Software Architecture Document

in fulfillment of Soen 344 Winter 2009 – Ver. 1.0

Date: 3/23/2009

Team X

Date	Rev.	Description	Author(s)	Contributor(s)

Concordia University Montreal

Winter 2009

Table of Contents

1. Introduction

1.1. Purpose of the Document

This document provides the architectural outline of the IEEE Montreal Web Portal system. Different architectural views are used to illustrate different aspects of the system. This document also presents the significant architectural decisions that are made on the system.

1.2. Scope

The scope of the document is to describe the architectural goals and constraints, the Use Case View, the Logical View, the Process View, the Deployment View and, the Implementation View, based on the "4+1 View Model" and the provided template.

1.3. Definitions, Acronyms, and Abbreviations

- I3EM – IEEE Montreal
- Architecture View - "A view of the system architecture from a given perspective; it focuses primarily on structure, modularity, essential components, and the main control flows." (Larman, pg. 656)
- UML - Unified Modeling Language
- WEAA – Web Enterprise Software Architecture Patterns
- SOEN 343 - Software Design course
- SOEN 344 - Software Architecture course
- AJAX - Asynchronous JavaScript and XML
- JSON - JavaScript Object Notation
- GWT – Google Web Toolkit

1.4. References

1. Fowler, Martin. *Patterns of Enterprise Application Architecture*. Boston: Addison-Wesley, 2003.
2. Larman, Craig. *Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd ed. Upper Saddle River: Prentice Hall PTR, 2005.
3. SOEN 343, 344, and 390 websites
4. Kruchten, Philippe (1995, November). Architectural Blueprints — The "4+1" View Model of Software Architecture. IEEE Software 12 (6), pp. 42-50

2. Architectural Representation

The architecture of I3EM is illustrated using the views defined in the "4+1" model [4], but using the RUP naming convention. The views used to document the I3EM application are:

Use Case view

Audience: all stakeholders
Area: describes the set of scenarios and/or use cases that are critical to the architecture
Related Artifacts: Use Case Document

Logical view

Audience: I3EM designers
Area: Functional Requirements, to assess functionality
Related Artifacts: Package Diagram

Process view

Audience: Integrators.
Area: Non-functional requirements: describes the design's concurrency and synchronization aspects. How logical view components are associated to processes.
Related Artifacts: Process View Diagram

Implementation view

Audience: Programmers.
Area: Software components: describes the structural organization of object files, libraries, test codes, etc.
Related Artifacts: Directory hierarchy structure layout

Deployment view

Audience: Deployment managers.
Area: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects.
Related Artifacts: UML Deployment Diagram

3. Architectural Goals and Constraints

The architectural choices where made considering the following goals and constraints.

- The focus of SOEN 343 and SOEN 344 are on Web Enterprise Application patterns (i.e., Front Controller, Front Command, Data Mapper, TDG, Identity Map) and Architectural Views (i.e. 4+1 View). The I3EM is designed based on what is learned in the SOEN courses.
- User authentication will always take place before any data access in I3EM. Complete security of data from unauthorized access will be enforced.
- All functional and non-functional requirements will be maintained as the architecture is being developed as specified in the Use Case Model .

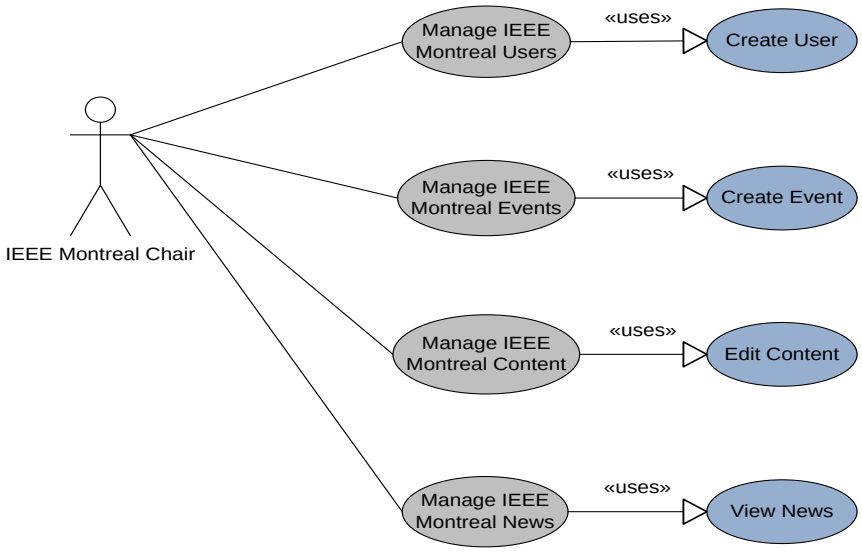
4. Use-Case View

"The use case view describes a view of the system's architecture that covers the general system behavior from the point of view of the involved stakeholders." This view exposes the set of use cases that defines the core functionality of the I3EM Portal.

