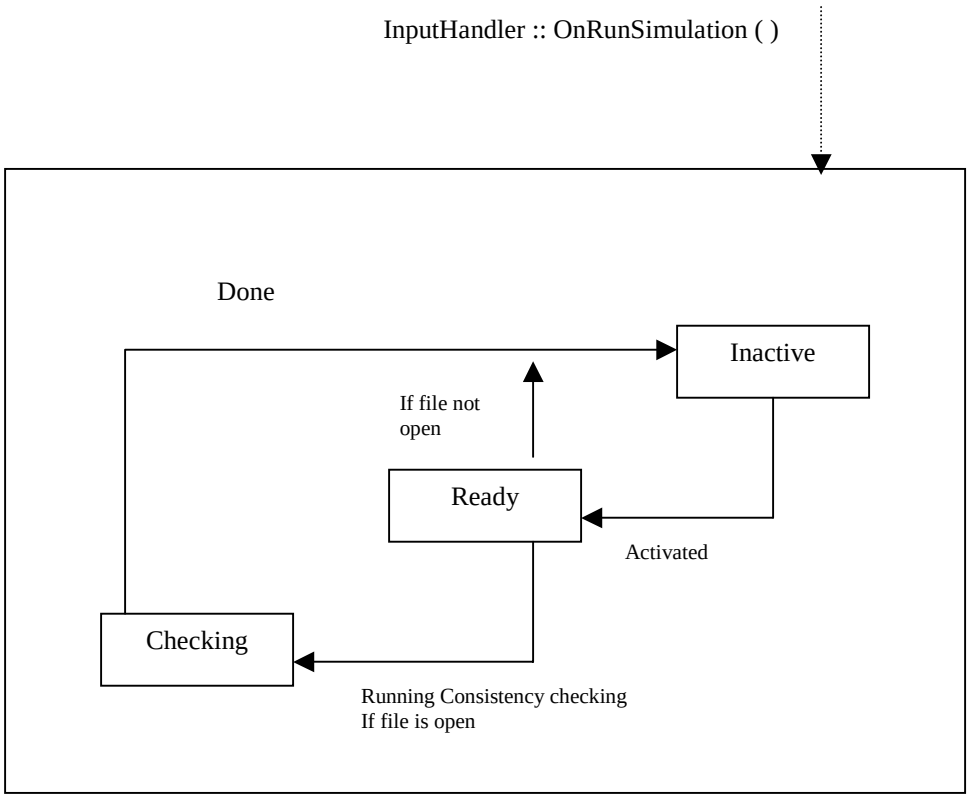
Input Handler Class - State Transition Diagram



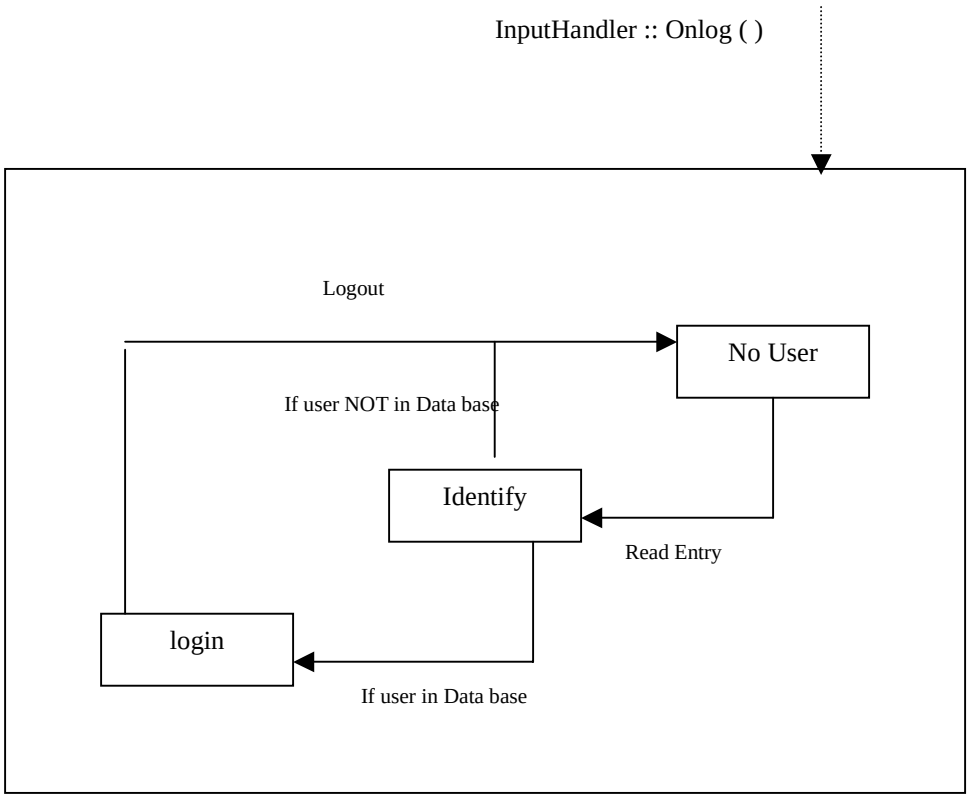
Project Manager Class - State Transition Diagram



Report_Manager_Class - State Transition Diagram



Simulation Class: State Transition Diagram

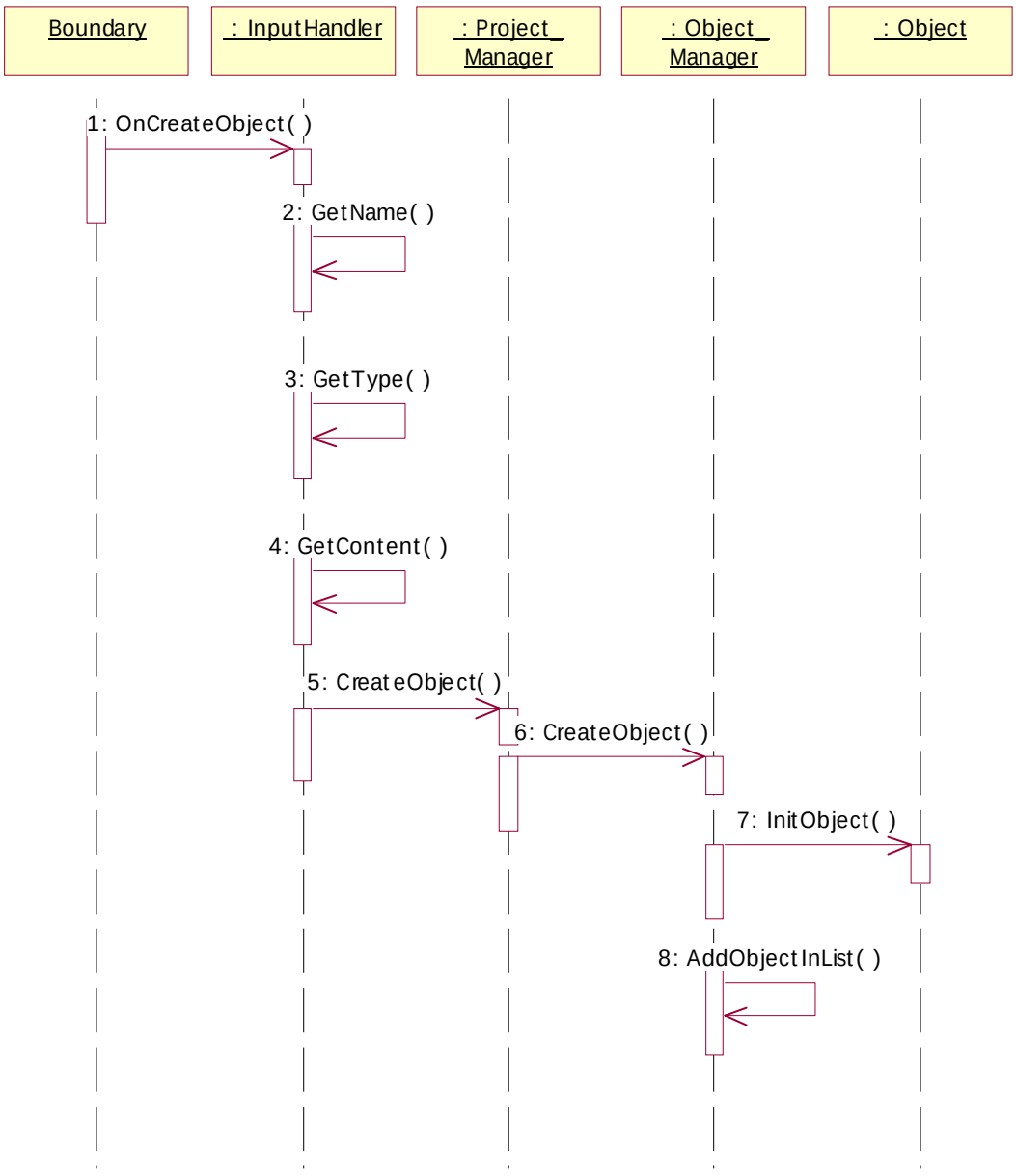


User_Manager_Class - State Transition Diagram

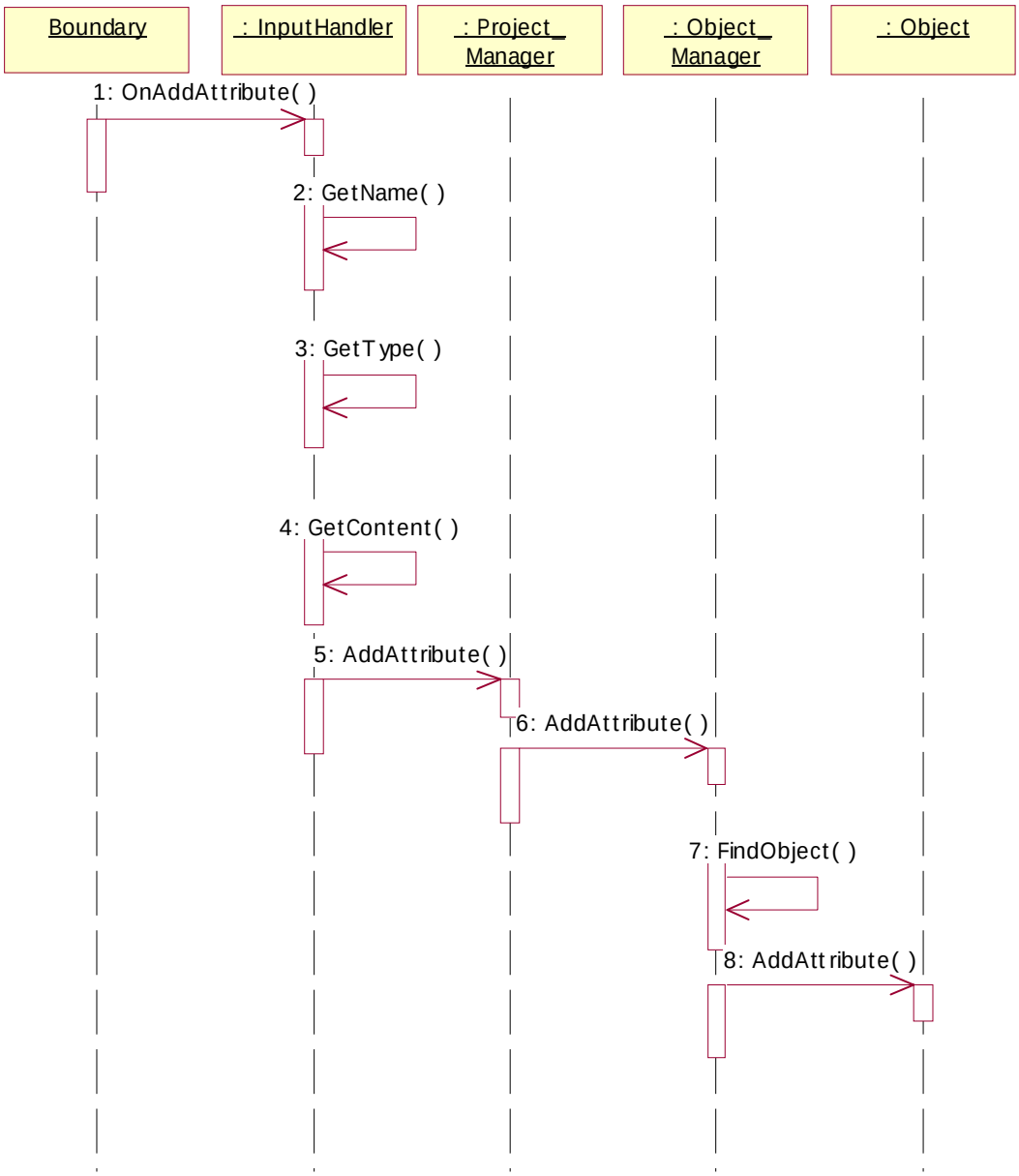
2 – Refining Category Interaction Diagram

