ClubUML

Architecture Notebook

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and other overarching aspects of the system that shape the design and implementation.

# Architectural goals and philosophy

The goal of the architecture is to provide an extensible design and design roadmap for the development of the ClubUML software according to the use cases and requirement specifications. The purpose of the tool is to a provide a platform for users to interact on UML Diagrams and give the user the capability for compare, merge and download UML diagrams from this tool.

# The Scope of the document

* This document provides a framework to support the developers throughout the development of ClubUML. Architectural components described here will be implemented and provided to the developers by the Architecture Team, as a set of high-level components, base class library, interfaces and development guidelines.
* The algorithms for comparison and merge are out of scope of this document.
* The facilities such as commenting are out of scope of architectural decisions.
* The details of the Ecore and UML2 metamodels are out of scope of this document.
* Database model and detailed implementation of the components are the concern of the design documents and will not be covered here.

# Assumptions and dependencies

The architecture assumes the following:

* A Linux server will be provided by Northeastern University.
* Software packages may be remotely administered and installed on this server.
* The team can program in Java.
* The team can develop on a MySQL database.
* The team has Java-based web service experience: JavaScript, JSP, etc.
* The users will have Eclipse with the Eclipse Modeling Framework (or equivalent) to create UML class diagrams in .ecore format.
* The users will have Papyrus tool (an Eclipse plugin) for creation of different UML diagrams.

# Terminologies

* The term *Diagram Visualization* refers to the action of converting UML files to images in order to be shown to the viewer of the website.

# Architecturally significant requirements

The major architecturally significant requirements are as follows:

* To support the different UML diagram types.

# Issues to be covered in later versions

As of now, the clear specification of the model that can generically capture all the different types of UML diagrams that are concerned, is yet to be finished. One of the students is working on it and it will be discussed in the class before any finalization and implementation.

# Decisions, constraints, and justifications

The following decisions were made to address the requirements:

### The language will be Java.

Java is a cross-platform language that is not tied to specific chip architecture. This means that it can run on any machine, which is a goal of ClubUML. Also, Java is the most familiar language to the developers, which reduces any learning curve in the development cycle.

### The IDE will be Eclipse.

Eclipse is a free, powerful Integrated Development Environment (IDE) that supports Java. Like Java, it is also available for any architecture. In addition, most of the developers are already familiar with Eclipse which will enable a quicker development cycle.

### The database will be MySQL.

MySQL is a fully-functional relational database that is free and open-source and with which many on the development team are familiar. MySQL is a relational database that is open source and free. Oracle was considered, however, the licensing is very expensive. PostGREs was also considered, but MySQL was chosen as it met the requirements and the developers have more experience in it.

### The web server will be Apache Tomcat.

Tomcat is an open-source Java-based web server. Again, many on the team have experience with Tomcat. Besides having built-in Java support, it is also standard in the field and current infrastructure in the deployment environment. No alternatives were considered.

### A Linux server provided by Northeastern will host the application.

This will be free to ClubUML and will be accessible from anywhere using the internet.

### The UML diagrams will be created using Papyrus plugin for Eclipse.

Papyrus is the tool of choice for ClubUML. This an open source tool, which can installed as an Eclipse Plugin. This tool supports UML 2.0 and also supports the following diagrams: Activity Diagram, Class Diagram, Communication Diagram, Component Diagram, Composite Structure Diagram, Deployment Diagram, Package Diagram, Sequence Diagram, State Machine Diagram, and Use Case Diagram. Only Class diagrams and Sequence diagrams will be supported this semester.

### The diagrams will be uploaded in the XMI Format or Ecore format.

The Ecore format for class diagrams will be supported for backward compatibility. The UML diagrams generated by Papyrus are stored in the XMI format. Papyrus model generates three files and the User Interface shall add support for uploading multiple files. The files uploaded will saved in a folder with a unique date and time field.

### Merging will be only be made between two diagrams at a time.

Only one user will be allowed to access the merging tool at a time. This will be done for system simplicity. By disallowing concurrent merging, ClubUML will not get out of sync while merging.