4.1. Architecturally significant use cases

The following are the set of use cases that are architecturally significant to the I3EM system:

- Create User
- Create Event
- Edit Content
- View News



4.2. Brief Description of Use Cases

- **Create User: UC 4.1**
  The logged in user can create new users for the system by logging in and selecting the "Add New User" link. Then s/he enters the requested information for the user; username, password, email address, first name, last name, and role. If any of the required information is not entered properly, system will display a warning and does not create the account.

- **Create Event: UC 4.15**
  A privileged user can Login and choose "Events Management" section and input field values and choose presented options then click on "New Event" to create a new event entry . Attachment and Invitation can also be attached.

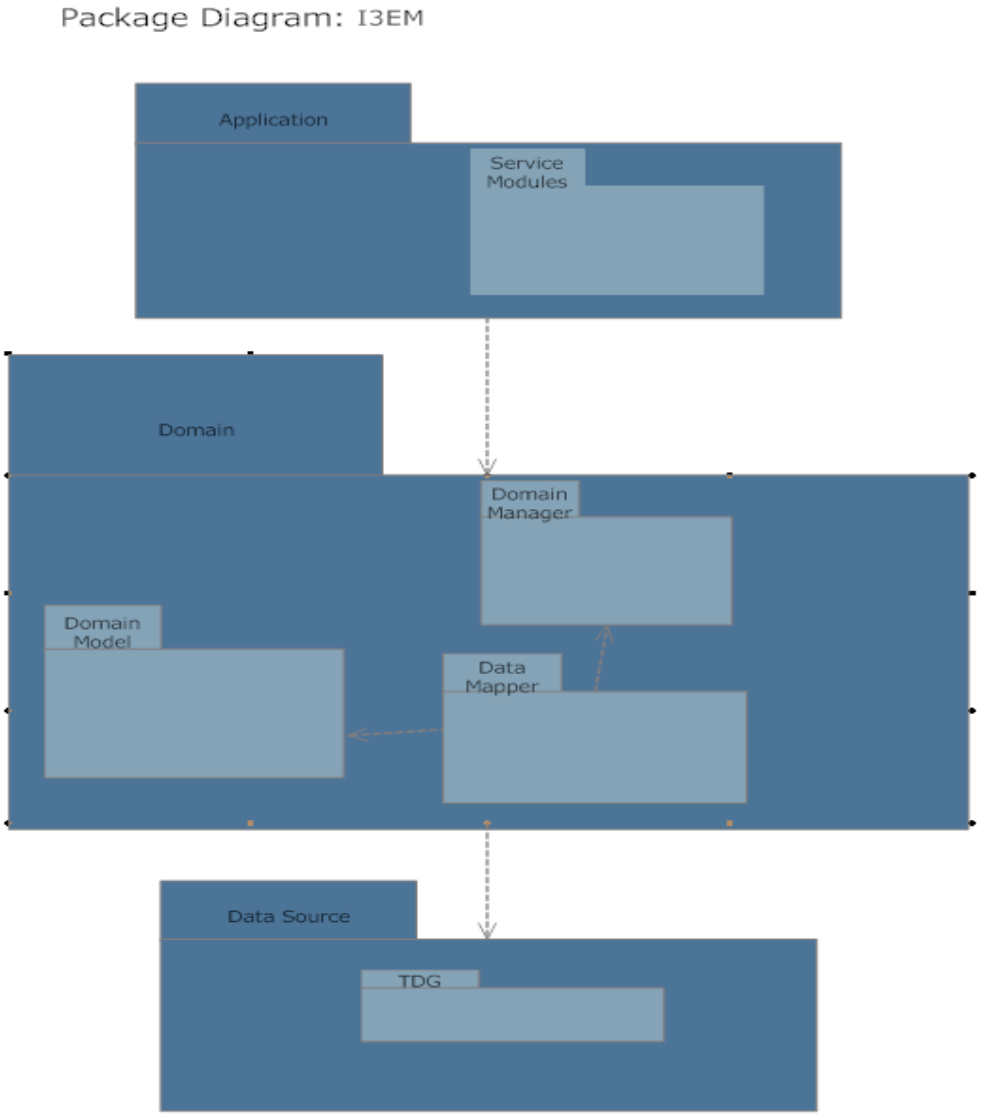
- **Edit Content: UC 4.33**
  A privileged user can Login and choose "Edit Content" section and edit field values s/he would like to change, then click on "Save Content" to update the new content.

- **View News: UC 4.19**
  Any user can view an existing event in the system by simple navigating to the "News Section" on the main menu bar.