The purpose of refining the CID and developing a software level ID is to show at a finer (more precise) level of detail all the classes and methods within each class that are required to implement a Use Case.

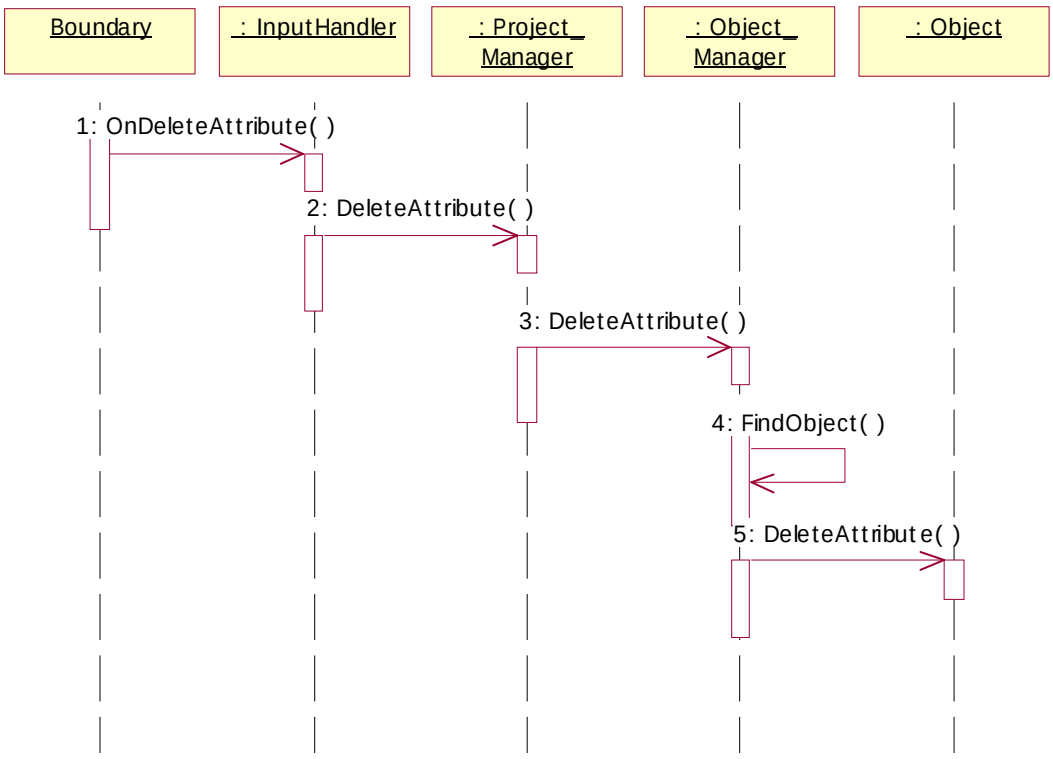
UseCase01_ Scenario1



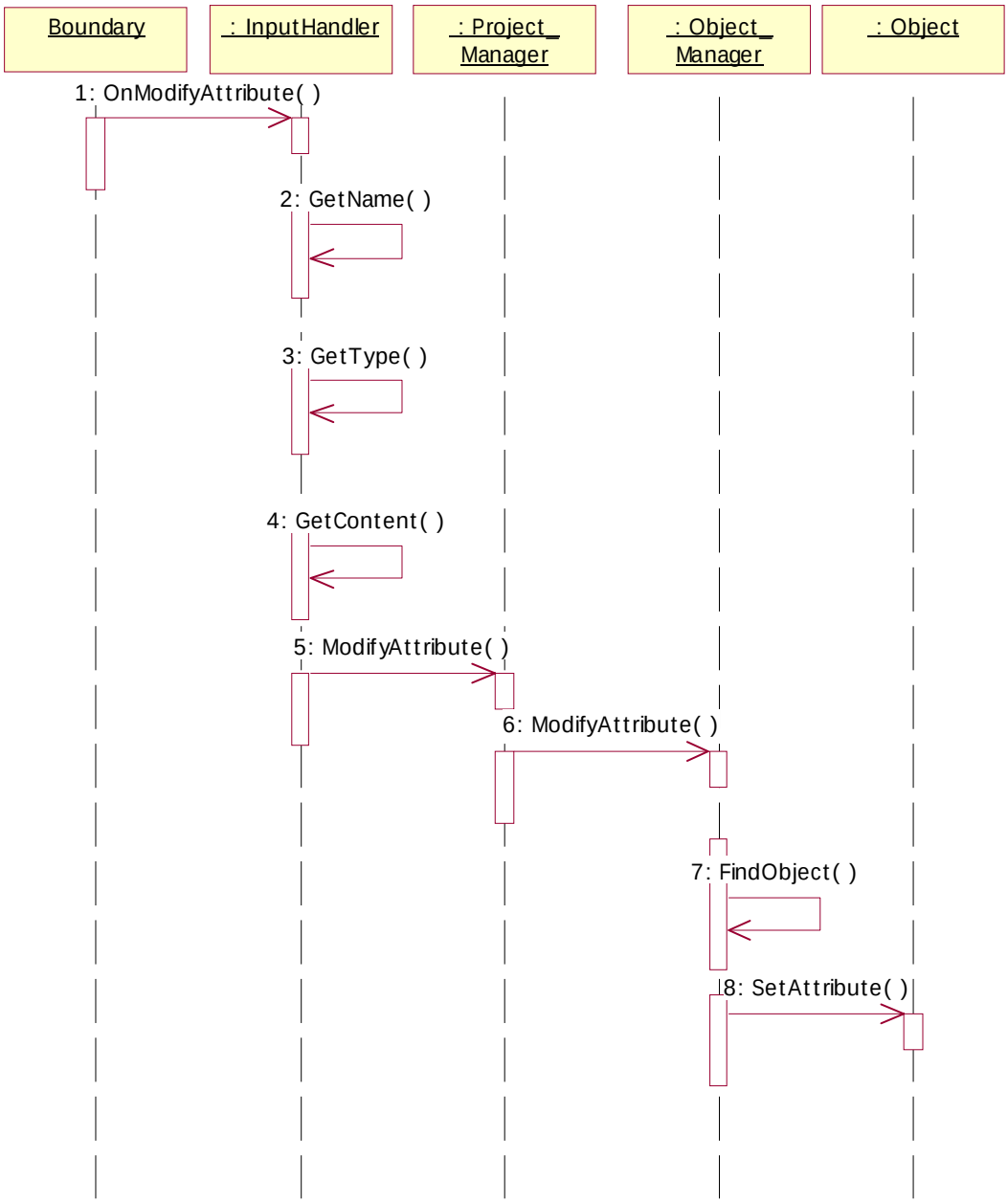
UseCase01_Scenario2



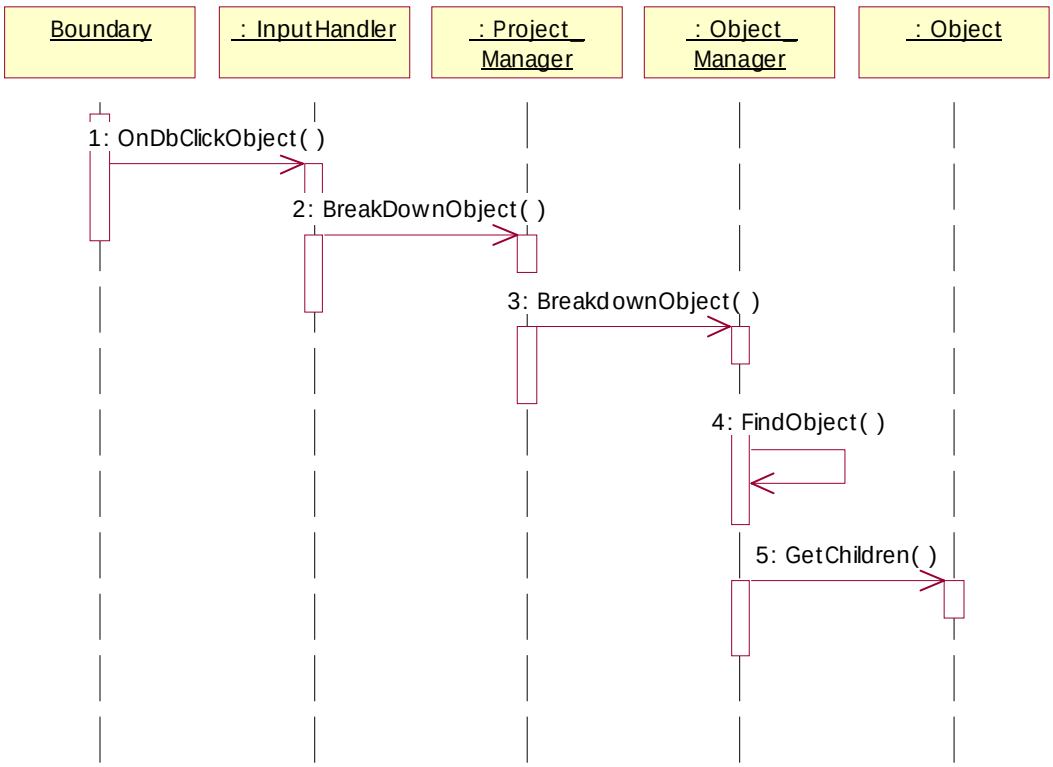
UseCase 01 _ Scenario 3



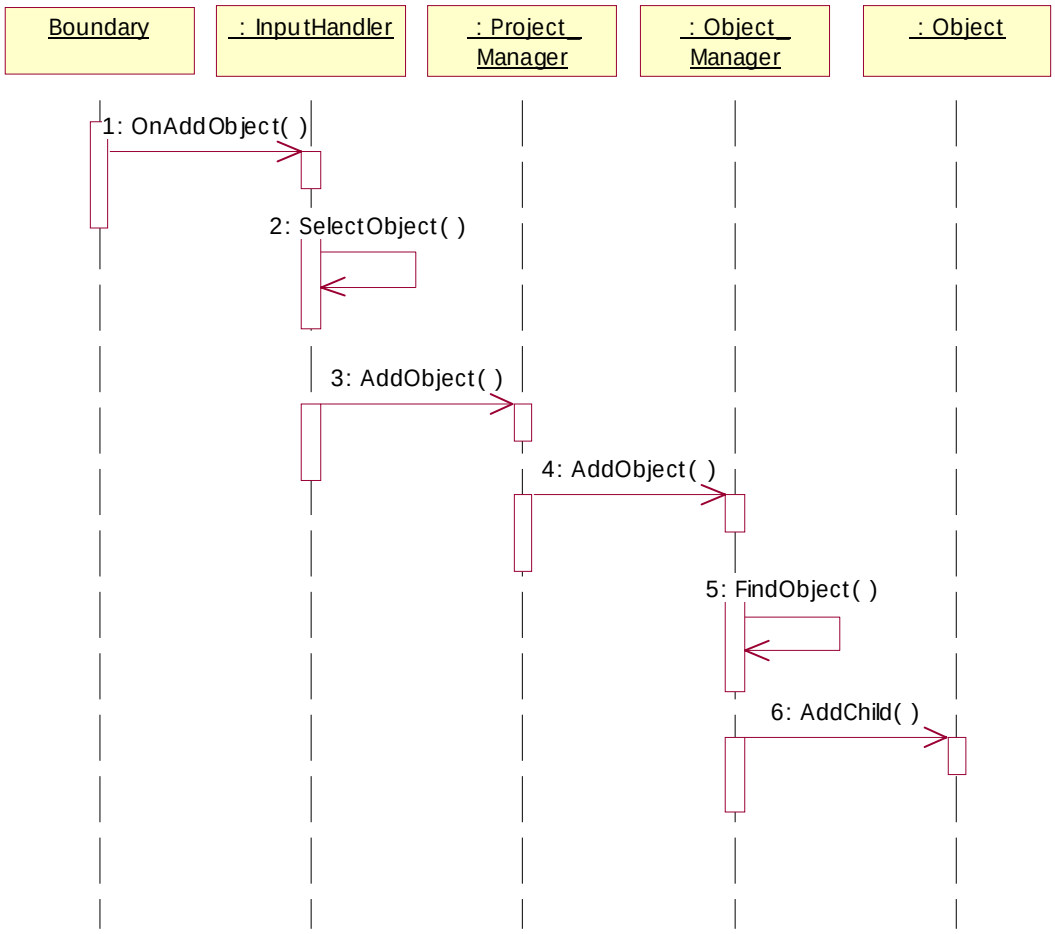
UseCase 01 _Scenario 4



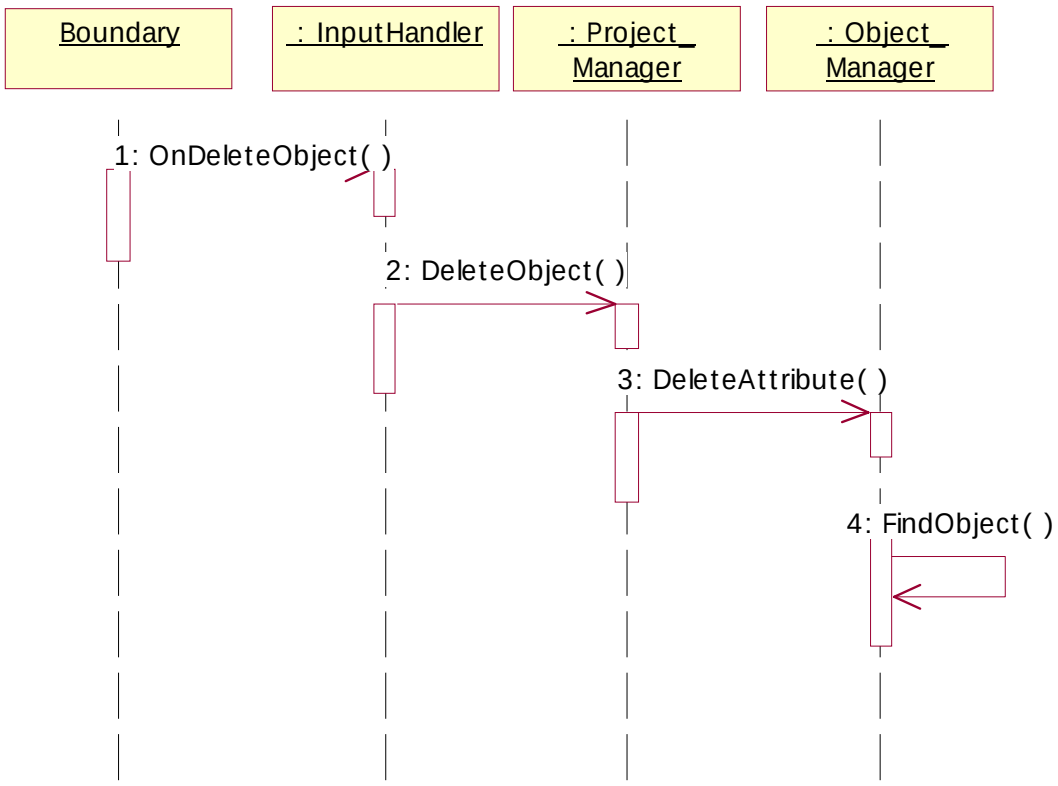
UseCase 01 - Scenario5



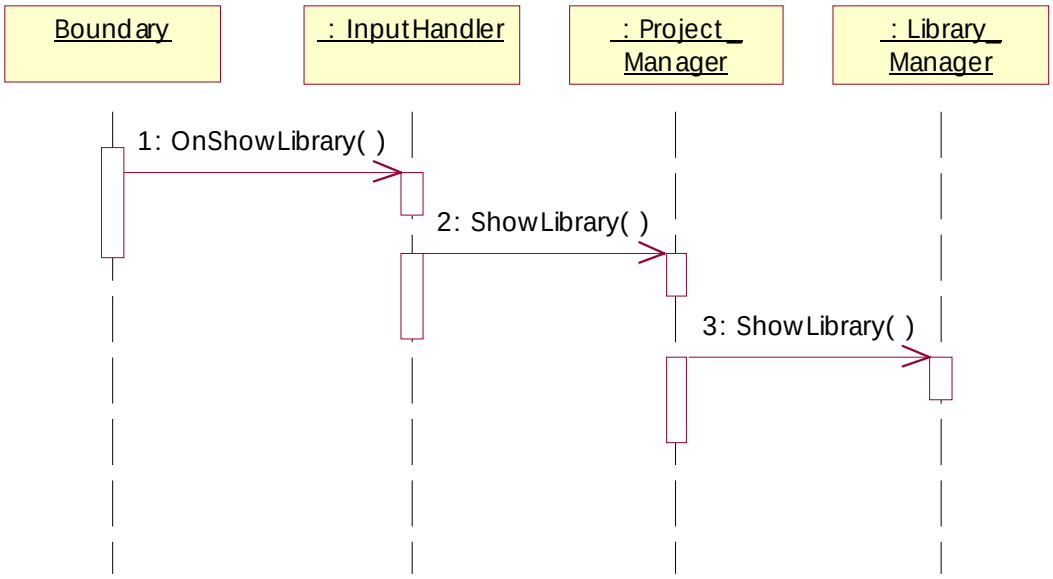
UseCase01 _ Scenario6



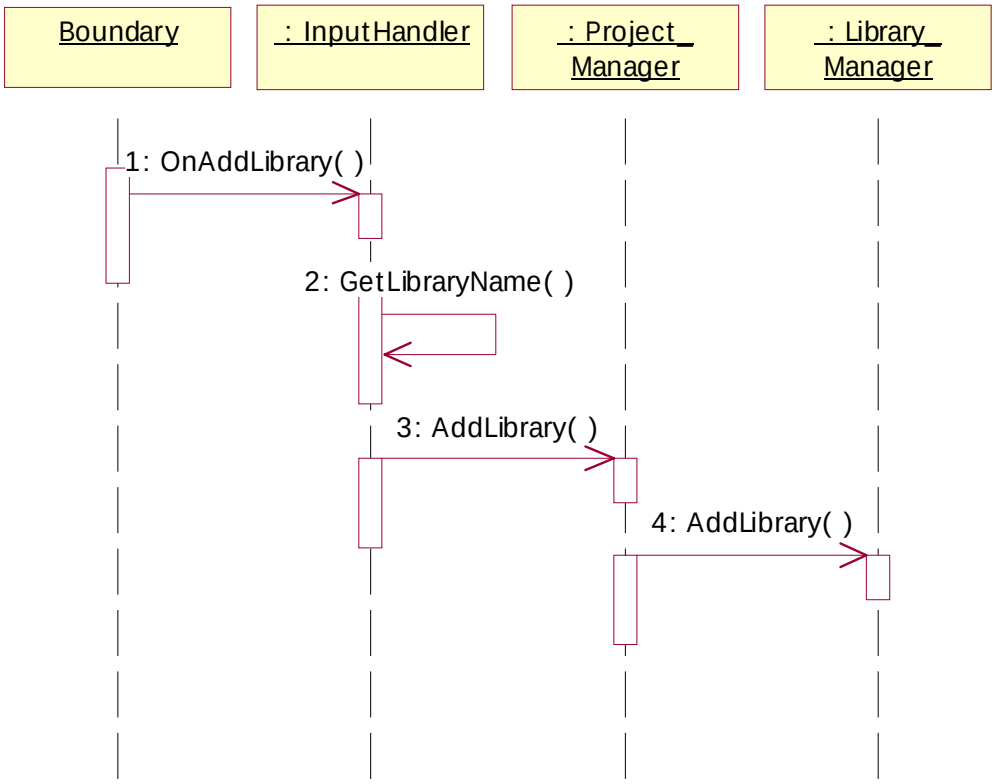
UseCase01 _ Scenario7



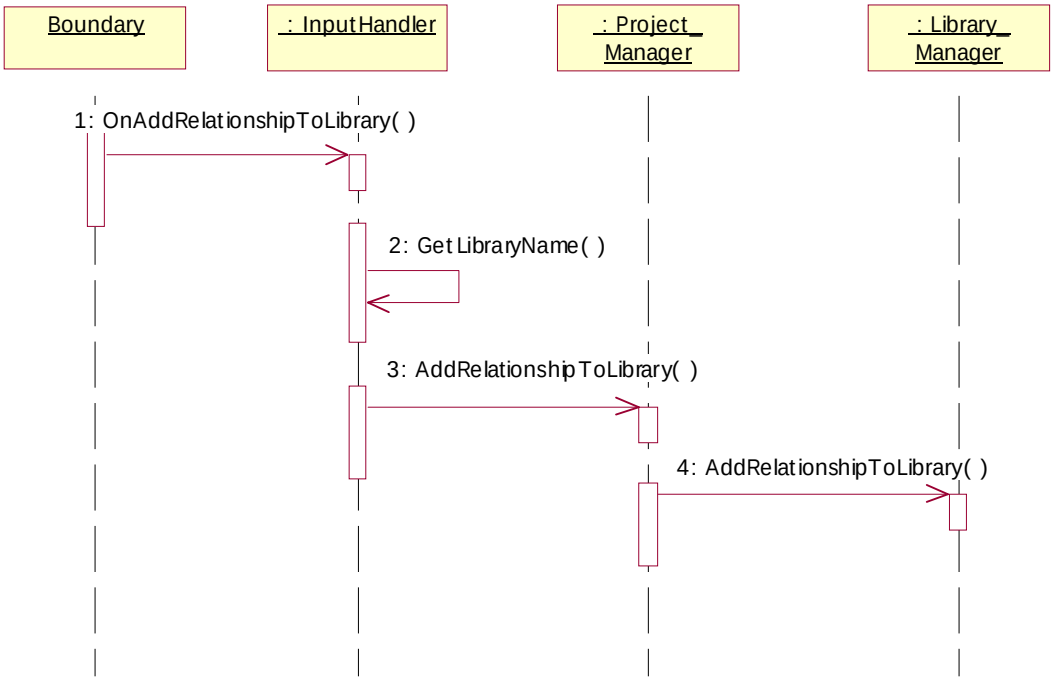
UseCase02 _ Scenario1



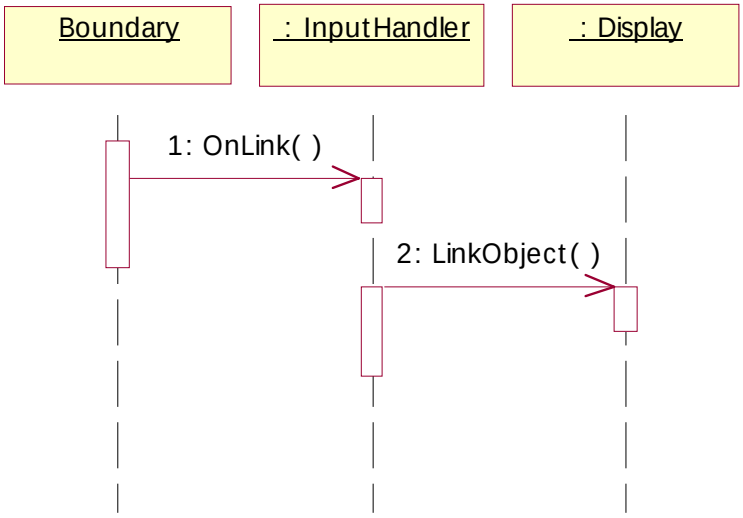
UseCase02 _ Scenario2



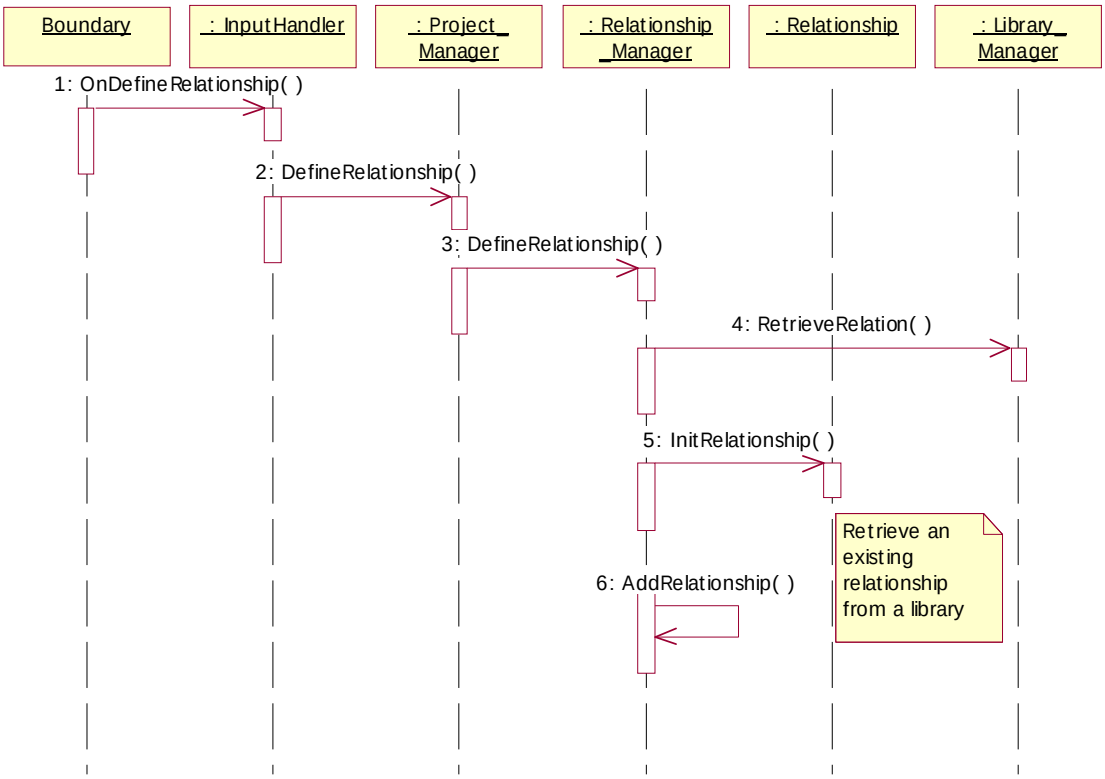
UseCase02_Scenario3



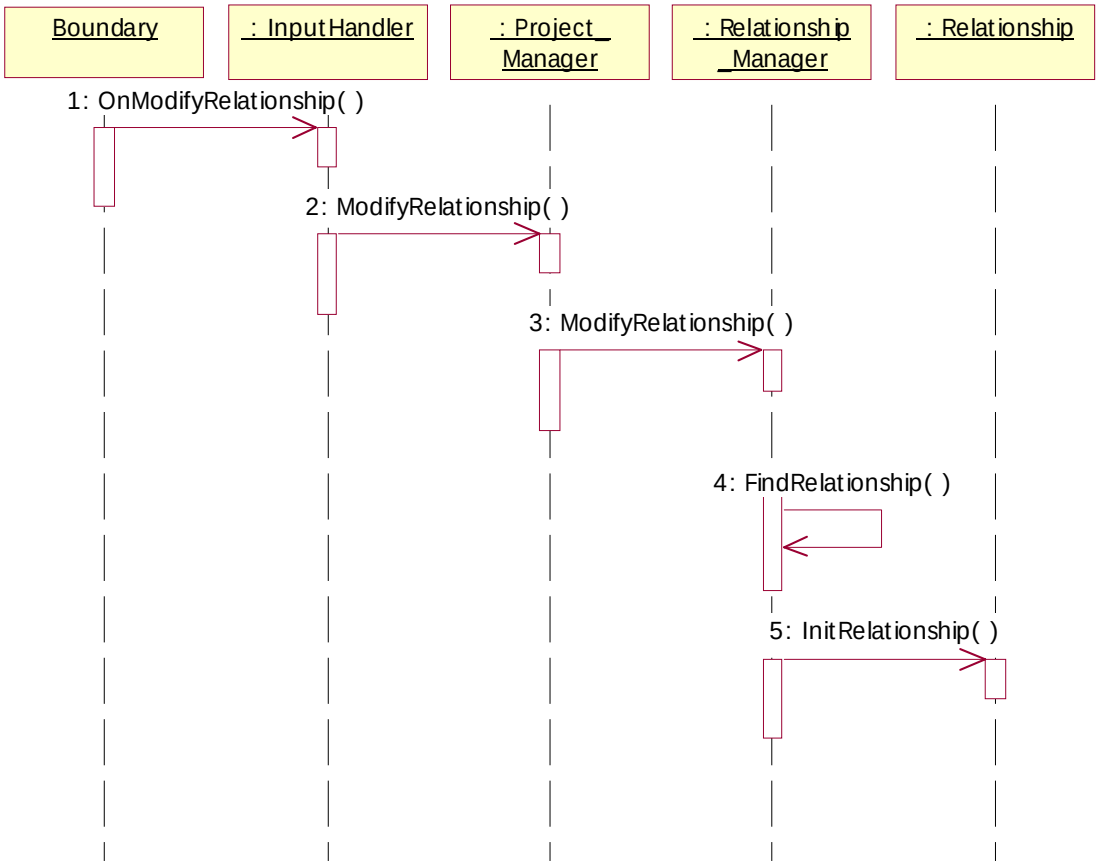
UseCase03 _ Scenario1



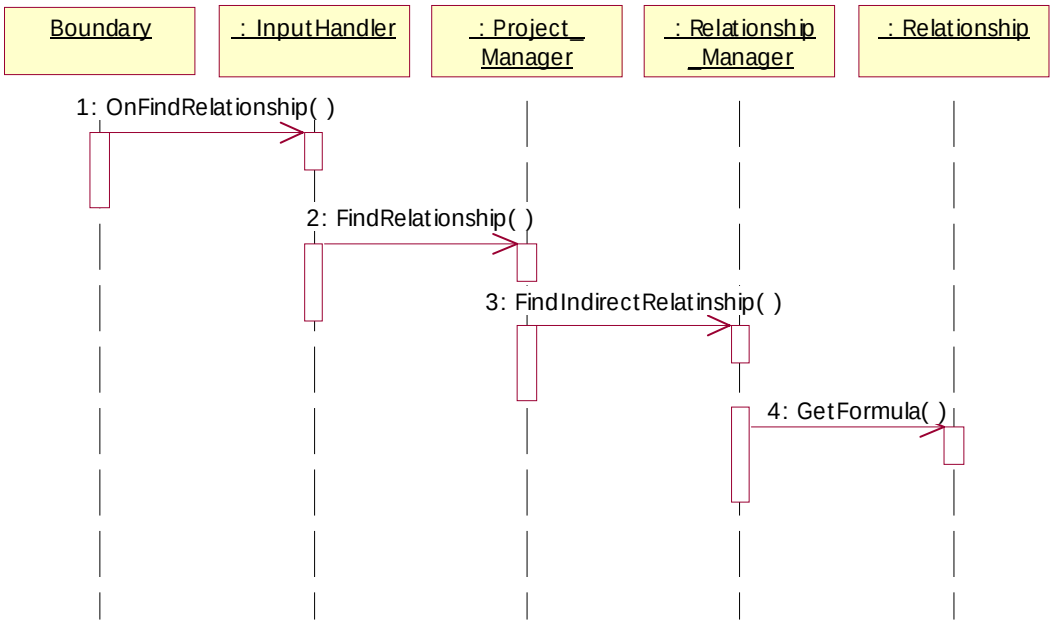
UseCase03_ Scenario2



