System and Software Architecture Description (SSAD)

Student Scheduling System

Team #10

Bo Wang: Project Manager / Quality Focal Point / Implementation Team
Bohan Zheng: Prototyper / Builder / Implementation Team
Chenyang Bai: Feasibility Analyst / Operational Concept Engineer / NDI NCS Evaluator / Implementation Team
Frank Varela: IIV&V / Shaper / Quality Focal Point
Rui Tong: Requirements Engineer / Operational Concept Engineer / NDI / NCS Acquirer / Implementation Team
Shuai Wang: System/Software Architect / UML Modeler / Implementation Team
Xiaoran Li: Life Cycle Planner / Tester / Implementation Team

Version History

Date	Author	Version	Changes made	Rationale
4/4/2014	Rui Tong	1.0	Document created. Filled in sections 1, 2.1.1 – 2.1.4, 4.1.3	Created initial version of the document.
10/22/2013	Rui Tong	1.1	Updated both the figure and the table directory	
12/01/2013	Rui Tong Bo Wang	2.0	Filled in sections 2.1.2, 4 Updated use case diagram	Class diagram, sequence diagram and deployment diagram added
12/06/2013	Rui Tong Bo Wang	3.0	Filled up all the sections Updated design rationale Added hardware and software component diagram	Final version for DCR phase
01/21/2013	Rui Tong	3.1	Updated backend class diagram, frontend class diagram, software component diagram and entity relationship diagram according to the change on database (semester entity deleted) which was raised up during the meeting on 01/20/2013	Changes are basically based on the change of database.
04/04/2013	Rui Tong	4.0	Updated Entity relationship diagram, Deployment diagram and Software Component Diagram according to instructor's feedback	Changes are basically based on instructor's feedback

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1 Purpose of the SSAD

The purpose of the SSAD is to document the results of the object-oriented analysis and design (OOA&D) of the Student Scheduling System. The SSAD is used by the builder (programmer/developer) as reference to the system architecture. The Student Scheduling System should be faithful to the architecture specified in this document. Furthermore, the SSAD is used by the maintainer and clients to help understand the structure of the system once the proposed system is delivered.

1.2 Status of the SSAD

This is SSAD – version 2.0 for draft DC package. Section 1 and section 2 are completed.

2. System Analysis

2.1 System Analysis Overview

The purpose of the Student Scheduling System is to so save students' and advisers' time spent on constructing study plan at Stevens University of Technology. The system is intended to achieve goal though automation of study plan construction. Course directors enter information in the system about the courses and degree requirements for each of three degrees (CS, SyS, IS) offered at Stevens department of Computer Science. After that students are able to enter their requirements for the study plan in the system such as desired year of graduation, preferred elective courses, and transfer credits; the system makes attempt to construct study plan satisfying these requirements. If it is not possible to find schedule satisfying the requirements the system suggests the student to relax constrains for the study plan and repeat attempt to construct the study plan.

2.1.1 System Context

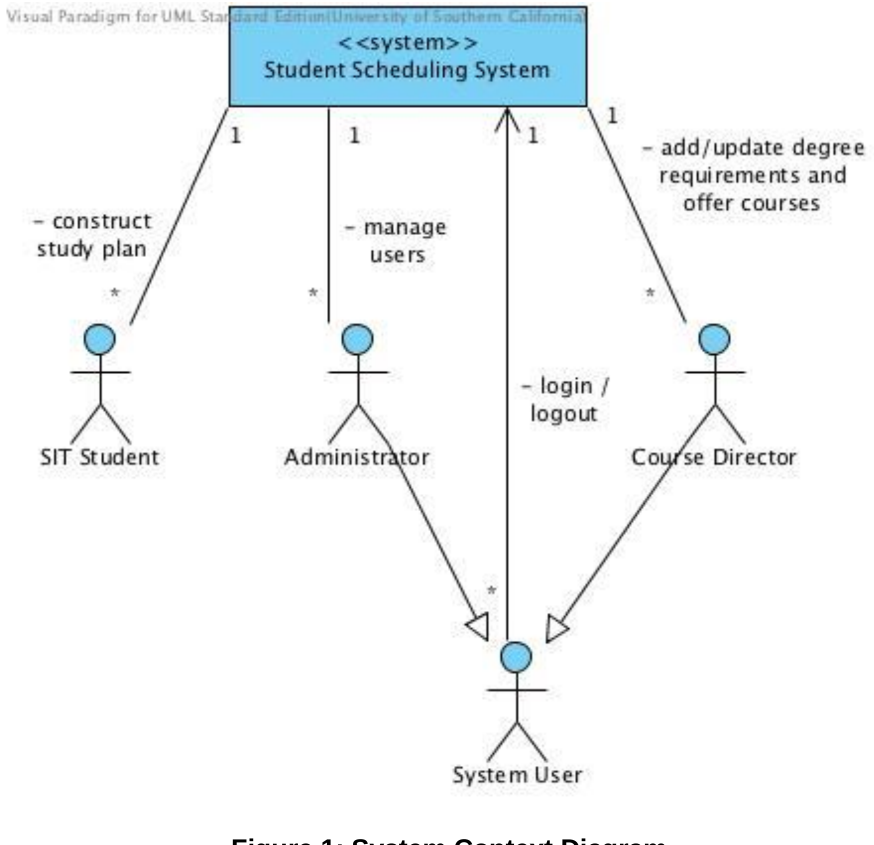
Visual Paradigm for UML Standard Edition(University of Southern California)
<<system>>
Student Scheduling System
1
1
1
1
- construct study plan
- manage users
- add/update degree requirements and offer courses
*
*
*
- login / logout
SIT Student
Administrator
Course Director
*
System User

Figure 1: System Context Diagram

Table 1: Actors Summary

Actor	Description	Responsibilities
System User	Any user of the system including course director and administrator	• Log in / log out
SIT Student	Stevens' students that use the system to construct study plan.	• Enter constraints for the study plan (year of graduation, electives, and so on) and request the system to construct the study plan.
Course director	Stevens' faculty staff who is responsible for adding and updating available courses and degree requirements.	• Add and/or update available courses and degree requirements.
System administrator	Stevens' staff who is responsible for adding, deleting, and updating user profile (accounts).	• Add, delete, and update user profile.