5. Logical View

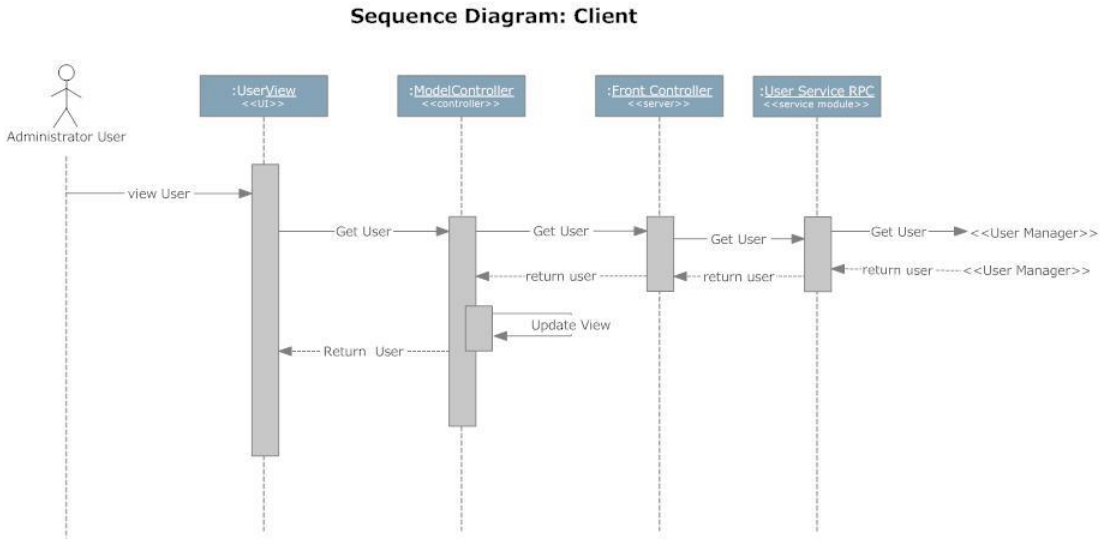
"The conceptual organization of the system is presented in the Logical view in terms of layers, packages, subsystems, and classes."

The logical view of the I3EM system is separated into 3 main Layers: the application layer, the domain layer and the data source layer. These 3 main Layers are structured following a 3-layer architecture convention: the presentation layer, the domain layer and the data source layer.



5.1. Application Layer

The application Layer in this implementation modifies the traditional Front Controller with the template view organization with Ajax components, developed with the Google Web Toolkit. The front Controller now handles request by referencing domain objects to and from the client to the server. The benefits of this approach allow a more robust and richer GUI implementation as the application can behave with characteristics of a native desktop application, additionally as most of the data gets cached locally on the first access subsequent access result in less stress to the server and more lively experience for the end user.



5.2. Domain Layer

The domain package encloses sub-packages used to describe the I3EM system's business logic rules that control and govern the way information is processed. The sub-packages are the domain model, the data mapper, the unit of work, the identity map, the virtual proxy layer.

The domain model sub-package contains a class for each of the major business entities: users, roles, events, news and content. It also contains a Layer Supertype which is DomainObject to make use of the generic Unit of work and Identity map.

These class objects are used by manager transactions and data mapper objects as data containers. Information about an entity is stored as attributes of the corresponding class. The information about an entity is passed through methods by passing the corresponding domain object as a parameter.

The data mapper sub-package contains a data mapper class for each domain object class located in the domain object package. This layer also contains a Layer Supertype which is AbstractMapper to make use of the advantage of reflection pattern used by the unit of work. These data mappers create a layer of indirection between the domain logic and the data source. The transactions do not know about the structure of the datasource as well as the different queries to access the information.

The unit of work is used in the application of the unit of work pattern as described in section 10.1.The identity map is used in the application of the identity map pattern as described in section 10.2.The virtual proxy layer contains a set of proxy classes and loader classes which are used in the application of the lazy load pattern as described in section 10.3.