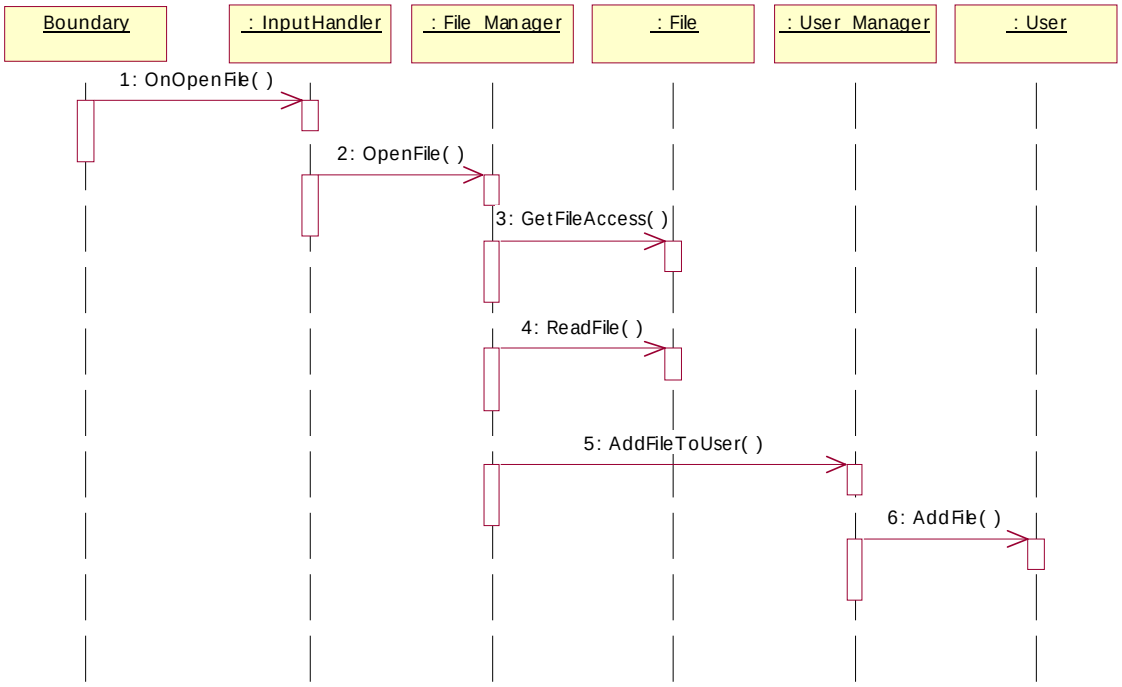
UseCase03 Scenario3



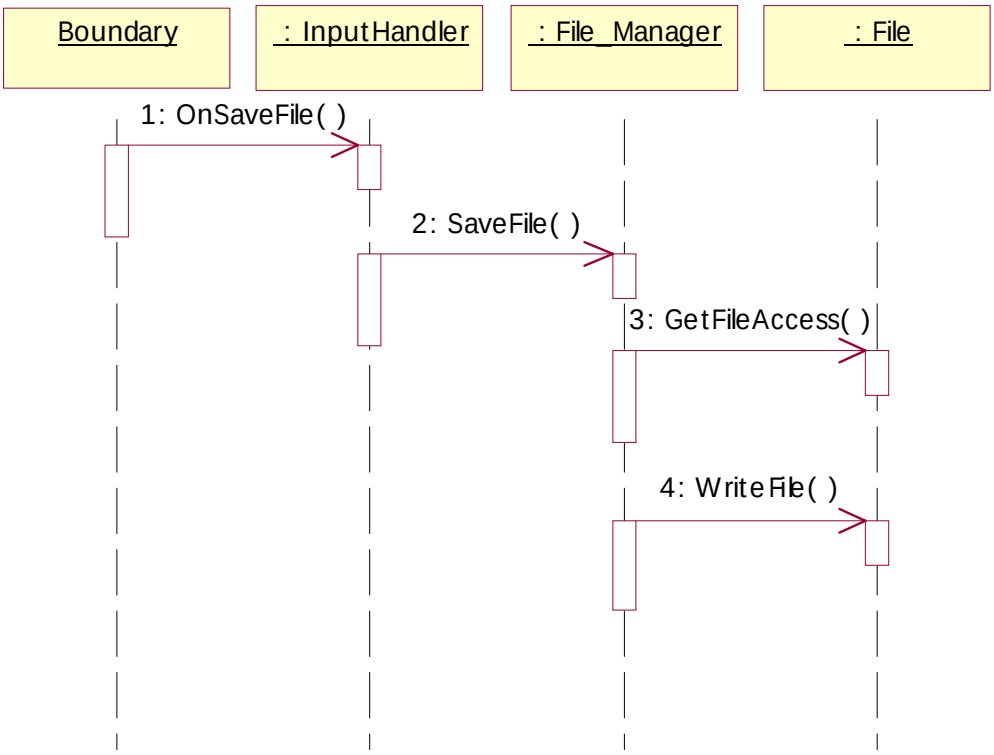
UseCase03 _ Scenario4



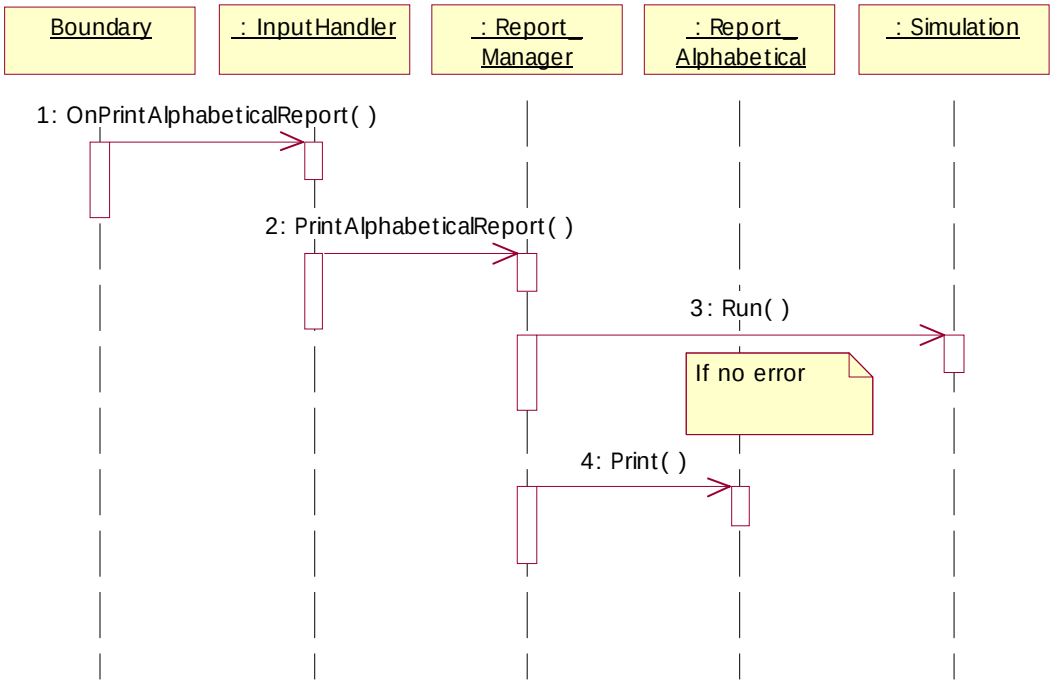
UseCase04 Scenario1



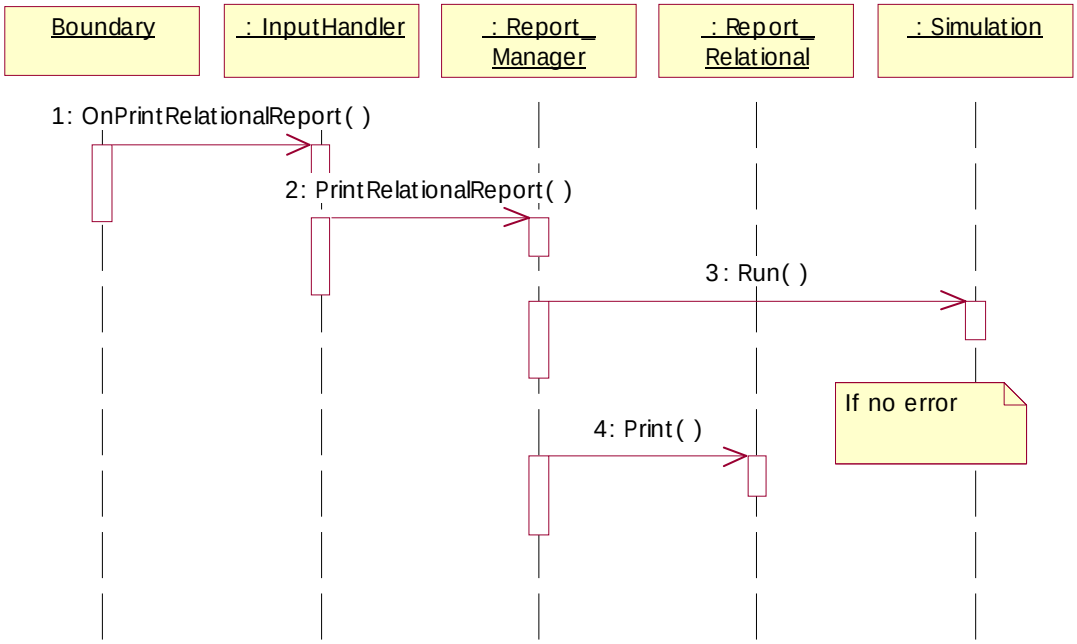
UseCase04_Scenario2



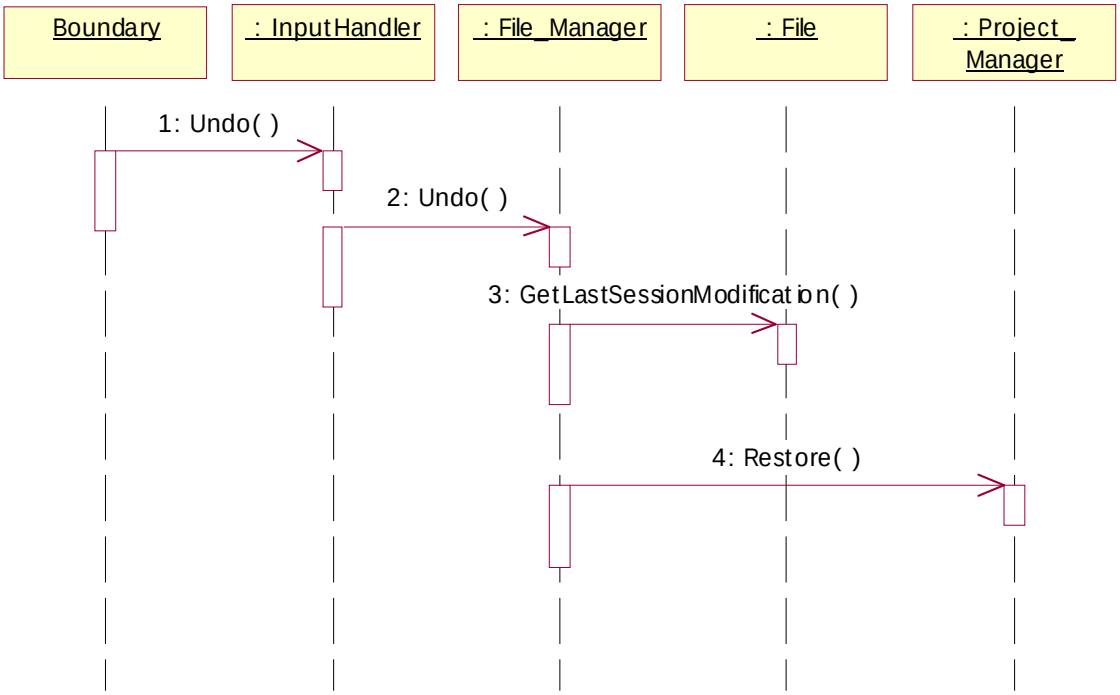
UseCase05 - Scenario1



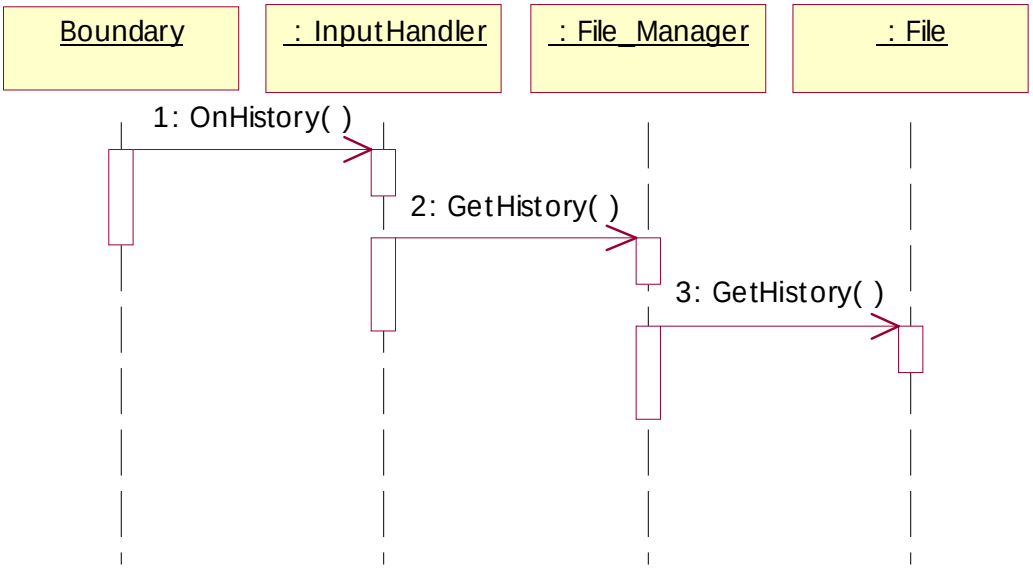
UseCase05 _ Scenario2



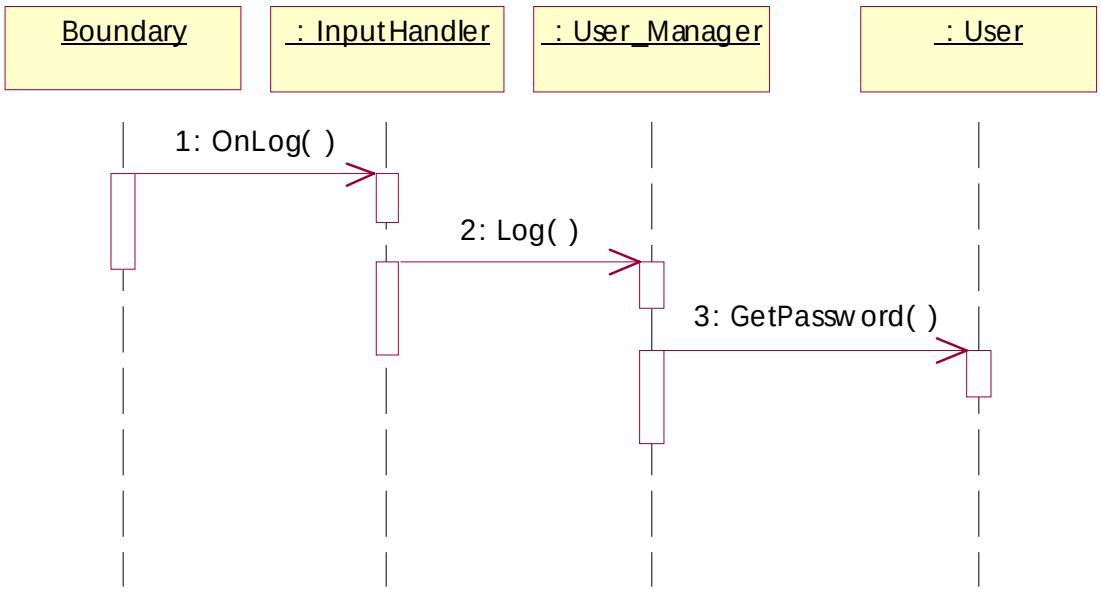
UseCase06 _ Scenario1



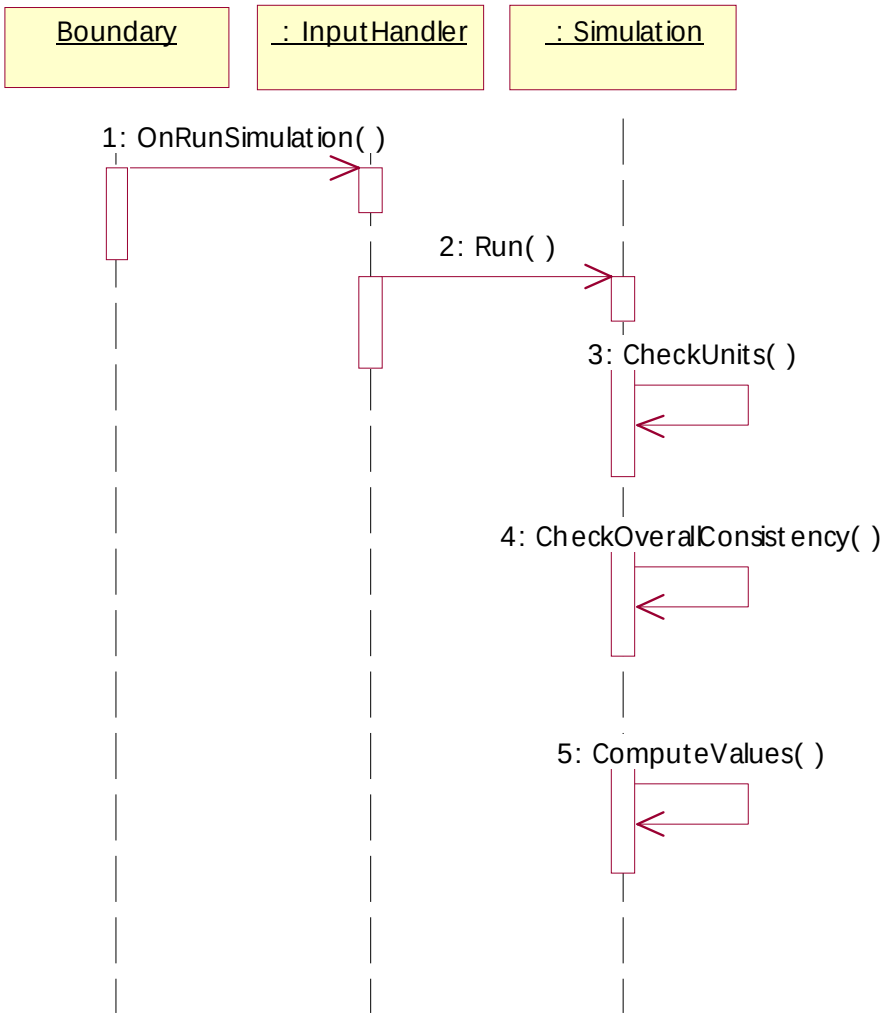
UseCase06_Scenario2



UseCase07 Scenario 1



UseCase08 Scenario1



3 – Refining Class Specification

The purpose of this activity is to keep the class description current and to reflect the information gleaned from developing STDs and IDs.

Category: PROJECT_MANAGER_CAT

Class: Object_Manager

Attributes:

Name	Type	Description
M_ptrListOfObjects	List of Object	Contains a list of Object

Methods:

Name	CreateObject
Description	Create a new object in memory
Input	Name of object
Output	None