2.1.2 Artifacts and Information

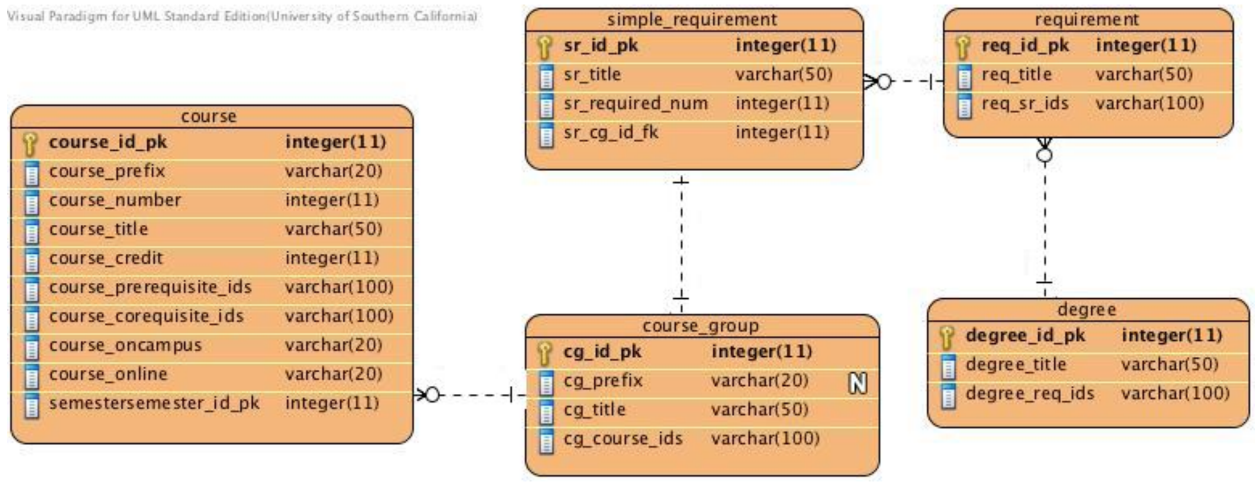
Visual Paradigm for UML Standard Edition(University of Southern California)
course
course_id_pk integer(11)
course_prefix varchar(20)
course_number integer(11)
course_title varchar(50)
course_credit integer(11)
course_prerequisite_ids varchar(100)
course_corequisite_ids varchar(100)
course_oncampus varchar(20)
course_online varchar(20)
semestersemester_id_pk integer(11)
simple_requirement
sr_id_pk integer(11)
sr_title varchar(50)
sr_required_num integer(11)
sr_cg_id_fk integer(11)
requirement
req_id_pk integer(11)
req_title varchar(50)
req_sr_ids varchar(100)
course_group
cg_id_pk integer(11)
cg_prefix varchar(20)
N
cg_title varchar(50)
cg_course_ids varchar(100)
degree
degree_id_pk integer(11)
degree_title varchar(50)
degree_req_ids varchar(100)

Figure 2 Artifacts and Information Diagram

Table 2 Artifacts and Information Summary

Artifact	Purpose
Semester	Contains description of the semester and its name (Fall2013, Spring2014…). These semesters are used to specify in which course is available.

Course	Contains description of the course and its attributes such as number of credits, title, prefix (CS, HUM, …), on campus or on line and its prerequisites and corequisites. These courses are used to specify course group (such as preferred electives).
Course Group	Contains description of the course group and its attributes such as title and the courses included. These course groups are used to specify simple requirement.
Simple Requirement	Contains description of the simple requirement and its attributes such as title and the course groups included. These simple requirements are used to specify requirement.
Requirement	Represents a constraint for the study plan satisfying the requirements of a particular degree.
Degree	Contains general information about the degree (such as name).

2.1.3 Behavior

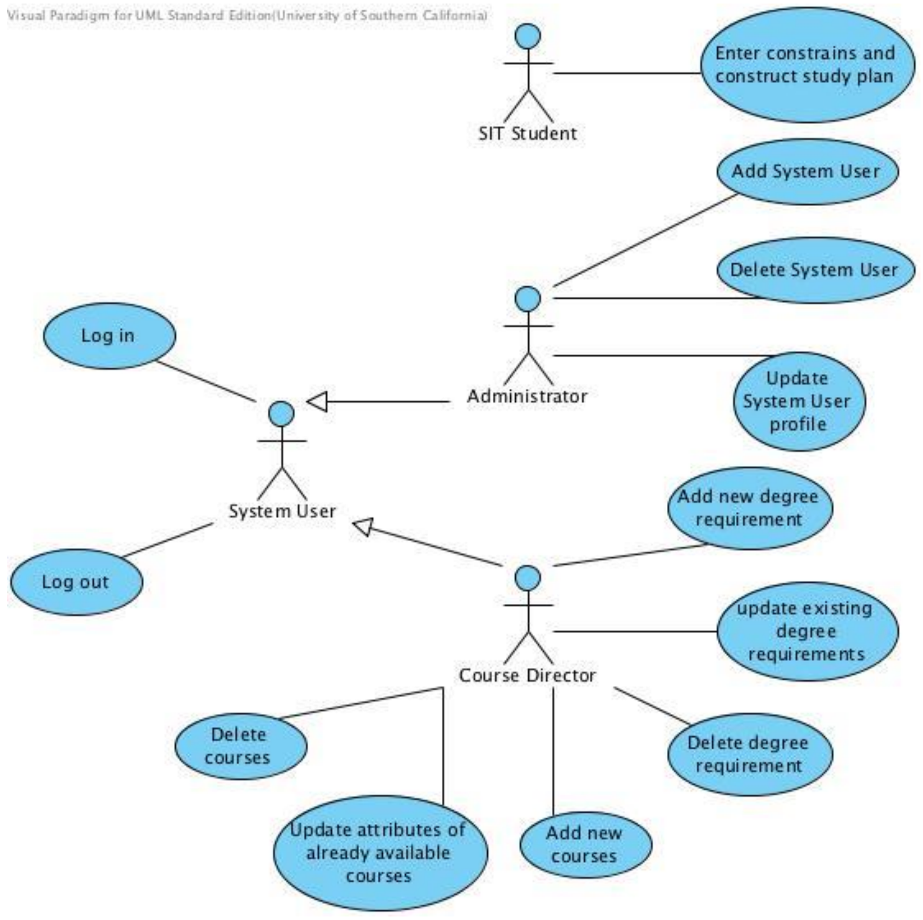
Visual Paradigm for UML Standard Edition(University of Southern California)
SIT Student
Enter constrains and construct study plan
Add System User
Delete System User
Update System User profile
Administrator
Log in
System User
Log out
Add new degree requirement
update existing degree requirements
Course Director
Delete degree requirement
Delete courses
Update attributes of already available courses
Add new courses

Figure 3: Process Diagram

2.1.3.1 Authentication

2.1.3.1.1 Log in

Table 3: Process Description

Identifier	UC-1: Log in
Purpose	WC_1533(Last Year): Authorize the system user to log in the system and assign the system user associated system role.
Requirements	System must provide user privileges according his/her role (course director/ administrator).
Development Risks	None
Pre-conditions	Log in page is opened.
Post-conditions	Authorized system users get access to the system, and initial page will be shown. Unauthorized system users will be denied.

Table 4: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Enters a username and password on the login page.	
2	Clicks the "Sign in" button.	
3		System retrieves information associated with given username (role, password hash) form DB and checks password (password hash). Then it shows initial page according to the system user's role.

Table 5: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	Enters a username and password on the login page.	
2	Clicks the "Sign in" button.	
3		System retrieves information associated with given username (role, password hash) form DB and checks password (password hash). Check result is false. Then system shows message: "Username or password is wrong. Please try again."
4		System shows log in page again.

There is no Alternate Courses of Action. Blank fields are treated as wrong username or password.

2.1.3.1.2 Log out

Table 6: Process Description

Identifier	UC-2: Log out
Purpose	WC_1533(Last Year): Allow the user to log out the system.
Requirements	System user already has the privileges according his/her role (course director/ system administrator).
Development Risks	None.
Pre-conditions	User logged in the system. User can be on any page (accept login page, because user is already in the system); "Log out" link is available on every page.
Post-conditions	User's session is closed, and log in page will be shown.

Table 7: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Clicks the "Sign out" button.	
2		System shows message that session was successfully closed, and then it shows log in page again.

There is no Exceptional or Alternate Courses of Action. Blank fields are treated as wrong username or password.

2.1.3.2 User management

2.1.3.2.1 Add user

Table 8: Process Description

Identifier	UC-3: Adding new user.
Purpose	WC_1533(Last Year): Give the administrator ability to add new user in the system.
Requirements	System must provide user privileges according his/her role (student/course director/system administrator). User is on the initial page.
Development Risks	None.
Pre-conditions	User successfully entered the system as a system administrator. User is on initial page.
Post-conditions	One user profile is added to the system.

Table 9: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Clicks the button "Add new user"	
2		System shows empty user profile.
3	Enters user's name, login, role, and password.	
4	Clicks the button "Add"	
5		System checks correctness of the data. Data is correct. System adds new user and confirmation that user was added successfully.