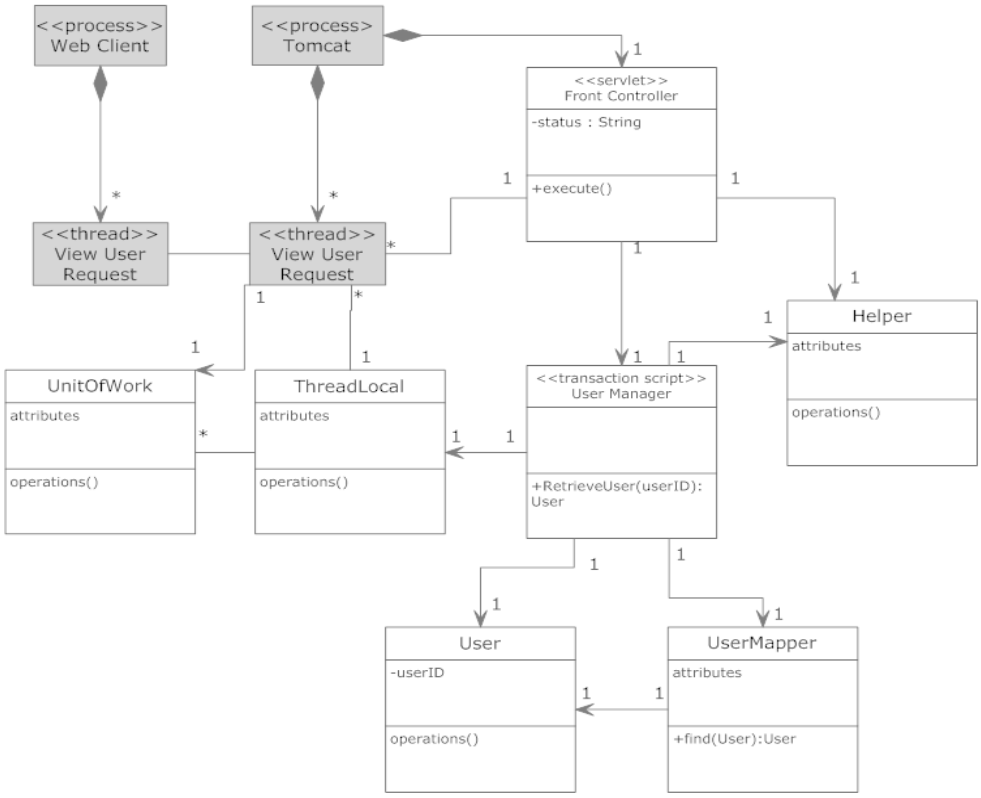
5.3. Data Source Layer

The data source package contains the classes that define the technical services infrastructure used to store persistent data. This package is used by the I3EM application to interface with the database located on the Stu03 server. The package includes a class source file for each class present in the data mapper sub-package. These data source objects (**Table Data Gateways**) are used by data mapper objects to access the database directly to store or retrieve information.

6. Process View

"*The Process view displays the responsibilities and collaborations among processes and threads of the system*", as well as the allocation of logical elements to them.

In this section, we will describe the tasks, which include processes and threads that are involved in the execution and interactions of the system. The following process model expresses the View User request organized as executable processes and threads.



The diagram illustrates a request that is initialized by the authorized user to gain details of specific user account in the I3EM system. The user launches a web browser located on a remote computer which creates the web client process. Once the user has authenticated his or herself, the user's main page appears. By selecting the view user option, the web client process creates and sends a request to the tomcat server process which was and remains online and available. This request thread remains in memory until a result is received from the tomcat server. The request thread is then passed on to the appropriate service module for further processing. The retrieve User method in User Manager takes the userID as the input and interacts with the UserMapper which in turns interacts with the datasource layer. Subsequently, it returns the User objects to the client controller in turns produces the updated view.

7. Deployment View

"The Deployment view shows the allocation of the Logical view elements to physical processing nodes, and the physical network configuration between nodes."

The I3EM system is designed as a web application and is meant to be used as a client/server. The I3EM web application is to be deployed on **Stu03** Concordia web server located at the following address: http://group0i.stu03.encs.concordia.ca/servlet/HomePage.html. The application is to be migrated to a similar setup on the IEEE servers.

In the following UML deployment diagram, the physical network elements involved in the deployment of the I3EM system is presented. In the next following section, a description of each network element is described.

7.1. Client Application Web Browser

Typical users utilize a client web browser, such as Internet Explorer running on to access the IEEE Montreal Web Portal. Communication between the client and the server is ensured by the HTTP network protocol. The user will use a graphical interface presented as a web page through which he/she can select the desired commands.

7.2. Web server

The I3EM web application is currently installed and operates from the **Stu03** server. This server remains under the control of Concordia University's Software Engineering department and is integrated to the ENCS computer lab network.

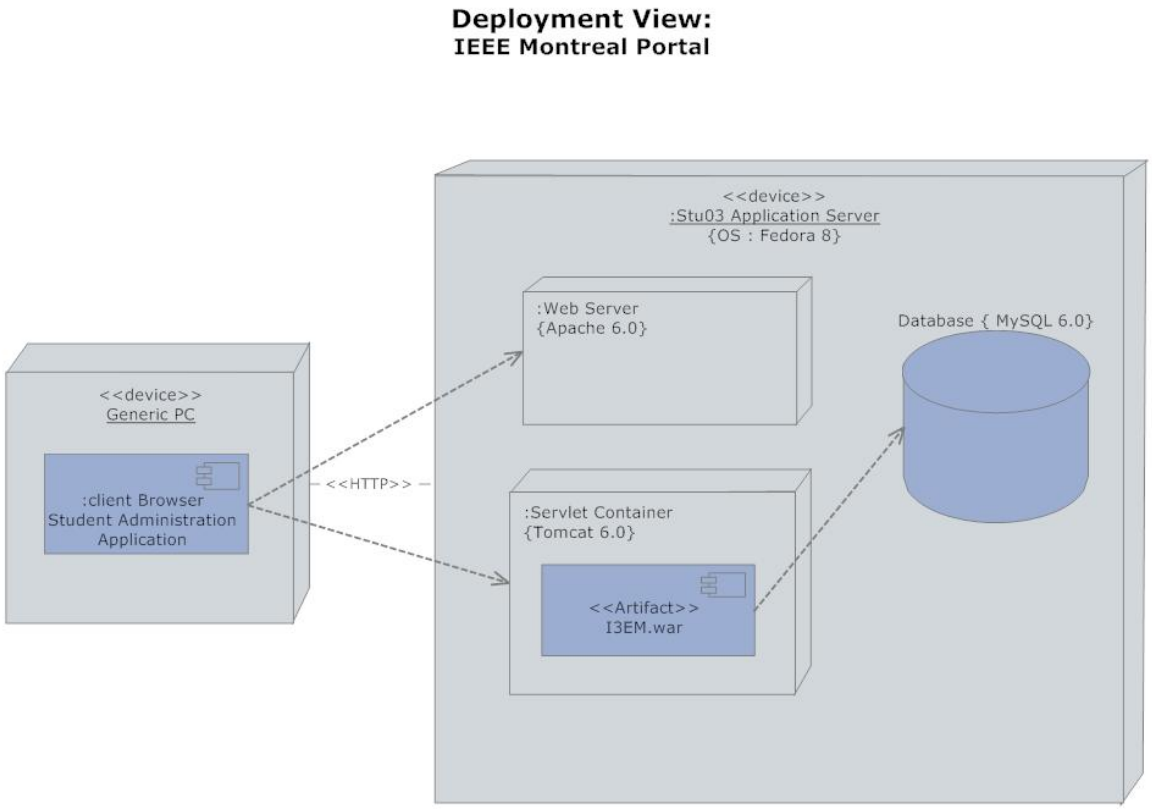
The web server is maintained by the Software Engineering department and its availability may be subject to scheduled (or unscheduled) ENCS network maintenance work. Deployment of the web application must be performed by SSH tunneling in order to respect Concordia University security restrictions. A client application, such as putty, can be used to connect to the **Stu03** server and perform the necessary deployment actions (I&C documentation).

