Course Registration Case Study

Table of Contents

Case Study Background

Course registration at the local university is currently done by hand. Students fill out forms that contain their course selections and return the forms to the registrar. Clerks then enter the selections into a database and a process is executed to create student schedules. The registration process takes from one to two weeks to complete.

The university decided to investigate the use of an online registration system. This system would be used by professors to indicate the courses they would teach, by students to select courses, and by the registrar to complete the registration process.

Course Registration System Problem Statement

At the beginning of each semester students may request a course catalogue containing a list of course offerings for the semester. Information about each course, such as professor, department, and prerequisites will be included to help students make informed decisions.

The new on-line registration system will allow students to select four course offerings for the coming semester. In addition, each student will indicate two alternative choices in case a course offering becomes filled or canceled. No course offering will have more than ten students. No course offering will have fewer than three students. A course offering with fewer than three students will be canceled. Once the registration process is completed for a student, the registration system sends information to the billing system, so the student can be billed for the semester.

Professors must be able to access the on-line system to indicate which courses they will be teaching. They will also need to see which students signed up for their course offering.

For each semester, there is a period of time that students can change their schedules. Students must be able to access the on-line system during this time to add or drop courses. The billing system will credit all students for courses dropped during this period of time.

The Role of Tools

A tool best supports any software development method. This lecture uses the tool Rational Unified Process methodology. Rational is organized around the architectural views - use case, logical, component and deployment. This case study will map the steps of the process into the views contained in the tool.

Project Summary

This system will have a short inception phase during which prototyping is used to select the database. The use case diagrams are started in the inception phase and matured in the elaboration phase. By the end of the elaboration phase, an architectural iteration is complete. The system is evolved in the construction phase in two iterations. The process components of requirements analysis, design, implementation and test are used in all phases of the project lifecycle.

The Inception Phase 1

Business Goals and Needs

The first question to address is the need for a new registration system. Does the University have the resources needed to design and implement the new system? In addition to the assessment of need for the system, the risks posed by the new system are elaborated. In the case of an on-line registration system, one of the major risks is the ability to store the information in a manner that is easily and quickly accessible by all.

For the purposes of this case study it was decided that the new system should be built. Prototypes were completed to address the database risks.

Definition of Actors

The following actors were defined for the problem:

- Student--someone who is registered to take courses at the University.
- Professor--someone who is licensed to teach at the University.
- Registrar--someone who is responsible for the maintenance of the Registration System.
- Billing System--external system that bills students each semester.

Definition of Use Cases

The following use cases were elaborated for each actor:

- Student
    - Register for courses.
- Professor
    - Select courses to teach.
    - Request course-offering roster.
- Registrar
    - Generate course catalogue.
    - Maintain professor information.
    - Maintain student information.
    - Maintain curriculum.

Drawing a Use Case Diagram in Rational Rose

The use case diagram is contained within a class diagram in the use case view of the tool. Actors are shown as stickmen and use cases are shown as ovals. The use case diagram is shown in Figure 1.

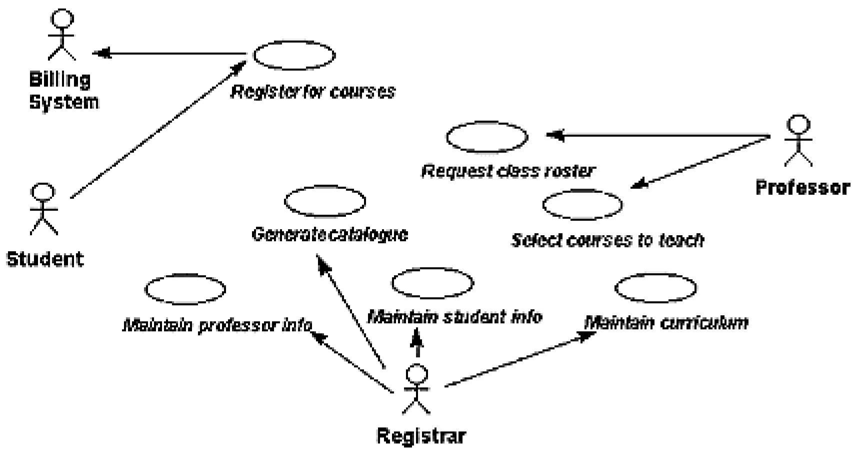


Figure 1 Use Case Diagram

A brief description is created for each use case. The brief description is entered in the Documentation field of the use case specification in the tool. The brief description of each use case follows:

- Register for courses
    - The use case is started by the student. It provides the capability to create, review, modify, and delete a course schedule for a specified semester. All pertinent billing information is sent to the Billing System. The Use-Case Specification example document is in Appendix A Register for Course Use Case.
- Request class roster
    - This use case is started by the professor. It provides the capability to request a printed list of all students assigned to a specified course offering.
- Select courses to teach
    - This use case is started by the professor. It provides the capability to select, review, modify, and delete a list of courses to teach for a specified semester. The Use-Case Specification example document is in Appendix B Select Course to Teach Use Case.
- Maintain professor information
    - This use case is started by the registrar. It provides the capability to create, review, modify, and delete professor information. The Use-Case Specification

    example document is in Appendix C Maintain Professor Information Use Case.

- Maintain student information
    - This use case is started by the registrar. It provides the capability to create, review, modify, and delete student information. The Use-Case Specification example document is in Appendix D Maintain Student Information Use Case.
- Maintain curriculum
    - This use case is started by the registrar. It provides the capability to create, review, modify, and delete a list of course offerings for a given semester.
- Generate catalogue
    - This use case is started by the registrar. It provides the capability to generate a catalogue containing a list of course offerings for a specified semester.

During Inception, the flow of events (including any identified alternate flows) for the most important use cases is documented. The flow of events for the Register for Courses use case is shown below.

Flow of Events: Register for Courses Use Case

This use case begins when the student enters the student id number. The system verifies that the student id number is valid and prompts the student to select the current semester or a future semester. The student enters the desired semester. The system prompts the student to select the desired activity:

- Create a schedule.
- Review a schedule.
- Change a schedule:
    - Delete a course.
    - Add a course.

The student indicates that the activity is complete. The system will print the student schedule and notify the student that registration is complete. The system sends billing information for the student to the billing system for processing.

Alternate flow

If an invalid id number is entered, the system will not allow access to the registration system.

If an attempt is made to create a schedule for a semester where a schedule already exists, the system will prompt for another choice to be made.

Create a Schedule

The student enters 4 primary course offering numbers and 2 alternate course offering numbers. The student then submits the request for courses. The system then:

1. Checks that prerequisites are satisfied for the requested course.
2. Adds the student to the course offering if the course offering is open.

Alternate flow

If a primary course offering is not available, the system will substitute an alternate course offering.

Review a Schedule

The student requests information on all course offerings in which the student is registered for a given semester. The system displays all courses for which the student is registered including course name, course number, course offering number, days of the week, time, location, and number of credit hours.

Change Schedule - Delete a Course

The student indicates which course offerings to delete. The system checks that the final date for changes has not been exceeded. The system deletes the student from the course offering. The system notifies the student that the request has been processed.

Change Schedule - Add a Course

The student indicates which course offerings to add. The system checks that the final date for changes has not been exceeded. The system then:

1. Verifies that the maximum course load for the student has not been exceeded.
2. Checks that prerequisites are satisfied for the requested course.
3. Adds the student to the course offering if the course offering is open.

The Elaboration Phase 2

During Elaboration, some of the most important and critical use cases are implemented. During this phase, the focus is good class structure and architecture.

Development of Scenarios

Each use case is a web of scenarios. Scenarios are documented using Sequence Diagrams. Objects are represented as vertical lines and messages between objects are shown as directed horizontal lines. Sequence diagrams are drawn in the Use Case View of the tool. The Sequence Diagram for the Add a Course scenario is shown in Figure 2.