Table 10: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1-4	Actions and responses are the same as in typical scenario.	
5		System checks correctness of the data. Data is incorrect (for example, username already exists). System shows appropriate message. System shows the same page.

There is no Alternate Course of Action.

2.1.3.2.2 Delete user

Table 11: Process Description

Identifier	UC-4: Delete user.
Purpose	WC_1533(Last Year): Give the administrator ability to delete existing user from the system.
Requirements	System must provide user privileges according his/her role (student/course director/system administrator). User is on the initial page.
Development Risks	None.
Pre-conditions	User successfully entered the system as a system administrator. System contains user for deletion. Edit user profile page is opened.
Post-conditions	One user profile is deleted from the system.

Table 12: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Presses the button "Delete user"	
2		System requests confirmation for deletion the user.
3	Confirms deletion	
4		System deletes the user and shows confirmation that user was deleted successfully. System redirects user to the initial page.

Table 13: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1-3	Actions and responses are the same as in typical scenario.	
4		The user cannot be deleted (DB is not accessible or user was not found). System shows appropriate message (ex. "DB is accessible" or "User was not found. Probably it was already deleted")

There is no Alternate Courses of Action that have special processing.

2.1.3.2.3 Update user profile

Table 14: Process Description

Identifier	UC-5: Update user profile.
Purpose	WC_1533(Last Year): Give the administrator ability to update existing user from the system (reset password, change name, and so on).
Requirements	System must provide user privileges according his/her role (student/course director/system administrator). User is on the initial page.
Development Risks	None.
Pre-conditions	User successfully entered the system as a system administrator. System contains user for updating. . Edit user profile page is opened.
Post-conditions	One user profile is updated.

Table 15: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Updates fields (for example, name or new password) and press the button "Save"	
2		System checks correctness of the data. Data is correct. System updates user profile and confirmation that user was added successfully.

Table 16: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	Actions and responses are the same as in typical scenario.	
2		System checks correctness of the data. Data it is incorrect (for example, username already exists). System shows

		appropriate message on the same page.

There is no Alternate Course of Action.

2.1.3.3 Study plan construction

2.1.3.3.1 Entering student defined constraints and study plan construction

Table 17: Process Description

Identifier	UC-6: Entering student defined constraints and study plan construction.
Purpose	Give the student a tool to specify his/her desires (degree, year of graduation, elective courses, and so on) and construct a study plan, which satisfies these constrains.
Requirements	WC_2658: The algorithm for generating the course schedule shall have guaranteed bounds on its execution time and/or bounds on optimality. WC_2657: The system shall auto generate class/course schedule for undergraduate students given the following constraint: 1. Degree program requirements 2. Students desired courses related to his schedule WC_2659: As a student, I can see which constraints need to be relaxed if the schedule is not generated in the fixed amount of time. WC_2655: Better error explanation: 1. Prerequisites and corequisites error descriptions 2. Complex course requirements error description WC_2654: Improve layout of course list of degree requirements on the student site.
Development Risks	Construction of the study plan may be algorithmically difficult. User interface for specifying all the possible constrains is difficult to implement. It may require extra time.
Pre-conditions	User successfully entered the system as a student. User is on the Degree program selection page.
Post-conditions	Student got a study plan or notification that it is impossible to construct it.

Table 18: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Chooses degree and year (it is not a year of graduation; system will use degree requirements for this year)	
2		System shows list of requirements such as mandatory courses, electives that student may choose.
3	Enters desired semester of graduation, coursed for which s/he has credits, desired electives, and presses the button "Construct study plan"	
4		System makes an attempt to construct study plan. Study plan was found. The system shows the study plan.

Table 19: Alternate Course of Action

Seq#	Actor's Action	System's Response
1-3	Actions and responses are the same as in typical scenario.	
4		System makes an attempt to construct study plan. It is impossible to construct study plan. The system shows a message "Unfortunately it is impossible to construct a study plan satisfying all the constraints. Try to relax constraints (for example, let the system choose elective courses) and repeat search again".
5		Then the system shows the page with the study plan constraints.

There is no Exceptional Courses of Action.

2.1.3.4 Entering and modification of the degree requirements

2.1.3.4.1 Add new requirement

Table 20: Process Description

Identifier	UC-7: Adding new requirement
Purpose	Give the course director a tool to specify new degree requirement.
Requirements	WC_1350(Last Year): Administrator can input degree requirements for each year of entry and degree combinations. WC_1329(Last Year): As an administrator I must be able to enter degree requirements as complex as those that have been in effect, for the CS, IS, and CyS undergrad degrees and are listed, on the Stevens CS dept. website as well as additional clarifying details provided by the client to the team.
Development Risks	Construction of the study plan may be algorithmically difficult. User interface for specifying all the possible constrains is difficult to implement. It may require extra time.
Pre-conditions	User successfully entered the system as a course director. User is on Add new requirement page.
Post-conditions	Updated degree requirements.

Table 21: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Chooses requirement type from the available list of types	
2		System shows parameters which must be specified for the requirement. (For example, if type is "Select *n* courses from the list" we need to specify *n* and the list of the courses. If it is a composition of other requirements then user need to choose other requirements)
3	Specifies necessary parameters and press the button "Add"	
4		System checks correctness of the data. Data is correct. The system updates requirement and shows confirmation that new requirement was added successfully.

Table 22: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1-3	Actions and responses are the same as in typical scenario.	
4		System checks correctness of the data. Data it is incorrect (for example, not all fields are field in). The system shows appropriate message on the same page.

There is no Alternate Courses of Action.

2.1.3.4.2 Update requirement

Table 23: Process Description

Identifier	UC-8: Updating the requirement
Purpose	Give the course director a tool to change existing degree requirement.
Requirements	WC_1350(Last Year): Administrator can input degree requirements for each year of entry and degree combinations. WC_1329(Last Year): As an administrator I must be able to enter degree requirements as complex as those that have been in effect, for the CS, IS, and CyS undergrad degrees and are listed, on the Stevens CS dept. website as well as additional clarifying details provided by the client to the team.
Development Risks	Construction of the study plan may be algorithmically difficult. User interface for specifying all the possible constrains is difficult to implement. It may require extra time.
Pre-conditions	User successfully entered the system as a course director. User is on Edit requirement page.
Post-conditions	Updated degree requirements.

Table 24: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Updates requirement's parameters and presses the button "Save"	
2		System checks correctness of the data. Data is correct; system updates user profile and shows confirmation that the requirement was updated successfully.

Table 25: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	Actions and responses are the same as in typical scenario.	
2		System checks correctness of the data. Data is incorrect (for example, not all fields are field in) system shows appropriate message on the same page.

There is no Alternate Courses of Action.

2.1.3.4.3 Delete requirement

Table 26: Process Description

Identifier	UC-9: Delete the requirement
Purpose	Give the course director a tool to delete existing degree requirement.
Requirements	WC_1350(Last Year): Administrator can input degree requirements for each year of entry and degree combinations. WC_1329(Last Year): As an administrator I must be able to enter degree requirements as complex as those that have been in effect, for the CS, IS, and CyS undergrad degrees and are listed, on the Stevens CS dept. website as well as additional clarifying details provided by the client to the team.
Development Risks	Construction of the study plan may be algorithmically difficult. User interface for specifying all the possible constrains is difficult to implement. It may require extra time.
Pre-conditions	User successfully entered the system as a course director. User is on Edit requirement page.
Post-conditions	Deleted degree requirements.

Table 27: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Delete requirement's parameters and presses the button "Yes"	
2		System checks correctness of the data. Data is correct; system updates user profile and shows confirmation that the requirement was deleted successfully.