Other security restrictions, such as the following, may apply during the deployment phase:

- Database connections are restricted locally (i.e. must connect to localhost)
- Servlets files reads/writes are restricted to the Stu03 file permission policies
- Database tables creation are restricted to the SQL script files deployed

7.3. Database

The database utilized by the I3EM web application is MySQL 5. The database server is located in the **Stu03** server over a Linux based operating system. Any database table's creation is restricted by the SQL script files available in the SQL. No user can create new tables through the I3EM system. However, anyone with access rights to the Stu03 server may edit the current SQL script files to add new tables, create new SQL script files or simply access the MySQL database through a command prompt.



8. Implementation View

8.1. Structural Organization of the files

The I3EM system files are divided into four main folders located under the same root folder. The root folder is named I3EM. All java source files are located under the src folder. All SQL scripts files are located under the SQL folder. Finally, all java test files are located under the test folder.

8.2. Source Directory Structure

The I3EM application source files are divided into a folder scheme that at the root separates the server and client files. After this components are further segregated in appropriate folders. A detailed listing is available in 8.5.

8.3. Test File Organization

Three main types of testing are scheduled to be performed: system-level testing, functional testing and unit testing.

The tests files are grouped together by component from independent from each other. The tests can be run by executing the corresponding test file, for example EventsManagerJUNIT.java for all tests related to events management.

8.4. SQL files

The SQL folders contain one SQL script files “init.sql” it is used to create the set of database tables used by the I3EM system. The table is not empty and filled with some meaningful information use for testing.

8.5. Detailed Overview

The following files directory structure describes the details of the hierarchy used to organize the java source files, class files, web pages.

Main Package	Comments
Soen390_IEEE_M6	Root
Soen390_IEEE_M6\SQL	SQL files for databases are here
Soen390_IEEE_M6\server\datasource	Table data gateway to interact with the database
Soen390_IEEE_M6\server\domainObject	Package for domain logic classes
Soen390_IEEE_M6\server\domainObject\dataMapper	Data Mapper which maps the domain model classes with the corresponding table data gateway
Soen390_IEEE_M6\server\domainObject\domainModel	Domain Model represents the major business entities
Soen390_IEEE_M6\server\domainObject\exception	Represent special cases
Soen390_IEEE_M6\server\domainObject\helper	An envelope to store the message and pass it along the application
Soen390_IEEE_M6\server\domainObject\identityMap	Apply the identity map pattern
Soen390_IEEE_M6\server\domainObject\unitOfWork	Apply the unit of work pattern
Soen390_IEEE_M6\server\domainObject\virtualProxy	Apply the lazy load pattern using virtual proxy
Soen390_IEEE_M6\client\pages	GWT source page placeholders with links to reference modules
Soen390_IEEE_M6\client\componentPages	Fully formed page views that correspond to what the client browser renders
Soen390_IEEE_M6\client\widgets	Components developed to enhance functionality of the application

9. Concurrency Issues

This section describes how concurrency is handled in the I3EM web portal application.