Name	AddObjectInList
Description	Add an existing object into the list of objects m_ptrListOfObjects
Input	Pointer to the Object to add
Output	None

Name	AddAttribute
Description	Add an attibute to an existing object
Input	Name, type, content of attibute
Output	None

Name	FindObject
Description	Find a specific object into the list of objects m_ptrListOfObjects
Input	Name of the object
Output	Pointer to the object

Name	DeleteAttibute
Description	Delete an attribute from an existing object
Input	Name of the attribute to delete
Output	None

Name	ModifyAttribute
Description	Modify an attribute from an existing object
Input	Old name, new name, new type, new content of the attribute
Output	None

Name	BreakdownObject
Description	Breakdown an object into its different child objects
Input	Name of object to breakdown
Output	None

Name	DeleteObject
Description	Delete an existing object

Input	Name of object to delete
Output	None

Name	AddObject
Description	Add an existing object as a child to an another existing object
Input	Name of the object to add, name of the father object
Output	None

Class: Relationship_Manager

Attributes:

Name	Type	Description
M_ptrListOfRelationships	List of Relationship	Contains a list of Relationship

Methods:

Name	DefineRelationship
Description	Create a new relationship in memory
Input	Name and type of relationship
Output	None

Name	AddRelationship
Description	Add an existing relationship into the list of realtionships m_ptrListOfRelationships
Input	Pointer to the Relationship to add
Output	None

Name	ModifyRelationship
Description	Modify an existing Relationship in memory
Input	Old name, new name and new type of the relationship
Output	None

Name	FindRelationship
Description	Find a specific relationship into the list of relationships m_ptrListOfRelationships
Input	Name of the relationship
Output	Pointer to the relationship

Name	FindIndirectRelationship
Description	Find the relationship formula between two objects
Input	Name of the two objects
Output	Formula linking the two objects

Class: Library_Manager

Attributes:

Name	Type	Description

Methods:

Name	ShowLibrary
Description	Display the different libraries stored in the database
Input	None
Output	None

Name	RetrieveRelation
Description	It retrieves a specific Relationship from a library
Input	Relationship name and library name
Output	Relationship

Name	AddLibrary
Description	Add a new library in the database
Input	Name of the file containing the library
Output	None

Name	AddRelationshipToLibrary
Description	Add a new relationship to a library
Input	Name of the relationship and name of the library
Output	None

Class: Object

Attributes:

Name	Type	Description
M_ListOfChildren	List of Object	Contains all the Object's children
M_father	Pointer to an object	Points to the parent Object
Name	String	Name of the object
M_attrib	List of Attribute	Contains all the Object's attributes

Methods:

Name	InitObject
Description	Initialize the name of the object
Input	Name of Object
Output	None