Table 28: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	Actions and responses are the same as in typical scenario.	
2		System checks correctness of the data. Data is incorrect (for example, the requirement can't be deleted) system shows appropriate message on the same page.

There is no Alternate Courses of Action.

2.1.3.4.4 Add new course

Table 29: Process Description

Identifier	UC-10: Adding new course in the system.
Purpose	WC_1349(Last Year): Give the course director a tool to change existing degree requirement.
Requirements	Administrator can input information about individual courses.
Development Risks	None.
Pre-conditions	User successfully entered the system as a course director. User is on Add new course page.
Post-conditions	Updated list of the available courses.

Table 30: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Enters course attributes such as name, number of credits, type and presses the button "Save"	
2		System checks correctness of the data. Data is correct; system updates user profile and shows confirmation that the requirement was added successfully.

Table 31: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	Actions and responses are the same as in typical scenario.	
2		System checks correctness of the data. Data is incorrect (for example, not all fields are field in); system shows appropriate message on the same page.

There is no Alternate Courses of Action.

2.1.3.4.5 Update the course

Table 32: Process Description

Identifier	UC-11: Updating the course attributes in the system.
Purpose	WC_1349(Last Year): Give the course director a tool to change existing degree requirement.
Requirements	Administrator can input information about individual courses.
Development Risks	None.
Pre-conditions	User successfully entered the system as a course director. User is on Edit course page.
Post-conditions	Updated list of the available courses.

Table 33: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Updates course attributes such as name, number of credits, type and presses the button "Save"	
2		System checks correctness of the data. Data is correct; system updates user profile and shows confirmation that the requirement was updated successfully.

Table 34: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	Actions and responses are the same as in typical scenario.	
2		System checks correctness of the data. Data is incorrect (for example, not all fields are field in); system shows appropriate message on the same page.

There is no Alternate Courses of Action.

2.1.3.4.6 Delete a course

Table 35: Process Description

Identifier	UC-12: Deleting a course in the system.
Purpose	Give the course director a tool to delete existing courses.
Requirements	WC_1349(Last Year): Administrator can input information about individual courses.
Development Risks	None.
Pre-conditions	User successfully entered the system as a course director. User is on Delete course page.
Post-conditions	Updated list of the available courses.

Table 36: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Choose the course and presses the button "Delete"	
2		System checks correctness of the data. Data is correct; system updates user profile and shows confirmation that the course was deleted successfully.

Table 37: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	Actions and responses are the same as in typical scenario.	
2		System checks correctness of the data. Data is incorrect (for example, not all fields are field in); system shows appropriate message on the same page.

There is no Alternate Courses of Action.

2.1.4 Modes of Operation

The Student Scheduling System operates only in one mode.

2.2 System Analysis Rationale

For our project, the most challengeable part is the algorithm. Generating a valid study plan based on student's specific desires is the main goal of our system. However, preliminary analysis done by last semester team shows that finding a satisfactory study plan in Student Scheduling System is typical combinatorial optimization problem, which is an NP-hard problem. In order to improve the efficiency of the algorithm developed by last year team, we came up with a way to separate the whole computing process into several steps, since it is really important to be sure that the system is able to find a solution in a reasonable time.

Another interesting aspect of the system is the data structure representing the degree requirements(prerequisites & co-requisites). As it was mention before Stevens, degree requirements are combine a wide range of different requirements of high complexity. For this reason, building a reasonable and suitable data structure is quite important. As shown below, we picked a modified DAG to be the data structure.

The process of this analysis includes the following steps:

1. Identification of degree requirement types (informal description)
2. Identification of possible student defined constraints (informal description)
3. Building data structure (DAG) for degree programs (data structure modeling)
4. Developing test case which covers most of the constraints (description of degree requirements for one degree and description of student desires for study plan).
5. Refining UI prototype for the test case.
6. Developing test program for finding study plan.

These steps allow getting feedback from the client as soon as possible and mitigating risks associated with constraint solving complexity.

3. Technology-Independent Model

Since we need to satisfy technology-dependent requirements defined by client stakeholders (otherwise they will not be able to maintain the system), this section is skipped. Moreover, we already know technology-specific details of the system design; therefore, this section is intentionally omitted.

4. Technology-Specific System Design

The following diagram (Figure 3) shows the Hardware structure. The web/application /DBMS server will be connected with client workstations through the Internet Network. This is a typical web application in MVC.

4.1 Design Overview

4.1.1 System Structure

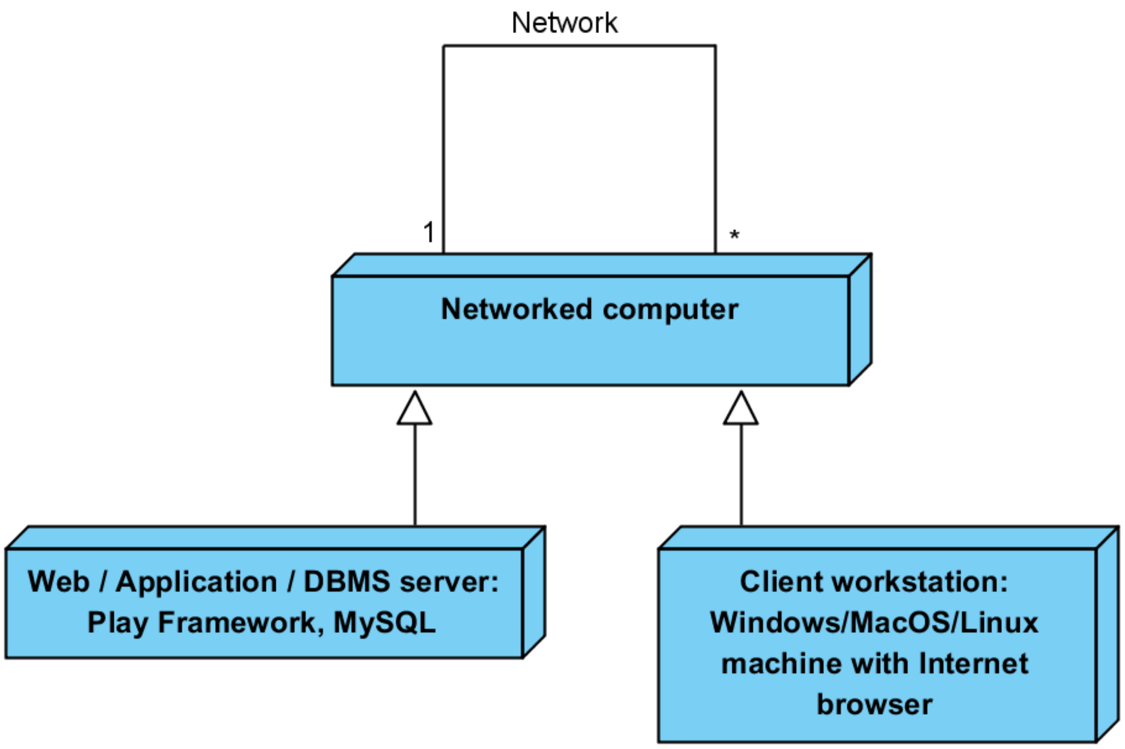
Network
1
*
Networked computer
Web / Application / DBMS server: Play Framework, MySQL
Client workstation: Windows/MacOS/Linux machine with Internet browser