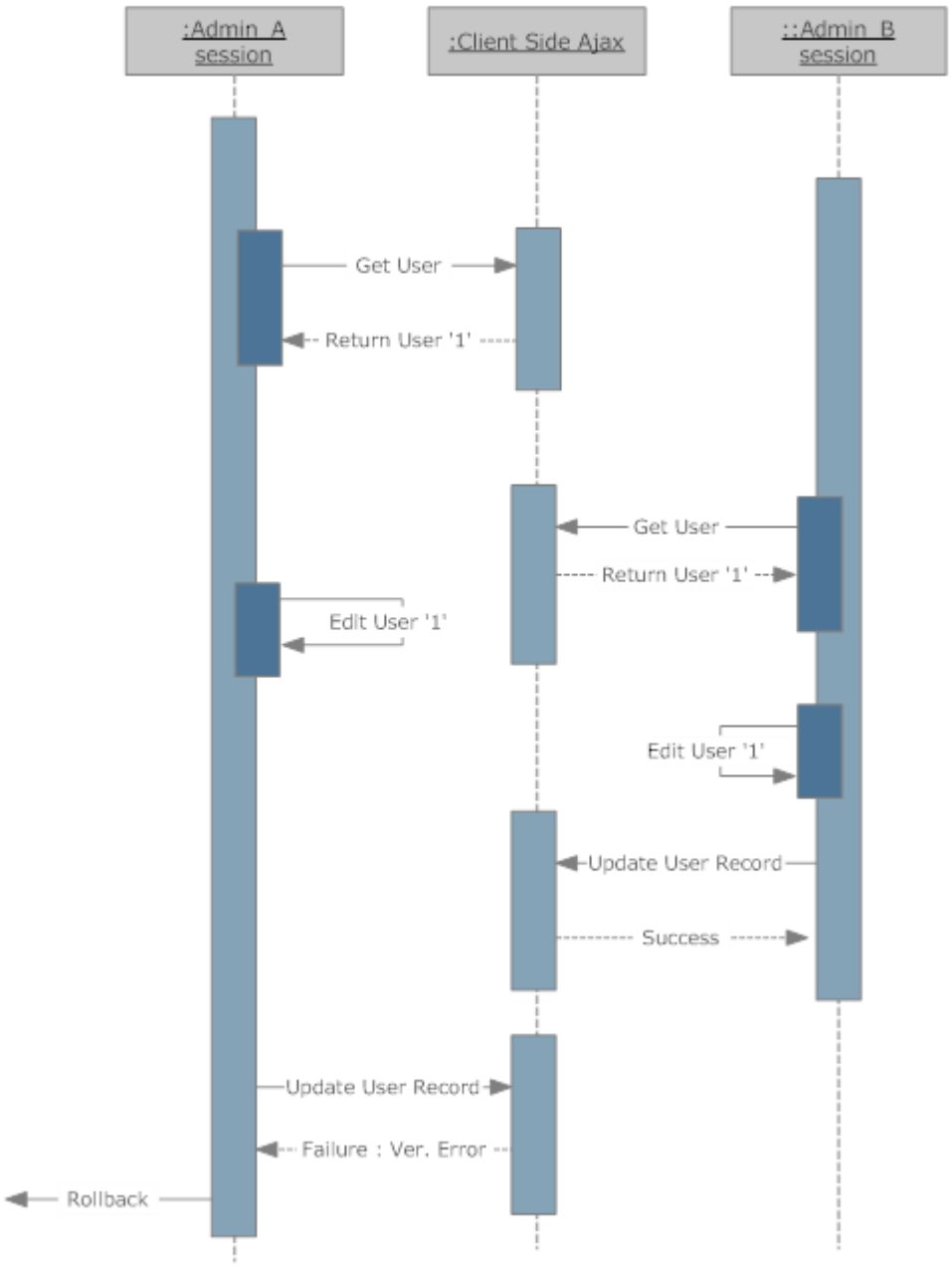
9.1. Optimistic Concurrency management

This "*approach prevents conflicts in data accesses by detecting a conflict and rolling back a transaction*". In the current version of the I3EM, this approach is implemented using a version attribute for the domain objects. An example below illustrates management of a conflict.

This example follows such that two users logged in Administrator privileges "Admin_A" and "Admin_B", two administrators trying to update a filed related to the same user account. Both administrators were previously authenticated and have suitable access to the system. Admin1 selects Manage User Accounts and selects a user account to update. The session of Admin1 now contains the user account record with an associated version number read from the database through the I3EM system application. The next administrator, Admin_B, also selects to update the same record a second session now contains the same user account record and associated version number.

As Admin_A completes the update, the I3EM web application increments the version number and stores the modified user account record back into the database. As Admin_B completes his/her update, the system will detect a mismatch in the record's referenced version number. Consequently, the system will return an error message declaring aversion mismatch. Following that is this case it will initiate a view refresh.

Concurrency Management



10. WEAA Patterns

In the project's last milestone (Milestone 6), a set of Enterprise application architecture patterns were implemented. The following describes these EA patterns.

10.1. Unit of Work

The unit of work pattern describes an object that is used to maintain a list of objects affected by a business transaction. It is used to coordinate writing of any changes. In this web application, the unit of work can help reduce the number of database calls by recording and keeping track of modification made to records in the database. No actual modification to the database is done until a call to update it (commit) is explicitly made. When a controller object needs to execute an insert, update or delete command on a database record, it registers with the unit of work the status of objects that needs to be tracked At the end of the transaction, the controller sends a final update command (commit) to the unit of work which, in turn, will execute all pending changes.