Name	AddAttribute
Description	Add an attribute to an object
Input	Name, type and content of Attribute
Output	None

Name	DeleteAttribute
Description	Delete an attribute of an Object
Input	Name of the attribute to delete
Output	None

Name	SetAttribute
Description	Modify an attribute of the Object
Input	Old name of the attribute, new name, type and content of the Attribute
Output	None

Name	GetChildren
Description	Retrieve the list of children
Input	None
Output	List of Object

Name	AddChild
Description	Add a new child Object to the list of children m_listOfChildren
Input	Pointer to an existing object
Output	None

Name	DeleteChild
Description	Delete a child Object from the list of children m_listOfChildren
Input	Pointer to an existing object from the list
Output	None

Class: Attribute

Attributes:

Name	Type	Description
Name	String	Name of the Attribute
Type	String	Type of the Attibute
Content	String	Content of the Attribute

Methods:

Name	InitAttribute
Description	Initialize the name, type an content of the Attribute
Input	Name, type and content of Attibute
Output	None

Name	GetName
Description	Get name of the Attribute
Input	None
Output	Name of the attribute

Name	GetType
Description	Get type of the Attribute
Input	None
Output	Type of the Attribute

Name	GetContent
Description	Get content of the Attribute
Input	None
Output	Content of the Attribute

Class: Relationship

Attributes:

Name	Type	Description
M_listOfObject	List of Object	List of objects linked by the relationship formula
M_myLibrary	Pointer to Library manager	Contains a pointer to the library manager
formula	String	Formula linking the two objects

Methods:

Name	VerifyConsistency
Description	Verify consistency of the relationship
Input	None
Output	Return 1 if no error return 0 otherwise

Name	VerifyUnit
Description	Verify the differents units used in the relationships
Input	None
Output	Return 1 if no error return 0 otherwise

Name	IdentifyRelationship
Description	Return the type of the relationship
Input	None
Output	Type of the relationship

Name	InitRelationship
Description	Initialize the relationship
Input	Formula and list of objects
Output	None

Name	GetFormula
Description	Get the relationship formula
Input	None
Output	Formula

Category: USER_MANAGER_CAT

Class: User_Manager

Attributes:

Name	Type	Description
M_ptrListOfUsers	List of User	Contains a list of users

Methods:

Name	AddFileToUser
Description	Add an existing file into a list of file currently used by a specific user
Input	Name of the file, name of the user, access mode of the file
Output	None

Name	Log
Description	Check password of a user and log him to the system (add him to the list of users m_ptrListOfUsers)
Input	User name, password
Output	Return 1 if user name and password match otherwise return 0

Class: User

Attributes:

Name	Type	Description
Login	String	Username of the user
Password	String	Password of the user
M_listOfFileRead	File*	List of the files the user can only read. (read access only)
M_listOfFileWrite	File*	List of the files the user can read and write. (read/write access)

Methods:

Name	GetPassword
Description	Retrieve the password
Input	String containing the username
Output	String containing Password

Name	AddFile
Description	Add a new file to the list of read or write access
Input	File to add to the list User access (boolean =0 if read access, =0 if write acess)
Output	none

Category: USER_INTERFACE_CAT

Class: InputHandler

Attributes:

Name	Type	Description

Methods:

Name	OnAddObject
Description	Triggered when clicking on Add object
Input	None
Output	None

Name	GetName
Description	Popup a dialog box and ask for a name
Input	None
Output	Name

Name	GetType
Description	Popup a dialog box and ask for a type
Input	None
Output	Type

Name	GetContent
Description	Popop a dialog box and ask fo content
Input	None
Output	Content

Name	OnAddAttribute
Description	Triggered when clicking on Add attribute
Input	None
Output	None

Name	OnDeleteAttribute
Description	Triggered when clicking on Delete attribute
Input	None
Output	None

Name	OnModifyAttribute
Description	Triggered when clicking on Modify attribute
Input	None
Output	None

Name	OnDbClickObject
Description	Triggered when double clicking on an object
Input	None
Output	None

Name	OnCreateObject
Description	Triggered when clicking on Create object

Input	None
Output	None

Name	OnAddObject
Description	Triggered when clicking on Add object
Input	None
Output	None

Name	SelectObject
Description	Popup a dialog box and ask for selecting an object
Input	None
Output	Object

Name	OnDeleteObject
Description	Triggered when clicking on Delete object
Input	None
Output	None

Name	OnShowLibraries
Description	Triggered when clicking on Show Libraries
Input	None
Output	None

Name	OnAddLibrary
Description	Triggered when clicking on Add Library
Input	None
Output	None

Name	GetLibraryName
Description	Popup a dialog box and ask for library name
Input	None
Output	Library Name

Name	OnAddRelationshipToLibrary
Description	Triggered when clicking on Add Relationship to Library
Input	Relationship
Output	None

Name	OnLink
Description	Triggered when dragging the mouse between two objects
Input	None
Output	None

Name	OnDefineRelationship
Description	Triggered when clicking on Define Relationship
Input	None
Output	None

Name	OnModifyRelationship
Description	Triggered when clicking on Modify Relationship
Input	None
Output	None

Name	OnFindRelationship
Description	Triggered when clicking on Find Relationship
Input	None
Output	None

Name	OnOpenFile
Description	Triggered when clicking on Open File
Input	None
Output	None

Name	OnSaveFile
Description	Triggered when clicking on Save File
Input	None
Output	None

Name	OnPrintAlphabeticalReport
Description	Triggered when clicking on Print Alphabetical Report
Input	None
Output	None

Name	OnPrintRelationalReport
Description	Triggered when clicking on Print Relational Report
Input	None
Output	None

Name	Undo
Description	Triggered when clicking on Undo
Input	None
Output	None

Name	OnHistory
Description	Triggered when clicking on Show History
Input	None
Output	None

Name	OnLog
Description	Triggered when clicking on Login the system
Input	None
Output	None

Name	OnRunSimulation
Description	Triggered when clicking on Run Simulation
Input	None
Output	None

Class: Display

Attributes:

Name	Type	Description

Methods:

Name	LinkObjects
Description	Display a line linking two objects
Input	Two objects to link
Output	None

Category: FILE_MANAGER_CAT

Class: File_Manager

Attributes:

Name	Type	Description
M_listOfFile	List of File	Contains a list of files

Methods:

Name	OpenFile
Description	Open a file specified by its location(directory) and name.
Input	String containing file name String containing file directory
Output	Error message if file doesn't exist

Name	SaveFile
Description	Save the current opened file, save all the objects and relationships defined in the file.
Input	none
Output	Error message if the disk is full

Name	Undo
Description	Restore the system to its previous state
Input	none
Output	none

Name	GetHistory
Description	Retrieve the sets of action the user perform on each file.
Input	none
Output	none

Class: File

Attributes:

Name	Type	Description
Name	String	Name of the file.
Directory	String	Location of the file.
m_history	List of modification	List of all modifications performed on the file since its creation.
m_session	List of modification	List of modification performed on a file since the beginning of the current session.
m_access	boolean	=1, if the file is not opened by any other user. =0, if the file is opened by another user.

Methods:

Name	ReadFile
Description	Open a file with read access
Input	String containing file name String containing file directory
Output	Error message if the file does not exist

Name	WriteFile
Description	Open a file with write access
Input	String containing file name String containing file directory
Output	Error message if the file does not exist

Name	GetFileAccess
Description	Retrieve the type of access the user can open the fil
Input	none
Output	Return 0 if only read Return 1 if read/write

Name	GetLastSessionModification
Description	Retrieve the last modification of the list of modification performed in the session (from the list m_session)
Input	none
Output	The last modification of the list m_session

Name	GetHistory
Description	Retrieve the list of modification performed on the file since it creation (the list m_history)
Input	none
Output	The list of modification m_history

Class: Modification

Attributes:

Name	Type	Description
m_user	String	Name of the user who performed the modification
m_date	String	Date when the modification occured
m_time	String	Time when the modification occured
comments	String	Modification details

Methods:

Name	Init
Description	Initialize a new modification
Input	Name of the user who performed the modification Date when the modification occurred Time when the modification occurred Modification details
Output	none

Name	GetUser
Description	Retrieve the name of the user who performed the modification
Input	none
Output	Name of the user who performed the modification

Name	GetDate
Description	Retrieve the date when the modification occurred
Input	none
Output	Date when the modification occurred

Name	GetTime
Description	Retrieve the time when the modification occurred
Input	none
Output	Time when the modification occurred

Name	GetComments
Description	Retrieve the Modification details
Input	none
Output	Modification details

Category: REPORT_MANAGER_CAT

Class: Report_Manager

Attributes:

Name	Type	Description

Methods:

Name	PrintAlphabeticalReport ( )
Description	This method print in an alphabetical order a report related to the name of the objects, the relationship between the objects as it appear in the file related to the project
Input	File containing the project relationship information
Output	Printed copy

Name	PrintRelationalReport ( )
Description	This method print in the report related to the relationship between the objects as it appear in the file related to the project
Input	File containing the project relationship information
Output	Printed copy

Name	
Description	
Input	
Output	

Class: Report

Attributes:

Name	Type	Description
m_file	FILE	Pointer to file

Methods:

Name	Print
Description	This method activate the print job in either one of the sub classes listed below
Input	File content
Output	Printed copy

Name	ReadFile
Description	This method read entirely the file pointed to by the file pointer
Input	File name
Output	File content in a data structure

Class: Report_Alphabetical

Attributes:

Name	Type	Description

Methods:

Name	Print
Description	This method activate the print the content of the file system in alphabetical order. Print the names, relationship and formula relating the object
Input	File content
Output	Printed copy

Class: Report_Relational

Attributes:

Name	Type	Description

Methods:

Name	Print
Description	This method print the content of the file system without any order. Print the names, relationship and formula relating the objects.
Input	File content
Output	Printed copy

Category: SIMULATION_MANAGER_CAT

Class: Simulation

Attributes:

Name	Type	Description
M_ProjectManager	ProjectManager	Pointer to the class project manager

Methods:

Name	CheckUnits
Description	Check the consistency of the units that would be used in the relationship
Input	Objects relationship formula
Output	Boolean

Name	CheckOverallConsistency
Description	Check the consistency of the relationship
Input	Links of two objects
Output	Boolean

Name	Compute Values
Description	Apply the relationship to compute certain values of the relationship
Input	Links of two objects
Output	Result of the computation

Name	Run
Description	Run in order the simulation for consistency. Execute in order CheckOverallConsistency ( ) , CheckUnits ( )
Input	Links of two objects
Output	void

TESTING & TRACEABILITY

The team developer made sure to test each single component of the analysis. The developer team made sure to apply traceability analysis .
To apply tracability the developers made sure that :

- Each shall statement will be captured in a RTM
- Each shall statement has a Use case associated with it.
- Each use case has a category associated to it.
- Each use case has a scenario associated with it.
- Each use case has its own Category Interaction Diagram and its own Interaction Diagram.
- Each Category figure in the System Category Diagram,
- Each category is composed of classes
- Each category has its own Category Class Diagram .
- Each Class has its own class speciafication.