Figure 4: Hardware Component Class Diagram

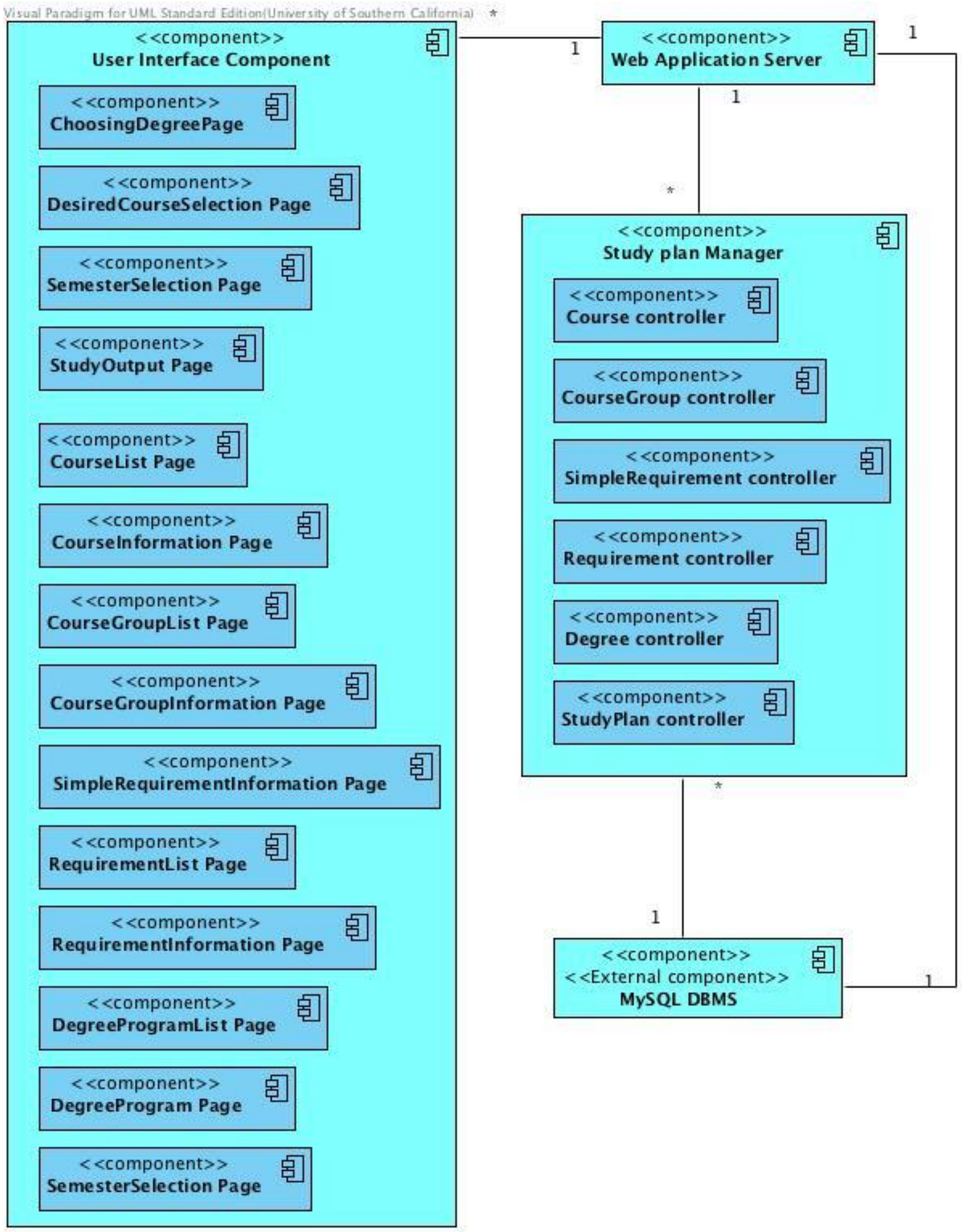

Figure 5: Software Component Class Diagram

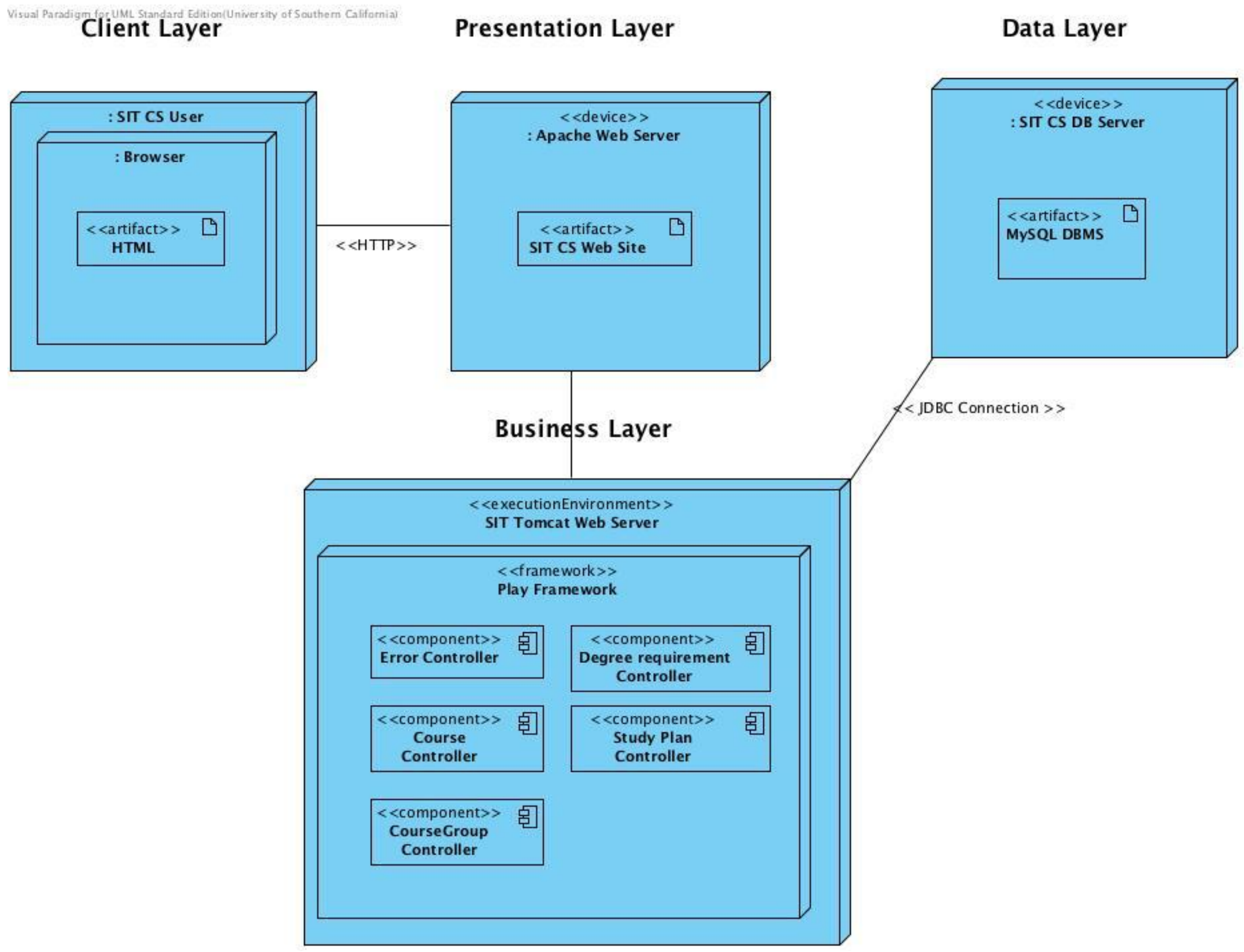

Figure 6: Deployment Diagram

Table 38: Hardware Component Description

Hardware Component	Description
Web/Application/DBMS Server	This component accepts connections from all users. Web and Application server are provided by Play framework. Since data base is comparatively small and does not require high-performance hardware it was decided to host with Web and Application server.
Client workstation	This is a user computer. It must have a web browser (Mozilla Firefox) and it must have Internet network.