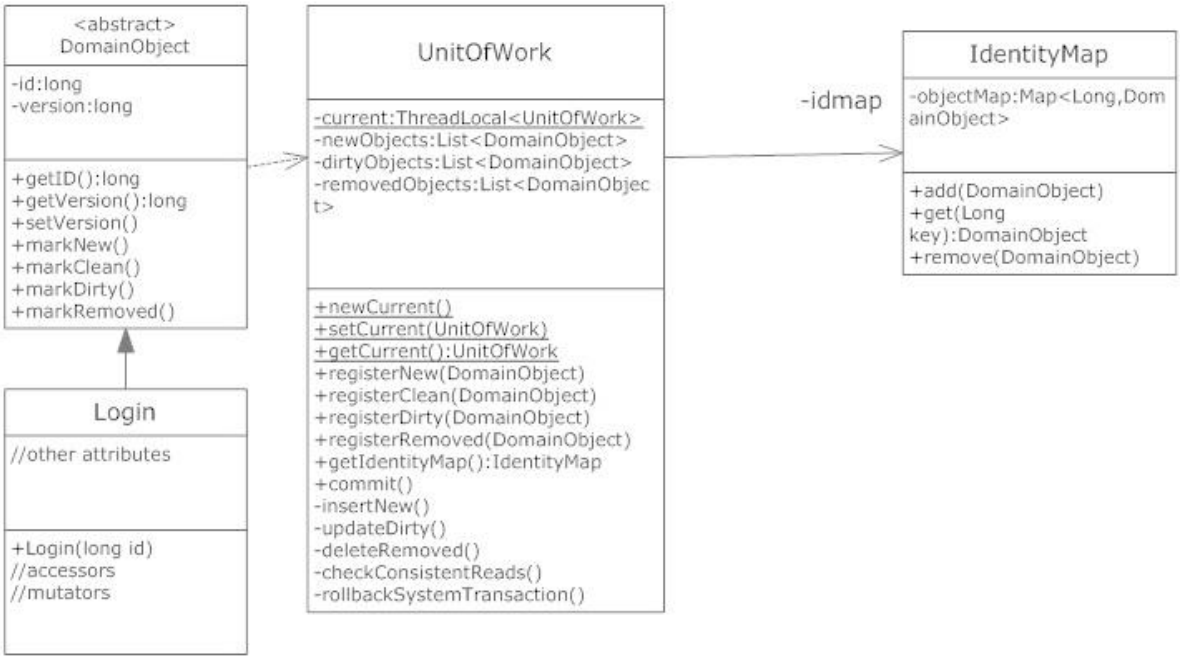
The unit of work class offers the advantage of avoiding multiple database calls which can become expensive operations. It also offers the advantage of avoiding inconsistent reads, a known concurrency problem. It also helps avoid lost updates by rolling back the changes at the commit phase if a concurrency exception is detected.

The unit of work class uses reflection to automatically find the corresponding data mapper to the domain object. This gives us the advantage of having only one generic unit of work for all objects instead of having to code new unit of work classes for each new domain object class.

10.2. Identity Map

The identity Map pattern ensures each object is loaded only once by keeping a map of every loaded object. In this web application system, the identity map can reduce loading from the database by reducing the number of method calls used to access the database. When a data mapper object, such as UserMapper, makes a method call to access a User object it will first check in the identity map object for it. If the object is there, it will be returned. If it is not there, the data mapper object will retrieve it from the database and register it as a clean object with the unit of work which adds the object to its identity map.

The identity map class offers the advantage of reducing the number of method calls made to the database. Since calls to the database can be an expensive operation, this pattern will help reduce costs and enhance system performance. The current developing team therefore decided on implementing identity maps for the advantage offered.



10.3. Virtual Proxy

The virtual proxy pattern describes an object that doesn't contain all of the data you need but knows how to get it. In this web-based enterprise application system, the virtual proxy pattern can reduce memory usage by delaying the loading of an object. An instance of the virtual proxy is placed instead of the real object. When this object is actually needed, for example, when the methods of this object is called, the same methods of the virtual proxy will load the real object from the database and delegate to the corresponding methods of the real object.

The virtual proxy class offers the advantage of reducing the unnecessary occupation of memory resources. This can also further improve system performance since the weight of the objects is lighter.