### Compare will be only performed on diagrams generated using the same tool.

The tool will support compare between ECORE class diagrams and support compare between class diagrams generated using the Papyrus and compare between sequence diagrams between sequence diagrams generated using Papyrus.

### Merge functionality will be only performed on diagrams generated using the Papyrus UML tool

The merge functionality will only supported for Class and Sequence diagrams generated using the Papyrus UML tool. Merging ECORE Class diagrams is out of the scope of this project.

### Download functionality will be supported

The User will be able to download a diagram from ClubUML and import it into Eclipse. The import mechanism is beyond the scope of this project.

### ClubUML will only have one project.

Again, this was done for simplicity, although some code is in place to allow multiple projects as a future enhancement.

### MySQL will only store links to the stored UML diagrams in the database, not the diagrams themselves.

This will save on the size of the database, which should make it faster. In addition, a MySQL database doesn’t handle Binary Large Object (BLOB) data like some other database applications.

### Servlets will be used as the controller mechanism.

This includes JSP (Java Server Pages), as well as generic web languages like HTML and CSS. These mesh well with Java, are handled well with Eclipse, and are more familiar to the developers than their Microsoft counterparts.

### SQL will be the database query language.

### This is the standard for modern relational databases, and is what MySQL requires. No alternatives were considered.

### The Java Database Connectivity (JDBC) API will interface between Java and MySQL.

An abstraction layer/library was considered, such as MyBatis, however, the developers were familiar with Java Database Connectivity (JDBC) and did not want to introduce another learning curve by adding an additional library.

### The UML diagrams will be generated into .png files and will be saved to disk.

This will prevent any redrawing of the same diagrams twice, which will give speed at the cost of disk space. This decision is mainly based on the first milestone of the software and subject to change.

### The architecture will be a layered architecture

The basic model of the layered architecture will be:

Web Browser

Web Server

Storage

Client

Uses

# Architectural Mechanisms

## Architectural Mechanism 1 – Web server

This will drive the entire project. An Apache Tomcat web server instance will be running on a Northeastern-provided Linux server. Tomcat is the Apache web server that supports Java-based applications. This has been prototyped and is working.

## Architectural Mechanism 2 - Database

All users and project data will either be stored in the database, or have a reference to their location stored (as in the case of the UML files, both .core and .png). MySQL is the database, because it’s an open source relational database that is familiar to most of the development team.

This has been prototyped and is working.

## Architectural Mechanism 3 – UI/Front End (view)

A combination of HTML/CSS will be used to design the webpage User Interface (UI), as most of the team has had some training in this as part of their academic program.

## Architectural Mechanism 4 – High-level Components

This consists of a series of high-level components as discussed in the next section. The main goal is to provide a level of abstraction for the developer’s code to compare and possibly merge the UML diagrams. It also caters for facilitating any extension during the next iteration of the software. The other goal is to isolate the development teams’ tasks and promote a low coupled design mindset.

## Architectural Mechanism 5 – UML Diagram Parsing

The diagrams will be parsed using a standard SAX Parser. The Elements and its corresponding attributes will be saved in a list. The UML Parser will then place each element based on its UML attribute.

## Architectural Mechanism 6 – Diagram Visualization

UMLGraph and GraphViz are responsible for class diagram visualization. The upload code will parse the .ecore/.uml and .notation file(s) and create a simple .java file which will be the input to UMLGraph. The output of UMLGraph (known as dot format) will be processed through GraphViz (a generic graph visualization tool, not UML specific), which will produce the diagram as a .png file.

For the extension to integrate sequence diagram in the existing project, the new system will continue with UMLGraph to define the sequence diagram in .pic file format and then visualize by pic2plot graphing tool, which will produce the diagram as a .png or .gif file.

## Architectural Mechanism 7 – MVC Architecture (model-view-controller)

The MVC architecture is used throughout the application. This allows a separation of business logic from data and from presentation code. This means that there will be separate code for the Controller (servlets), and that does not mix with our Presentation code (JavaScript/JSP). Similarly, the data is stored in JavaBeans, which are again separate.