Table 39: Software Component Description

Software Component	Description
User interface component	This component contains all the webpages of the system.
Study plan manager	This component contains controller for study plan construction webpage and classes that performs prerequisites and corequisites checker and constraint solver, then output final study plan.
Web/Application Server	This component is the web/application server in Stevens Institute Technology.
MySQL DB server	Data base that stores all entities of the system. This is data layer of the system.

4.1.2 Design Classes

The design classes describe the relationships among the boundary, control, and entity classes of the Student Scheduling System.

4.1.2.1 Frontend class Diagram

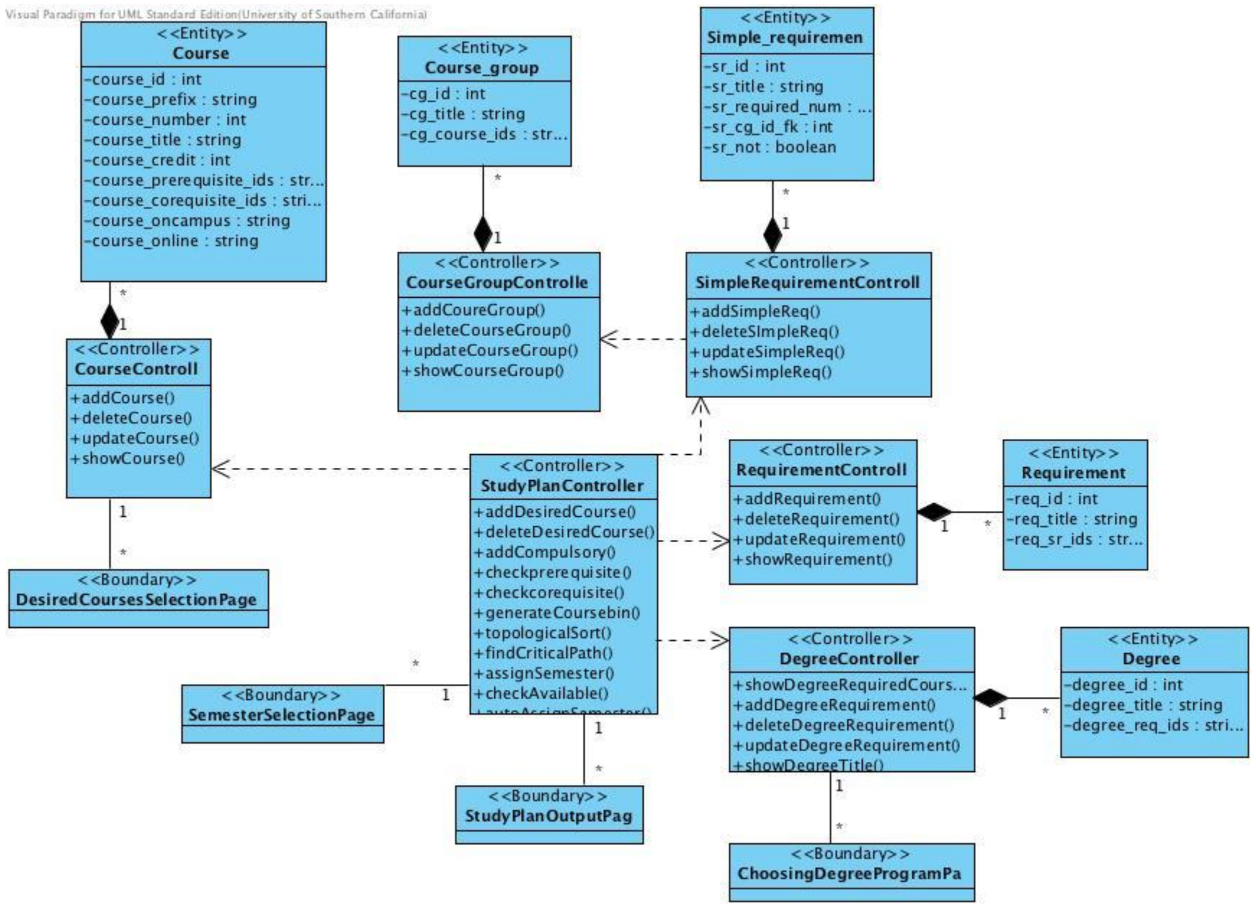

Figure 7: Frontend Class Diagram

Table 40: Front End Classes Description

Class	Type	Description
ChossingDegreeProgramPage	Boundary	This webpage allows student to choose degree program for study plan
DesiredCoursesSelectionPage	Boundary	This webpage allows student to choose Desired courses for study plan
SemesterSlectionPage	Boundary	This webpage allows student to specify number of semesters in study plan, all preferred courses, courses that were already taken, and other constraints
StudyPlanOutputPage	Boundary	This webpage shows student study plan if it was found
Course	Entity	Course information: prefix, number, title, credit, prerequisites, corequisites, oncampus, online.
Semester	Entity	Semester information: year, name
Course_group	Entity	Course group information: title, correspongding courses
Simple_requirement	Entity	Simple requirement information: title, required course number, corresponding course group
Requirement	Entity	Requirement information: title, corresponding simple requirement
Degree	Entity	Degree information: title, corresponding requirement.
CourseController	Controller	This component implements all CRUD operations over courses.
SemesterController	Controller	This component implements all CRUD operations over semesters.
CourseGroupController	Controller	This component implements all CRUD operations over course groups.
SimpleRequirementController	Controller	This component implements all CRUD operations over simple requirements.
RequirementController	Controller	This component implements all CRUD operations over requirements.
DegreeController	Controller	This component implements all CRUD operations over degree programs
StudyPlanController	Controller	This component performs prerequisites and corequisites checker and constraint solver, then output final study plan .