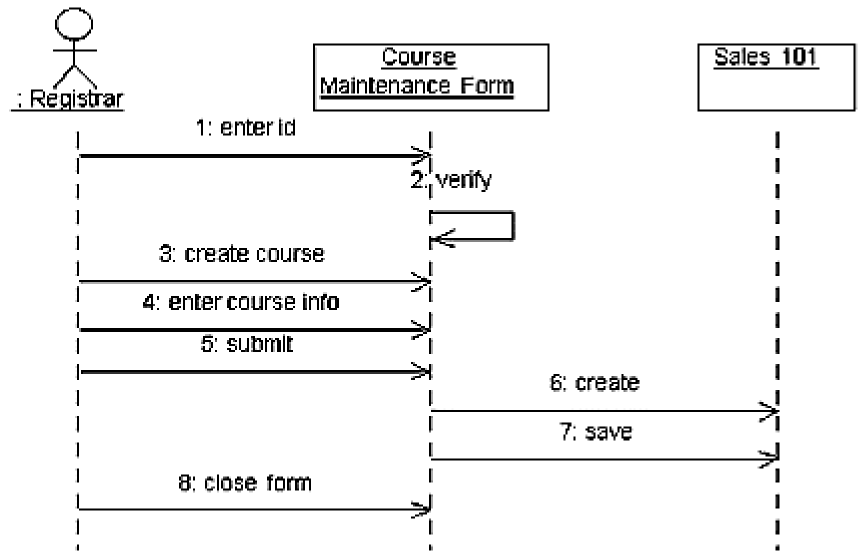


Figure 2 Sequence Diagram for the Add a Course Scenario

Creating "Real World" or "Business" Classes

Objects are discovered by examining the use cases and scenarios and grouped into classes. Each class should have a definition which states the purpose of the class. Packages are created to hold logical groups of classes. Classes and packages are drawn in the Logical View of the tool. The following packages and classes have been created for the registration system:

- People
  - StudentInfo--Information about the student actor needed by the registration system (for example, name, address, phone, idNumber, major, gradDate).
  - ProfessorInfo--Information about the professor actor needed by the registration system (for example, name, address, phone, idNumber, tenureStatus).
- UniversityArtifacts
  - Course--General information about selections for a semester (for example, name, description, creditHours).
  - CourseOffering--Specific information about selections for a semester (for example, daysOffered, timeOfDay, location).
  - StudentSchedule--Output report containing the list of registered course offerings generated when a student registers for a course.
  - CourseRoster--Output report containing the list of registered students for a specific course offering generated for a professor.
- Interfaces

- RegistrationForm--Form which provides the capability for a student to select registration options.
- Add/DropForm--Form which provides the capability for a student to modify a course schedule.
- CourseSelectionForm--Form which provides the capability for a professor to add/drop courses to teach.
- StudentMaintenanceForm--Form which provides the capability for the registrar to add/delete/modify student information.
- ProfessorMaintenanceForm--Form which provides the capability for the registrar to add/delete/modify professor information.
- CourseMaintenanceForm--Form which provides the capability for the registrar to add/delete/modify course and course offering information.

Class diagrams are created to graphically depict the packages and classes in the model. The Main class diagram typically contains only packages. Each package contains its own class diagrams. The Main class diagram for a package contains the public classes of the package (classes that communicate with classes in other packages). Other class diagrams are created as needed. Class diagrams are contained in the Logical View of the tool.

Use cases and scenarios are examined to determine the relationships needed by the system. Relationships between classes are created and displayed on selected class diagrams.

Attributes (structure) and operations (behavior) are added to the classes to carry out the functionality specified in the use cases.

Sequence diagrams are updated to show the allocation of objects to classes and the replacement of messages with operations.

Some class diagrams for the Registration System are shown in Figures 3 through 7. An updated sequence diagram is shown in Figure 8.