## Architectural Mechanism 8 – Servlet

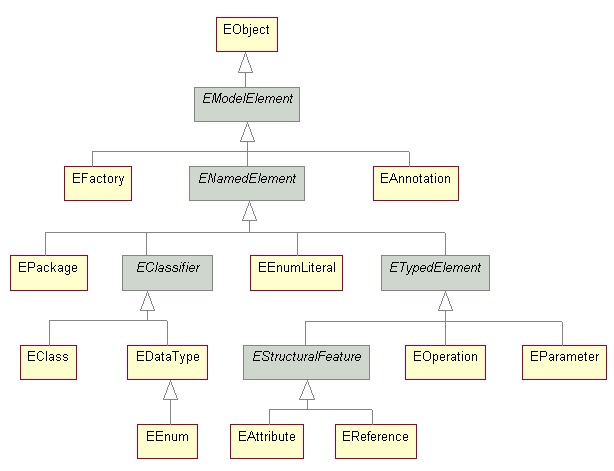
Java servlets will provide our web mechanism. This is where all of the application logic is done. This is considered the ‘Controller’ portion of our MVC architecture.

## Architectural Mechanism 9– Model

The JavaBeans model is used in the software, which makes the code a system of reusable software components. This is done by writing classes in a particular convention. Using JavaBeans allows us to encapsulate the data and have a standard method of accessing it, which means it can be completely separate from the view or controller portions of the code. These JavaBeans are the only code pieces that directly manipulate data in the system.

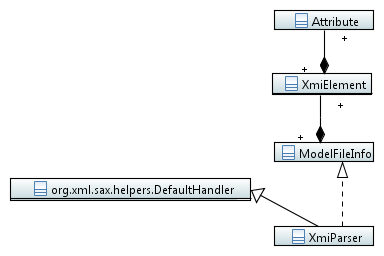
## Architectural Mechanism 10 – Parsing through Ecore Data Model

The Ecore Data Model, as defined here, will model the diagrams in code:



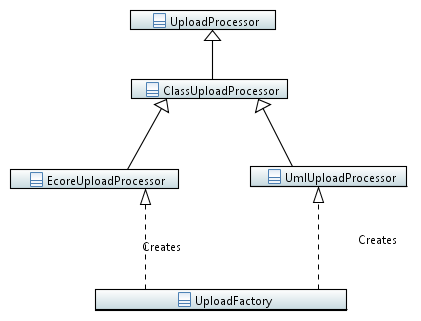
## Architectural Mechanism 11 – Parsing XMI Model

The SAX Parser will be used for parsing the UML diagram. The XMI Parser Model, as defined here, will be developed.



## Architectural Mechanism 12 – Upload Model

The upload model will implement Parameterized Factory Method Design Pattern. The model below will be developed.



## Architectural Mechanism 13 – Compare and Merge Model

The compare model will implement the Factory Method design pattern. The model below will be developed.

## 

Comparer will also include ClassElement, AttributeElement, OperationElement and RelationshipElement to store information for comparing operations.

Merge details are currently being worked on. Will most likely involve some kind of factory like the rest to return a merge object.)

## Architectural Mechanism 14 – Download Model

[TBD]

## Architectural Mechanism 15 – Delete Entries

[TBD]

# Layers or architectural framework

ClubUML will use the Layer framework. A high level view of the layers is given here:

Web Browser

Client-Side Logic

Web Server

Server-Side Logic

Storage

Client

Uses

Interface

Application Logic

# Architectural views

## Recommended views

* **Logical:** 
  + Layered Architecture
    - The following is a diagram of the layers of the system

Delete

Parse Ecore

ModelType

UML Parser

Storage

XML Parser

CompareProvider

Compare

Disk

Database

UserManagement

FileManagement

Interface

Application Logic

DisplayDiagram

WebClient

WebServer

* **Operational:**
  + ClubUML runs in a single process, aside from the web server which handles multiple connections to the server at a time. That is done by Tomcat automatically, and is not controllable by ClubUML
* **Use case:** 
  + <This section will be completed as the analysis team provide the required artifacts.>