4.1.2.2 Backend class diagram

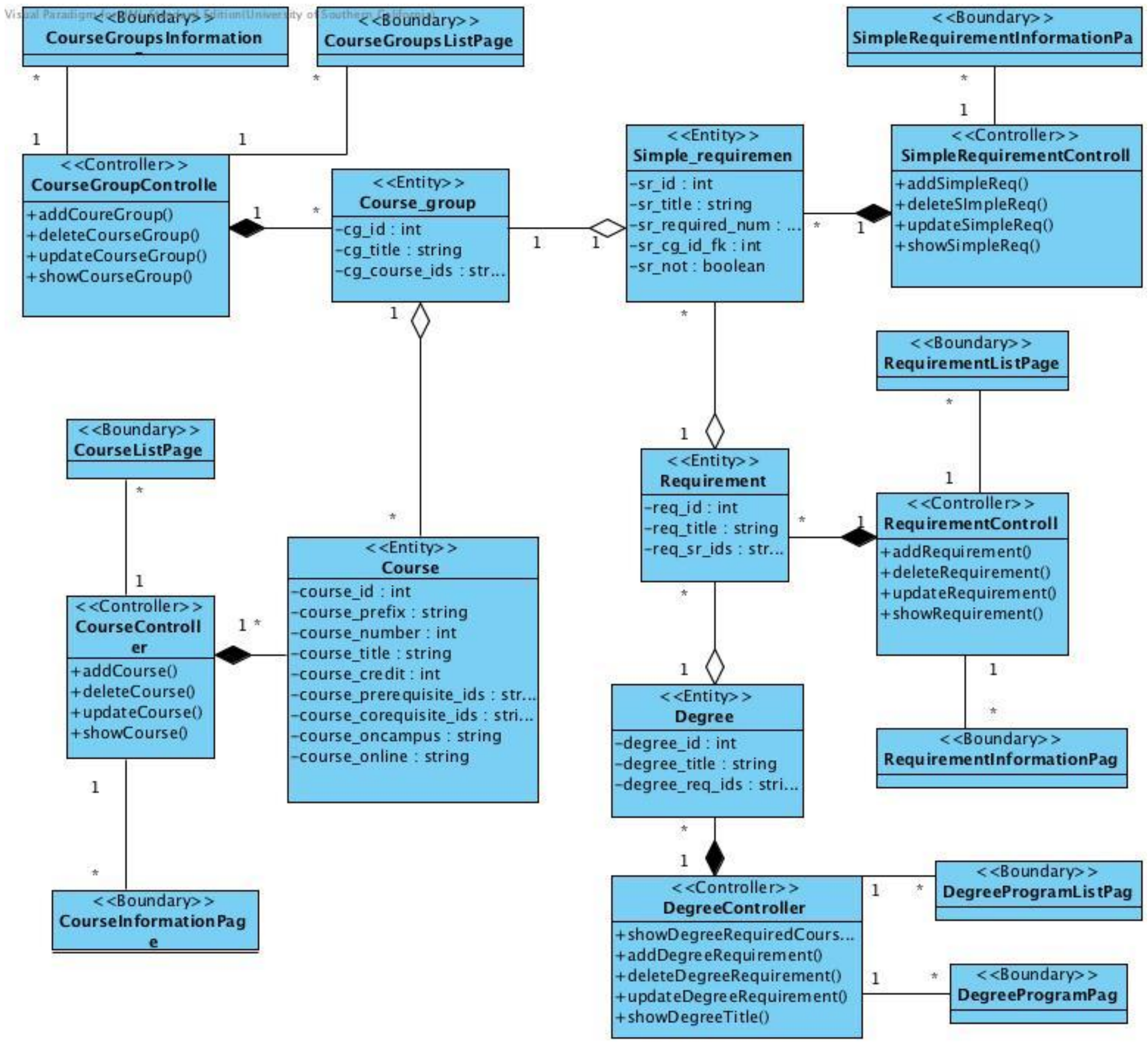
Visual Paradigm for UML Standard Edition(University of Southern California)
<<Boundary>> CourseGroupsInformation
<<Boundary>> CourseGroupsListPage
<<Boundary>> SimpleRequirementInformationPa
<<Controller>> CourseGroupControlle
+addCoureGroup()
+deleteCourseGroup()
+updateCourseGroup()
+showCourseGroup()
<<Entity>> Course_group
-cg_id : int
-cg_title : string
-cg_course_ids : str...
<<Entity>> Simple_requiremen
-sr_id : int
-sr_title : string
-sr_required_num : ...
-sr_cg_id_fk : int
-sr_not : boolean
<<Controller>> SimpleRequirementControll
+addSimpleReq()
+deleteSImpleReq()
+updateSimpleReq()
+showSimpleReq()
<<Boundary>> RequirementListPage
<<Boundary>> CourseListPage
<<Entity>> Requirement
-req_id : int
-req_title : string
-req_sr_ids : str...
<<Controller>> RequirementControll
+addRequirement()
+deleteRequirement()
+updateRequirement()
+showRequirement()
<<Controller>> CourseController
+addCourse()
+deleteCourse()
+updateCourse()
+showCourse()
<<Entity>> Course
-course_id : int
-course_prefix : string
-course_number : int
-course_title : string
-course_credit : int
-course_prerequisite_ids : str...
-course_corequisite_ids : stri...
-course_oncampus : string
-course_online : string
<<Entity>> Degree
-degree_id : int
-degree_title : string
-degree_req_ids : stri...
<<Boundary>> RequirementInformationPag
<<Boundary>> CourseInformationPage
<<Controller>> DegreeController
+showDegreeRequiredCours...
+addDegreeRequirement()
+deleteDegreeRequirement()
+updateDegreeRequirement()
+showDegreeTitle()
<<Boundary>> DegreeProgramListPag
<<Boundary>> DegreeProgramPag

Figure 8: Backend Class Diagram`

Table 41: Backend Class Description

Class	Type	Description
CourseListPage	Boundary	On this webpage course director can see all courses created in the system and choose create/edit/delete operation.
CourseInformationPage	Boundary	On this webpage course director can update or create course information.
CourseGroupListPage	Boundary	On this webpage course director can see all

		course groups created in the system and choose add/edit/delete operation.
CourseGroupInformationPage	Boundary	On this webpage course director can update or create course group information.
SimpleRequirementInformationPage	Boundary	On this webpage course director can update or create simple requirement information.
RequirementListPage	Boundary	On this webpage course director can see all requirements created in the system and choose add/edit/delete operation.
RequirementInformationPage	Boundary	On this webpage course director can update or create requirement information.
DegreeProgramListPage	Boundary	On this webpage course director can see all degree programs created in the system and choose add/edit/delete operation.
DegreeProgramPage	Boundary	On this webpage course director can update or create degree program information.
SemesterSelectionPage	Boundary	On this webpage course director can choose semester to get information of corresponding degree program.

4.1.3 Process Realization

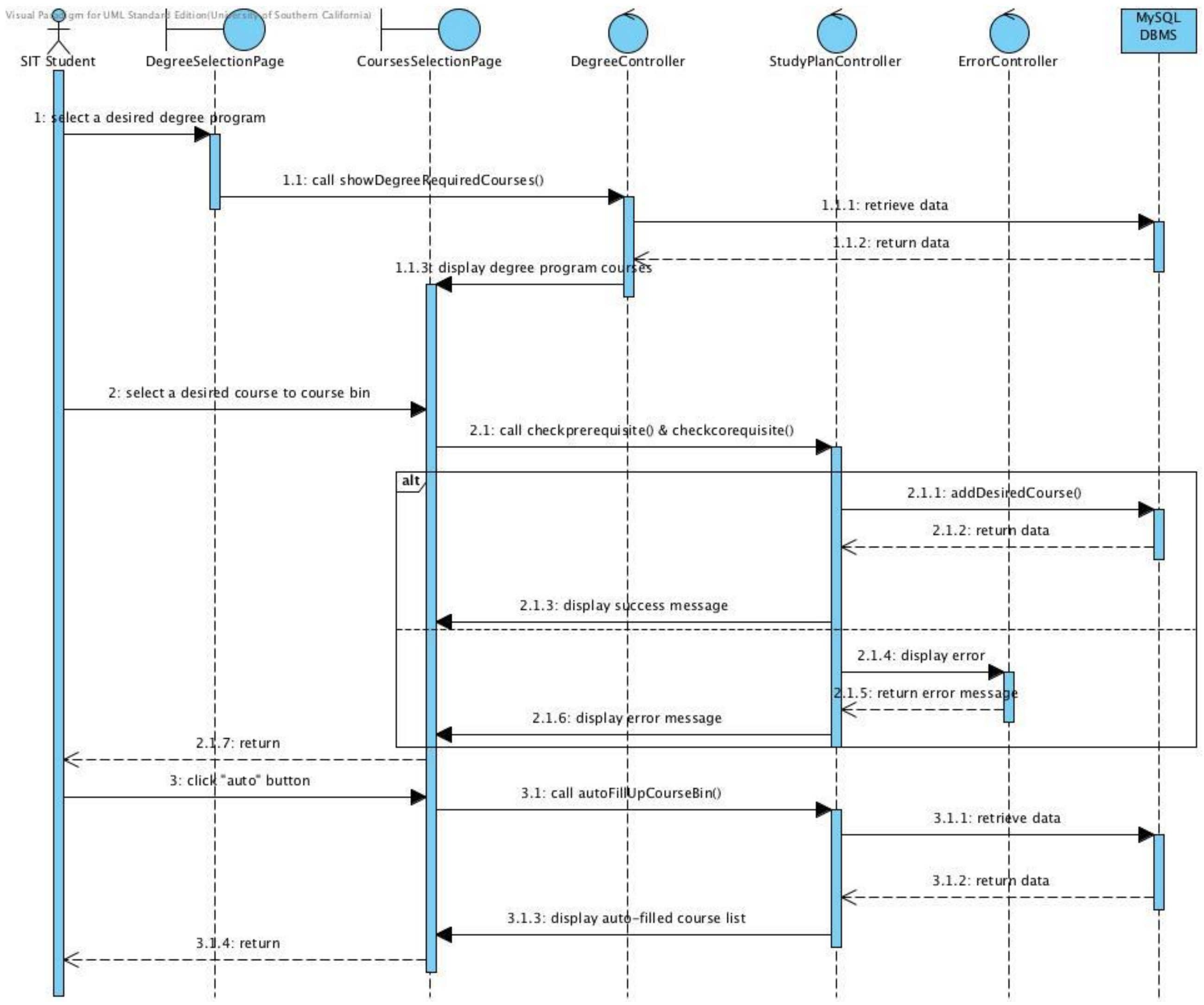
Visual Paradigm for UML Standard Edition(University of Southern California)
SIT Student
DegreeSelectionPage
CoursesSelectionPage
DegreeController
StudyPlanController
ErrorController
MySQL DBMS
1: select a desired degree program
1.1: call showDegreeRequiredCourses()
1.1.1: retrieve data
1.1.2: return data
1.1.3: display degree program courses
2: select a desired course to course bin
2.1: call checkprerequisite() & checkcorequisite()
alt
2.1.1: addDesiredCourse()
2.1.2: return data
2.1.3: display success message
2.1.4: display error
2.1.5: return error message
2.1.6: display error message
2.1.7: return
3: click "auto" button
3.1: call autoFillUpCourseBin()
3.1.1: retrieve data
3.1.2: return data
3.1.3: display auto-filled course list
3.1.4: return