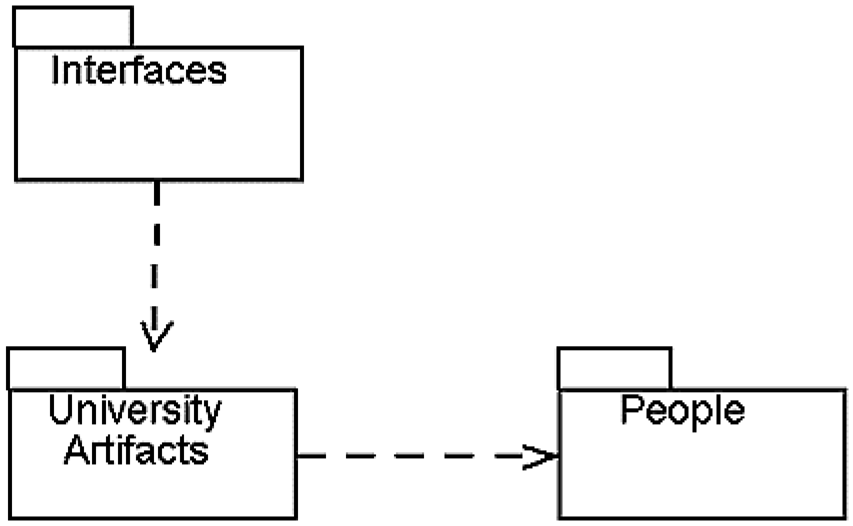


Figure 3 Main Class Diagram

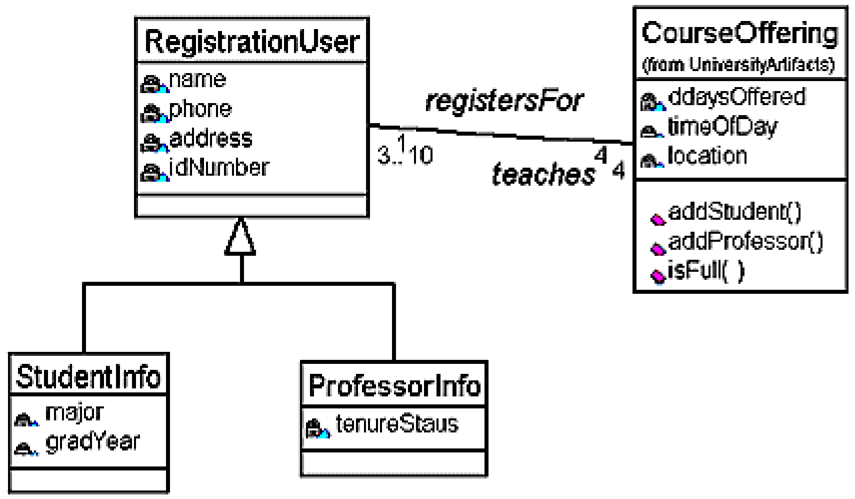


Figure 4 Main Class Diagram for the People Package

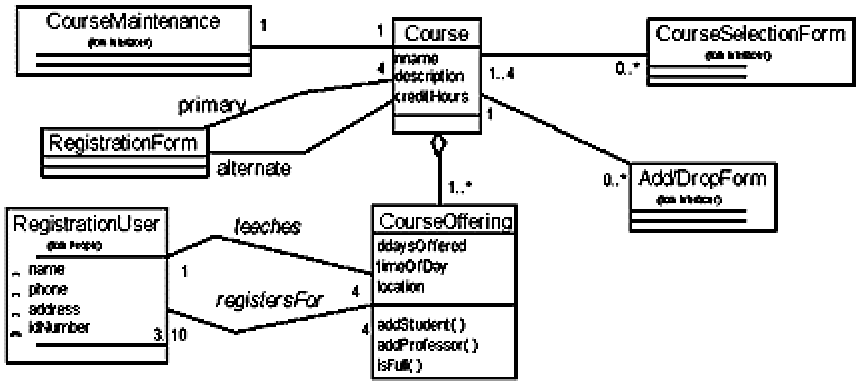


Figure 5 Main Class Diagram for the University Artifacts Package

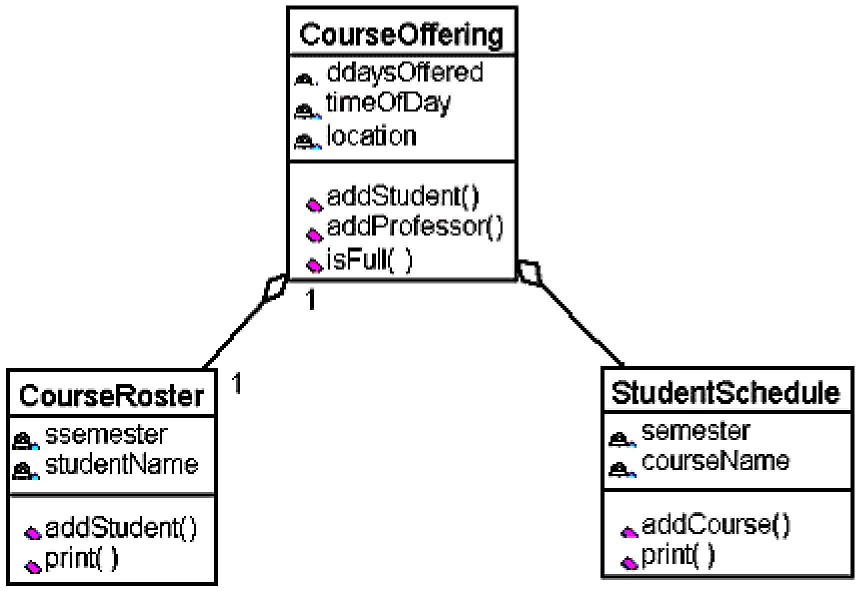


Figure 6 Course Reporting Class Diagram in the UniversityArtifacts Package

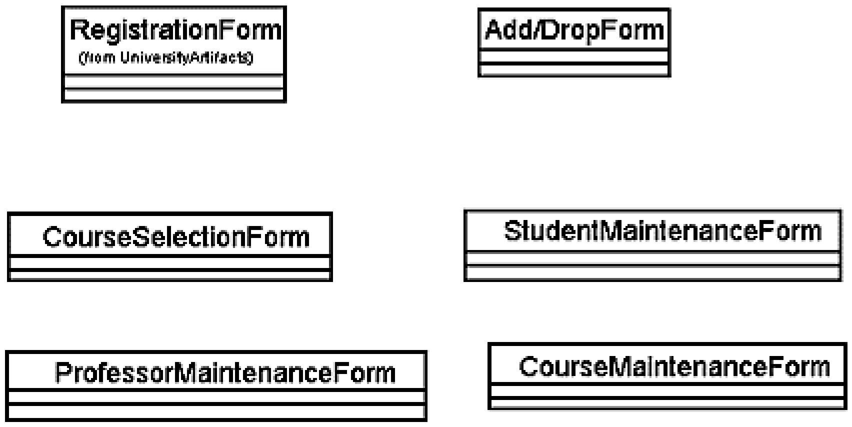


Figure 7 Main Class Diagram for the Interfaces Package

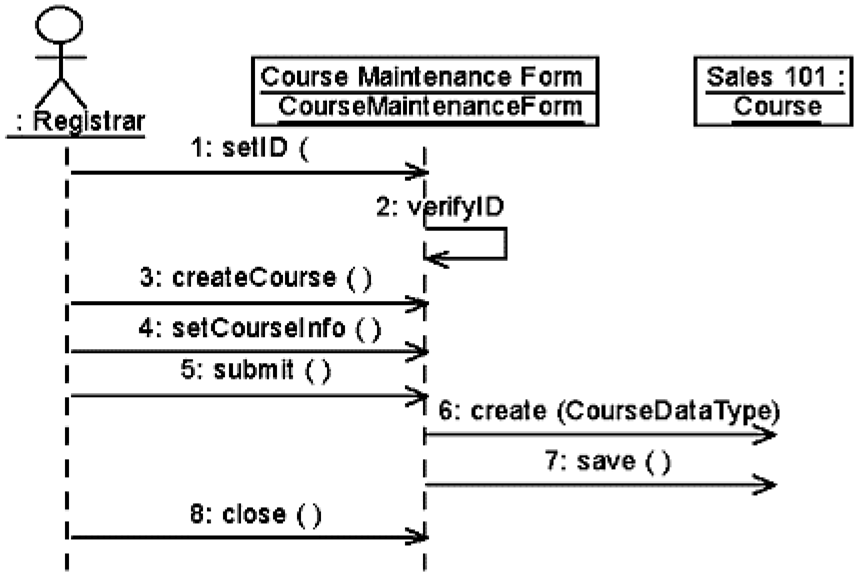


Figure 8 Updated Sequence Diagram

Software Architecture

As the elaboration phase of development continues, decisions concerning the architectural framework for the project are made. Scenarios are updated to show the interaction of the real world objects with the objects representing the architectural decisions. Packages and classes that carry out the architectural functionality are added to the logical view.

In the Course Registration system, the following architectural decisions were made:

- Containers and GUI classes to be used are in the MFC class library.

