



Appendix E
Campus-Based Modernization
Use Case Realization Report Overview
May 2001



Table of Contents

1. OVERVIEW	3
2. HOW TO VIEW THIS APPENDIX	3
3. USING THE INFORMATION ON THE CD	3
<i>Use Case View</i>	3
<i>Logical View</i>	3
<i>Component View</i>	3
4. TERMINOLOGY	3



1. Overview

Rational Rose, part of the Rational Suite Development Studio, is a visual modeling tool based on the Unified Modeling Language (UML) standard notation for software design. The eCampus Based project is using Rational Rose to help improve communication, manage complexity, analyze requirements, define the software architecture, enable reuse when possible, and capture vital business processes. Appendix E: Use Case Realization Report is a CD that allows viewing of the modeling for this project.

2. How to View this Appendix

Option 1: View the information directly from the CD:

Step 1- Place the CD into your CD drive.

Step 2- Open 'Windows Explorer' (located on your desktop).

Step 3- Double-click on 'My Computer.'

Step 4- Double-click on the 'Compact Disk' drive.

Step 5- Double Click on eCampusBasedDesignIteration1(May11).htm. The Appendix will open with a Navigation Menu on the left.

*Note - If you follow these steps and are not able to view the navigation bar, proceed to Option 2.

Option 2: Copy the information on the CD to your local Hard Drive:

Step 1- Place the CD into your CD drive.

Step 2- Open 'Windows Explorer' (located on your desktop).

Step 3- Double-click on 'My Computer.'

Step 4- Double-click on the 'Compact Disk' drive.

Step 5- Select all of the files. (Click 'Edit' in the top menu bar. Click on 'Select All.')

Step 6- Copy all of the files. (Click 'Edit' in the top menu bar. Click on 'Copy.')

Step 7- Paste the files to your local Hard Drive. (Decide where you would like to paste the files. Click on 'Edit' in the top menu bar. Click on 'Paste.')

Step 8- Double Click on eCampusBasedDesignIteration1(May11).htm. The Appendix will open with a Navigation Menu on the left.



3. Using the information on the CD

The information contained in Appendix E is organized through the Navigation Menu. To view the information contained in each folder, click on the '+' button. This will expand the section and provide access to the files.

Use Case View

This folder contains user case folders for particular iterations of the system along with an "actors" folder. Iteration 1 Use Cases and Use Case diagrams are located in the "FISAP ENTRY Use Cases" and "FISAP Entry Use Case Diagrams" folders. There are two ways to refer to the 12 Use Cases included in Deliverable 70.1.2 – Iteration I Detailed Design Document:

1. Refer to Appendix D: Use Case Specifications for hard copies of the Use Case documents.
2. Browse to the following path on the CD - documents\racerock\rational projects\cbp\cbp. Electronic copies of the Use Cases are available in this folder.

Definitions of a Use Case, a Use Case diagram and an actor are provided below:

Use Case - used in UML to analyze system requirements and to define how the user accesses the features of the system. The information available in a Use Case may include:

- Use Case (function) name and description
- Flow of events, including the basic flow and all alternative flows
- Special requirements, when needed
- Pre-conditions required for the function to occur
- Post-conditions that exist after the function occurs
- Related Use Cases (called "extension points") that describe a variation of the normal behavior of the function

Use Case Diagram - graphically depicts system behavior (Use Cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective.

Actors - represent system users. Actors help delimit the system and provide an illustration of what the system should do. An actor is someone or something that either interacts with the system, provides input to the system, receives information from the system, and is external to the system and has no control over the Use Cases.

Logical View

This view contains the various folders that hold the actual elements of the design. This includes classes, packages and relationships. The following items are available in this view:

Analysis Model - The analysis model leads to the static class design. The object developed in this model translate into web pages, classes that reside on the server, and the data base interface.

Static Design Model - each package in the design will contain one or more classes and is the package that the code is developed from. This model is comprised mostly of the client and server pages associated with Iteration 1.



Design Model - within this model Use Case realizations are found for each part of the FISAP delivered within Iteration 1. Each Use Case realizations will contain participating class diagrams and where the design required, sequence and/or collaboration diagrams. The class diagrams capture the structure of the classes that form the system's architecture. The sequence diagrams are graphical views of a scenario that show object interaction in a time-based sequence. They primarily establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

Package "fisapservlets" - contains the related classes that control the user's session.

Package "fisapbeans" - contains the related classes that comprise the Business logic and database access for Iteration 1.

Package "fisaputils" - contains the common utilities used within Iteration 1.

Package "log" - contains the common logging utilities used within Iteration 1.

Component View

Component packages represent clusters of logically related components, or major pieces of the system. Component packages parallel the role played by logical packages for class diagrams.

4. Terminology

The following terms are used with Appendix E: Use Case Realization Report.

Class

A class is a set of objects that share a common structure and common behavior (such as attributes, operations, relationships and semantics). An instance of a class is referred to as an object. A class icon is drawn as a 3-part box, with the class name in the top part, a list of attributes in the middle, and a list of operations in the bottom part. A class diagram describes the types of objects in the system and the kinds of static relationships that exist among the objects.

Stereotype

A stereotype represents a class within the UML metamodel (a type of modeling element). Some stereotypes are already predefined in Rational Rose, but you can also define your own to add new kinds of modeling types. The stereotypes used in this document are predefined in Rational Rose:

- **Boundary** - A boundary class represents an interface between the system and some entity outside the system: a person or another system. Its role is to mediate the exchange of information with the outside world, and to insulate the system from changes in its surroundings.
- **Control** - A control class is a class used to model control behavior specific to one or a few Use Cases. Control objects (instances of control classes) often control other objects, so their behavior is of the coordinating type. Control classes encapsulate use-case specific behavior.
- **Entity** - An entity class is a class used to model information and associated behavior that must be stored. Entity objects (instances of entity classes) are used to hold and update information about some phenomenon, such as an event, a person, or some real-life object. They are usually

persistent, having attributes and relationships needed for a long period, sometimes for the life of the system.

Attribute Stereotype

An attribute stereotype defines the characteristics of a data element within a Boundary class. Each object in a class has the same attributes, but the values and characteristics of the attributes may be different. Examples of some of the attribute stereotypes used in this project are text boxes, combo boxes and check boxes.

Use Case Realization

A Use Case realization is a graphic sequence of events, also referred to as a scenario or an instance of a Use Case. These realizations or scenarios are depicted in either sequence, collaboration diagrams and most often participating class diagrams. The two most common scenarios are the typical scenario where all is well and the exception scenario which addresses deviations.

Actor

Actors represent system users. They help define the system and give a clearer picture of what the system should do. An actor is someone or something that:

- Interacts with or uses the system
- Provides input to and receives information from the system
- Is external to the system and has no control over the Use Cases

Package

A package is a model element that organizes a logical group of classes.

UML Notation

The following legend defines the UML notation used throughout this document