Individual Task Planning Template - Form TASK1

Name Francoise Boudigou Team 4 Instructor Dr. Gutcher
Date 04-10-2001 Cycle No.

Task		Plan					*Actual*			
#	Name	Hrs	PV	Cum. Hrs	Cum. PV	Date Mon.	Date	Hrs	EV	Cum. EV
1	**PLAN PHASE 1 Requirement Engineering**	1.5	1.5	1.5	1.5	2/5	2/5	1.5	2	2
2	Context Diagram	2	2	3.5	3.5	2/5	2/5	3	4.2	6.2
3	Capture the shall statements	2	2	5.5	5.5	2/5	2/5	.5	0.7	6.9
4	Allocate and Prioritize requirements	2	2	7.5	7.5	2/5	2/5	1	1.4	8.3
5	PLAN PHASE 2 System Object Oriented Analysis	5	5.1	12.5	12.6	2/12	2/12	4	5.5	13.8
6	Identify Use Cases	6	6.1	18.5	18.7	2/12	2/12	2	2.7	16.5
7	Develop Scenarios	13	13.4	31.5	32.1	2/12	2/12	8	11	27.5
8	Develop draft GUI	7	7.2	38.5	39.3	2/19	2/19	5	6.8	34.3
9	Establish project categories	10	10.3	48.5	49.6	2/19	2/19	9	12.35	46.65
10	Allocate use cases to categories	2	2	50.5	51.6	2/19	2/19	2	2.7	49.35
11	Develop System Category Diagram (SCD)	3	3	53.5	54.6	2/19	2/19	2	2.7	52.05
12	**PLAN PHASE 3 System Object Oriented Analysis**	1.5	1.5	55	56.1	2/19	2/19	2	2.7	54.75
13	Allocate Category to category manager/Leads	7	7.1	62	63.2	2/26	2/26	9	12.35	67.1
14	Develop Category Interaction Diagram (CID)	2	2	64	65.2	2/26	2/26	2	2.7	69.8
15	**PLAN PHASE 4&5: Software Object Oriented Analysis**	10	10.3	74	75.5	2/26	2/26	3	4.2	74
16	1 – Initiate Category Class Diagram	5	5.1	79	80.6	2/26	2/26	2	2.7	76.7
17	2 – Refine Inheritance, Aggregation, hierarchies	8	8.2	87	88.8	2/26	2/26	8	11	87.7
18	3 – Update Category Class Diagram	3	3	90	91.8	2/26	2/26	4	5.5	93.2
19	4 – Complete Class Specification (CCS)	8	8.2	98	100	03/05	03/05	5	6.8	100
	total	98	100					73		

Individual Task Planning Template - Form TASK1

Name Chokri Oueslati Team 4 Instructor Mr Gutcher
Date 04-10-2001 Cycle No. *

Task		Plan					*Actual*			
#	Name	Hrs	PV	Cum. Hrs	Cum. PV	Date Mon.	Date	Hrs	EV	Cum. EV
1	**PLAN PHASE 1 Requirement Engineering**	1.15	1.17%	1.15	1.17%	03/19	03/17	1	1.17%	1.17%
2	Context Diagram	2	2.04%	3.15	3.21%	03/19	03/17	3	2.04%	3.21%
3	Capture the shall statements	2	2.04%	5.15	5.25%	03/19	03/19	1	2.04%	5.25%
4	Allocate and Prioritize requirements	2	2.04%	7.15	7.28%	03/19	03/19	3	2.04%	7.28%
5	PLAN PHASE 2 System Object Oriented Analysis	5	5.09%	12.15	12.38%	03/26	03/20	2	5.09%	12.38%
6	Identify Use Cases	6	6.11%	18.15	18.49%	03/26	03/22	4	6.11%	18.49%
7	Develop Scenarios	13	13.25%	31.15	31.74%	03/26	03/25	9	13.25%	31.74%
8	Develop draft GUI	7	7.13%	38.15	38.87%	03/26	03/26	3	7.13%	38.87%
9	Establish project categories	10	10.19%	48.15	49.06%	03/26	03/26	2	10.19%	49.06%
10	Allocate use cases to categories	2	2.04%	50.15	51.10%	03/26	03/26	1	2.04%	51.10%
11	Develop System Category Diagram (SCD)	3	3.06%	53.15	54.15%	03/26	03/26	9	3.06%	54.15%
12	**PLAN PHASE 3 System Object Oriented Analysis**	1	1.02%	54.15	55.17%	04/03	03/28	2	1.02%	55.17%
13	Allocate Category to category manager/Leads	7	7.13%	61.15	62.30%	04/03	03/29	9	7.13%	62.30%
14	Develop Category Interaction Diagram (CID)	2	2.04%	63.15	64.34%	04/03	04/03	2	2.04%	64.34%
15	**PLAN PHASE 4&5: Software Object Oriented Analysis**	4	4.08%	67.15	68.42%	04/10	04/04	1	4.08%	68.42%
16	1 – Initiate Category Class Diagram	5	5.09%	72.15	73.51%	04/10	04/10	3	5.09%	73.51%
17	2 – Refine Inheritance, Aggregation, hierarchies	8	8.15%	80.15	81.66%	04/10	04/10	6	8.15%	81.66%
18	3 – Update Category Class Diagram	3	3.06%	83.15	84.72%	04/10	04/10	5	3.06%	84.72%
19	4 – Complete Class Specification (CCS)	15	15.28%	98.15	100.00	04/10	04/10	4	15.28%	100.00
total		**98.1**	**100**							

Individual Task Planning Template - Form TASK1

Name Thomas Pombourg Team 4 Instructor Mr Gutcher
Date 04-10-2001 Cycle No. 2

Task		Plan					*Actual*			
#	Name	Hrs	PV	Cum. Hrs	Cum. PV	Date Mon.	Date	Hrs	EV	Cum. EV
1	**PLAN PHASE 1 Requirement Engineering**	1	1.47	1	1.47	2/5	2/5	1.25	1.47	1.47
2	Context Diagram	2	2.94	3	4.41	2/5	2/5	1	2.94	4.41
3	Capture the shall statements	2	2.94	5	7.35	2/5	2/5	1	2.94	7.35
4	Allocate and Prioritize requirements	2	2.94	7	10.29	2/5	2/5	1	2.94	10.29
5	PLAN PHASE 2 System Object Oriented Analysis	5	7.35	12	17.64	2/12	2/12	3	7.35	17.64
6	Identify Use Cases	4	5.88	16	23.52	2/12	2/12	4	5.88	23.52
7	Develop Scenarios	5	7.35	21	30.88	2/12	2/12	6	7.35	30.88
8	Develop draft GUI	4	5.88	25	36.76	2/19	2/19	6	5.88	36.76
9	Establish project categories	6	8.82	31	45.58	2/19	2/19	7	8.82	45.58
10	Allocate use cases to categories	2	2.94	33	48.52	2/19	2/19	2	2.94	48.52
11	Develop System Category Diagram (SCD)	3	4.41	36	52.94	2/19	2/19	3	4.41	52.94
12	**PLAN PHASE 3 System Object Oriented Analysis**	1	1.47	37	54.41	2/19	2/19	1	1.47	54.41
13	Allocate Category to category manager/Leads	5	7.35	42	61.76	2/26	2/26	5	7.35	61.76
14	Develop Category Interaction Diagram (CID)	2	2.94	44	64.7	2/26	2/26	2	2.94	64.7
15	**PLAN PHASE 4&5: Software Object Oriented Analysis**	2	2.94	46	67.64	2/26	2/26	4	2.94	67.64
16	1 – Initiate Category Class Diagram	3	4.41	49	72.05	2/26	2/26	4	4.41	72.05
17	2 – Refine Inheritance, Aggregation, hierarchies	6	8.82	55	80.88	2/26	2/26	7	8.82	80.88
18	3 – Update Category Class Diagram	3	4.41	58	85.29	2/26	2/26	4	4.41	85.29
19	4 – Complete Class Specification (CCS)	10	14.7	68	100	03/05	03/05	10	14.7	100
	Total	98.25		68				86.25		100
total										

Appendix B : Team Meeting Report

Meeting Report Form – Form MRF

Project Name:	Assignement 3	**Semester:**	Spring 2001
Meeting Location:	Computer Lab	**Date:**	February 27, 2001
Start Time:	2:00 p.m.	**End Time:**	3:00 p.m.

Attendees:	Role:
Chokri Oueslati	Team Leader
Thomas Pombourg	Documentation Manager
Francoise Boudigou	Record coordinator

Comments:	Launching phase
Brief Description of Meeting:	
	Decided basic strategy
	Defined basic planning
Action Items Given:	
	For all : Strategy and scope to refine.
	For all: Complete planning template.
Action Items Returned:	N/A
Next Meeting:	Saturday, March 10, 2001 at 2:00 p.m. in Computer Lab

Meeting Report Form – Form MRF

Project Name:	Assignement 3	**Semester:**	Spring 2001
Meeting Location:	Computer lab	**Date:**	March 10, 2001
Start Time:	2:00 p.m.	**End Time:**	3:00 p.m.

Attendees:	Role:
Chokri Oueslati	Team Leader
Thomas Pombourg	Documentation Manager
Francoise Boudigou	Record coordinator

Comments:	Elicitation and analysis phase.
Brief Description of Meeting:	Review of the Interview with customer.
	Divide tasks amongst members to complete Elicitation and Analysis
	templates
	Context Diagram
	USE CASES
	Categories
Action Items Given:	All : Identify Possible Category
	Refine USE CASES and establish Categories
Action Items Returned:	
Next Meeting:	Saturday, March 31, 2001 at 2:00 p.m. in Computer Lab

Meeting Report Form – Form MRF

Project Name:	Assignement 3	**Semester:**	Spring 2001
Meeting Location:	Computer lab	**Date:**	March 31, 2001
Start Time:	2:00 p.m.	**End Time:**	3:00 p.m.

Attendees:	Role:
Chokri Oueslati	Team Leader
Thomas Pombourg	Documentation Manager
Francoise Boudigou	Record coordinator

Comments:	.
Brief Description of Meeting:	USE CASES
	System Category Diagram
Action Items Given:	
	Chokri: Activity Diagram
	Thomas: Category Interaction Diagrams
	Thomas: Category Class Diagram
	Francoise: Class Specifications
Action Items Returned:	
Next Meeting:	Saturday, April 7, 2001 at 2:00 p.m. in Computer Lab

Meeting Report Form – Form MRF

Project Name:	Assignement 3	**Semester:**	Spring 2001
Meeting Location:	Computer lab	**Date:**	April 7, 2001
Start Time:	2:00 p.m.	**End Time:**	3:00 p.m.

Attendees:	Role:
Chokri Oueslati	Team Leader
Thomas Pombourg	Documentation Manager
Francoise Boudigou	Record coordinator

Comments:	Final meeting
Brief Description of Meeting:	For all: refine Top level Context Diagram and Use cases
	Activity Diagram
	Category Interaction Diagrams
	Category Class Diagram
	Class Specifications
	State Transition Diagram
Action Items Given:	
Action Items Returned:	Activity Diagram
	Category Interaction Diagrams
	Category Class Diagram
	Class Specifications State Transition Diagram
Next Meeting:	