- A commercial relational database was chosen and classes to communicate with the database were created.
- A set of error classes were created to facilitate common error handling strategies.

The updated Main Class Diagram and an updated Sequence Diagram are shown in Figures 9 and 10.

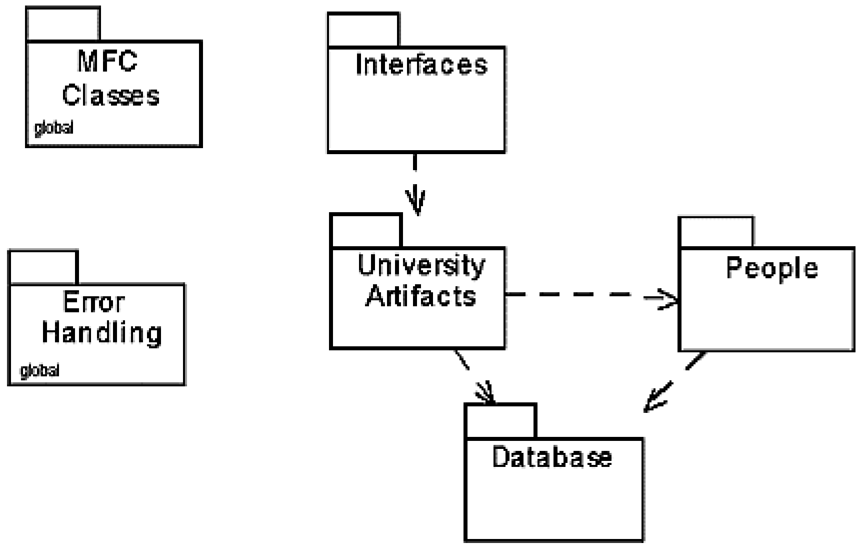


Figure 9 Main Class Diagram

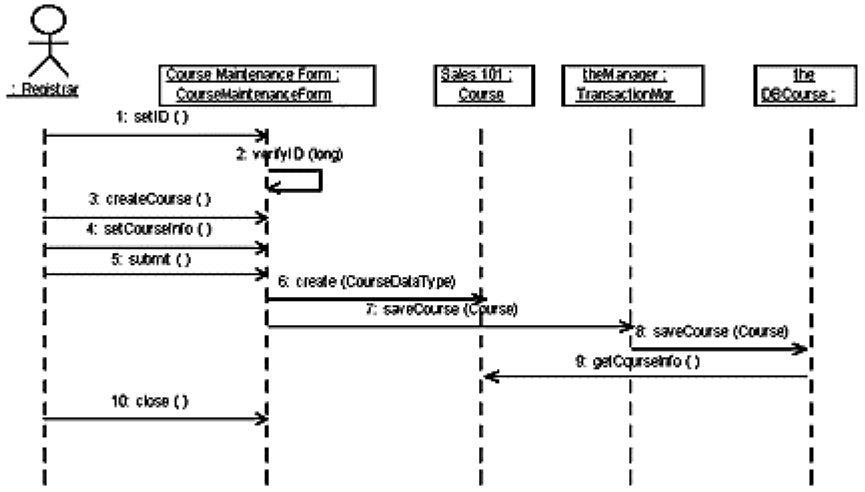


Figure 10 Updated Sequence Diagram

The next step is to implement a set of scenarios that address the major architectural issues. This is done to ensure early feedback and identification of problems. For this problem, the Maintain Curriculum Use Case was implemented since it addressed the major risk of this system--the database risk.

Iteration Planning

Another activity in the elaboration phase is the creation of the iteration plan. The goal of an iteration is to reduce risk in the system while incrementally building the final product. Use cases and scenarios are examined and prioritized to create the initial project plan. As each iteration is completed, risks are re-evaluated and the project plan is updated as needed.

For the Course Registration system the iteration plan is:

- Iteration 1
    - Maintain curriculum.
- Iteration 2
    - Maintain student info.
    - Maintain professor info.
    - Select courses to teach.
    - Generate catalogue.
- Iteration 3
    - Register for courses.
    - Request class roster.

The Construction Phase 3

Construction Activities

During Construction, all remaining scenarios will be specified and implemented. At this time, many of the secondary scenarios are addressed.