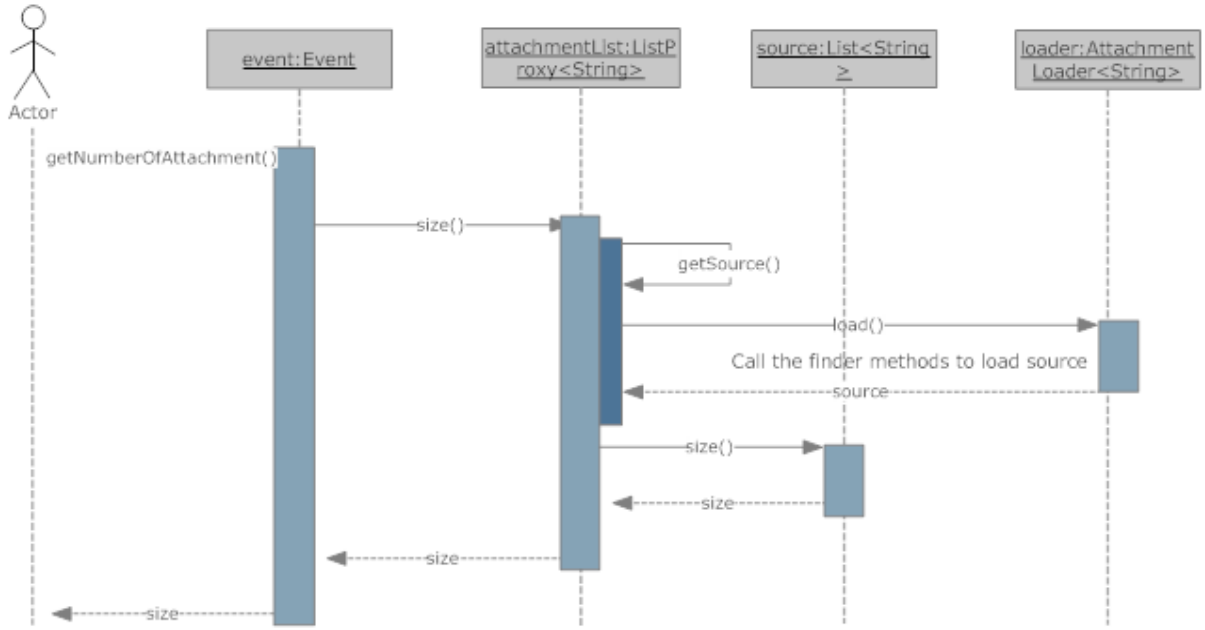


Illustration of Virtual proxy create phase

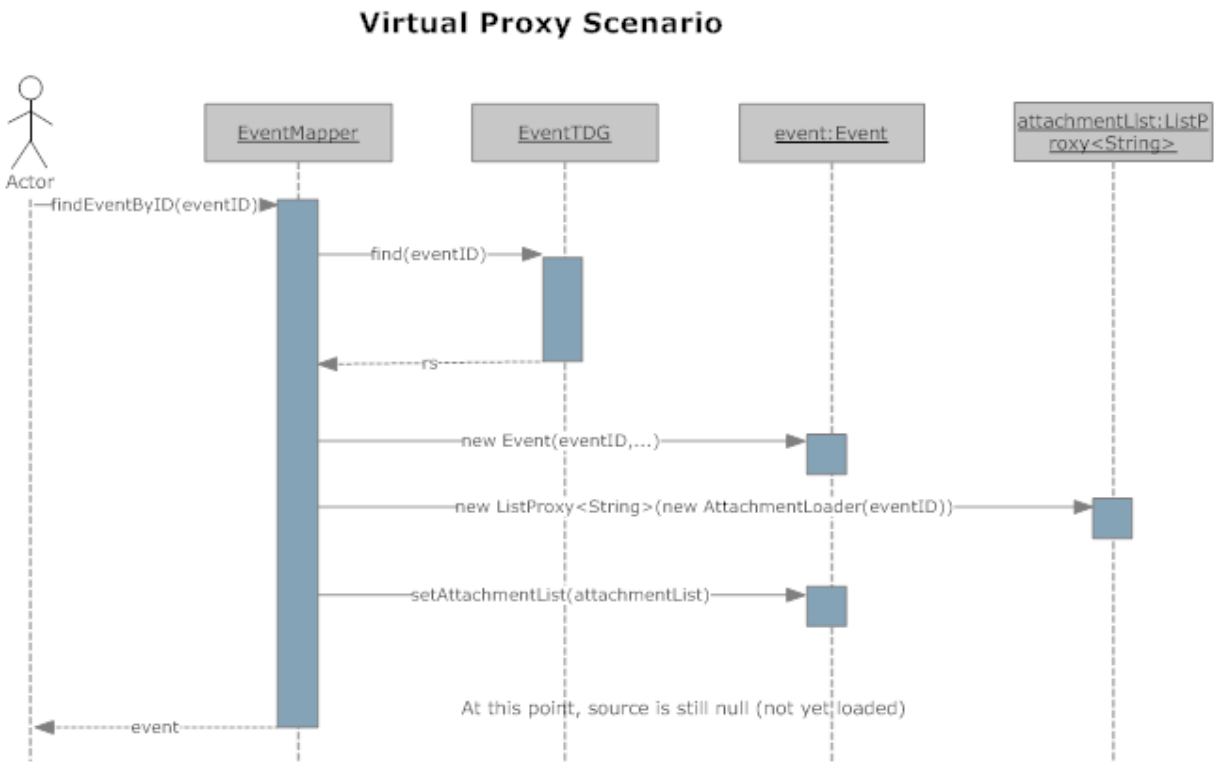


Illustration of Virtual proxy load phase