Figure 9: Select Desired Course Sequence Diagram

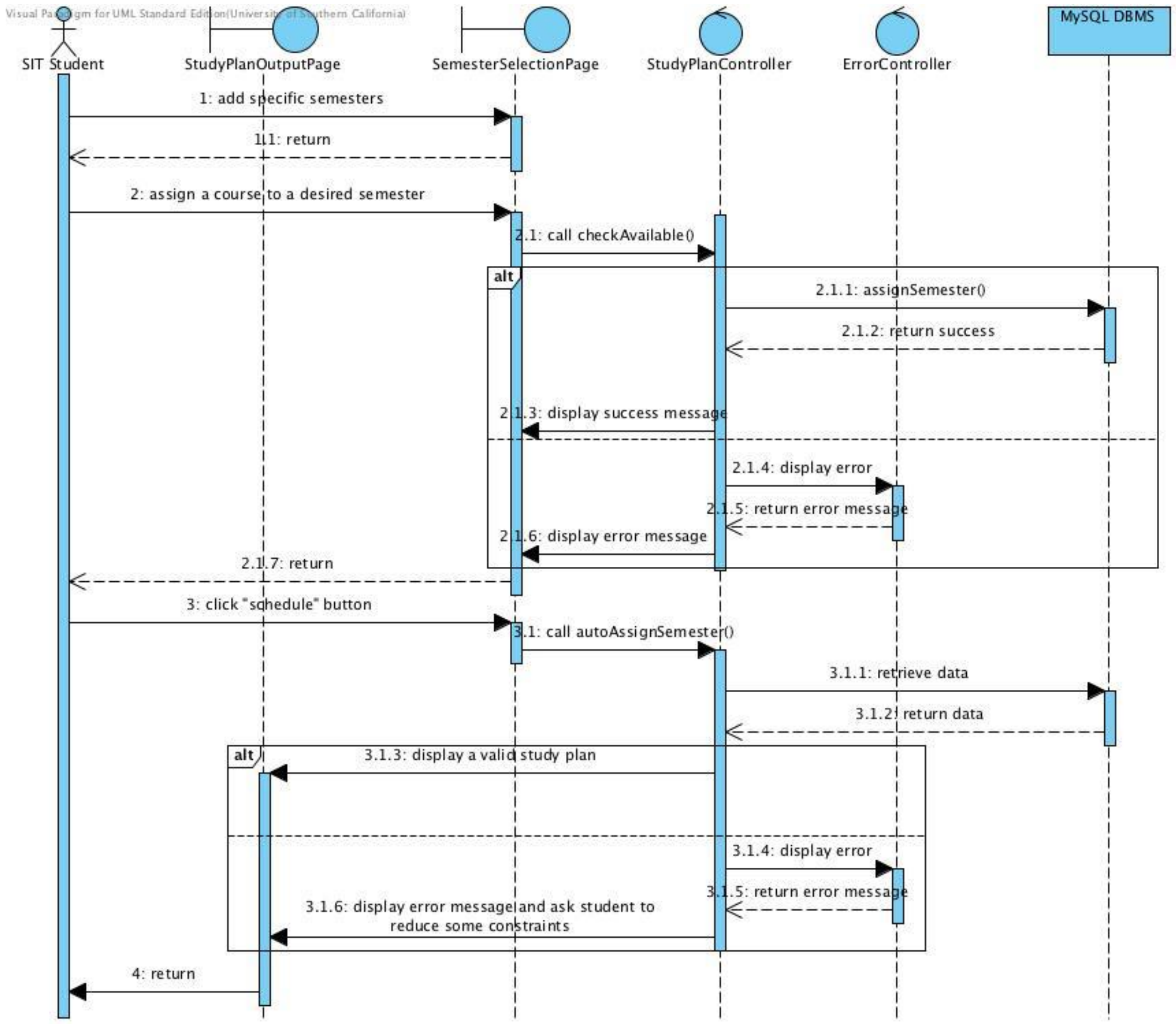
Visual Paradigm for UML Standard Edition(University of Southern California)
SIT Student
StudyPlanOutputPage
SemesterSelectionPage
StudyPlanController
ErrorController
MySQL DBMS
1: add specific semesters
1.1: return
2: assign a course to a desired semester
2.1: call checkAvailable()
alt
2.1.1: assignSemester()
2.1.2: return success
2.1.3: display success message
2.1.4: display error
2.1.5: return error message
2.1.6: display error message
2.1.7: return
3: click "schedule" button
3.1: call autoAssignSemester()
3.1.1: retrieve data
3.1.2: return data
alt
3.1.3: display a valid study plan
3.1.4: display error
3.1.5: return error message
3.1.6: display error message and ask student to reduce some constraints
4: return

Figure 10: Assign Courses to Semesters Sequence Diagram

4.2 Design Rationale

In order to make Student Scheduling System used by Stevens University students and staff, it must be accessible via the Internet for all users. Therefore, Student Scheduling System is designed as a web application.

In database design, to make CRUD of degree program more efficient and convenient, we construct the degree course DAG into the database. To make the study plan construction even more efficient, we store frequently used study plan in the database.

To make system more flexible and open to future changes three-tier architecture pattern was chosen. This pattern allows us to separate three layers of the system:

- User interface (web pages)
- Business logic (controls that implement study plan construction and other use cases)
- Data layer (date base, which store persistent information).

In order to make application simpler and more service oriented we chose REST architecture style for interaction with the system. Play framework for java was chosen as one of the most convenient frameworks for web application development that supports REST style. Another reason to choose Play framework is that it already has secure module, which enable us to set up basic authentication and authorization management to our application.

For persistence layer we chose MySQL DBMS because it's one of the most wide-used open source DBMS. Moreover, it is easy to integrate it with Play framework; they are absolutely compatible.