Building an Iteration

This case study concentrates on the "Add a Course" scenario which is shown in Figure 14. During this phase of development, the classes that participate in the iteration are designed and implemented. Class diagrams are created to show the focus of the iteration.

For the Course Registration problem, the following design decisions are made:

- Controller class--CurriculumManager added. This class knows the business rules associated with the management of a curriculum.
- Scenario diagrams are updated to show new interactions with the CurriculumManager.

    - Some interactions between objects are deleted due to the addition of the controller.

- Data types and signatures are provided for all attributes and operations.
- Association navigation is designed.
- Associations are changed into dependency relationships where appropriate.

An updated Sequence diagram showing the interaction with the added controller class is shown in Figure 11.

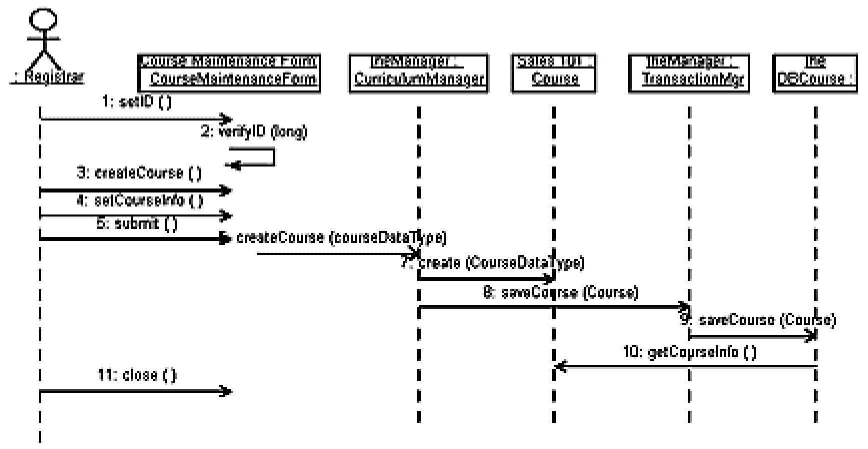
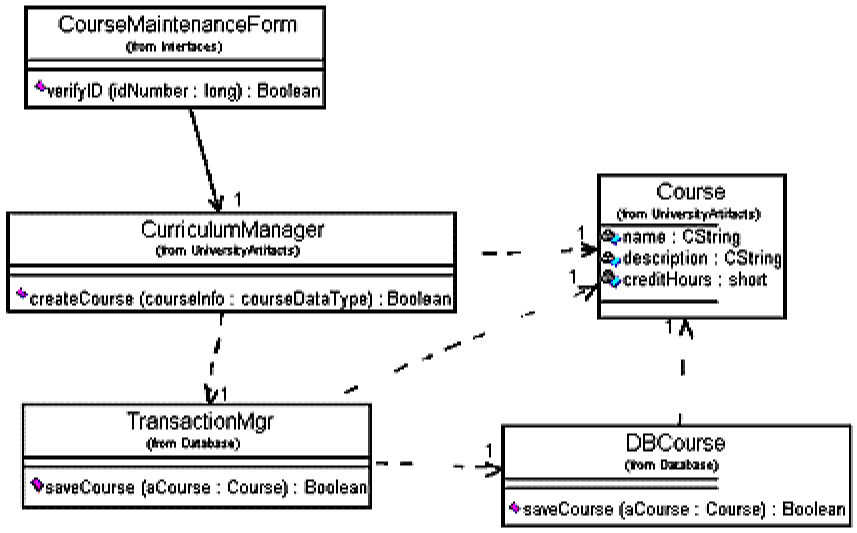


Figure 11 Updated Sequence Diagram

Add a Course

A package called Iteration 1 is added to the logical view of the model. Class diagrams showing the classes in the iteration are added to the package. A class diagram showing the design decisions made for the "Add a Course" scenario is shown in Figure 12.



CourseMaintenanceForm
(from Interfaces)
verifyID (idNumber : long) : Boolean
1
CurriculumManager
(from UniversityArtifacts)
createCourse (courseInfo : courseDataType) : Boolean
Course
(from UniversityArtifacts)
name : CString
description : CString
creditHours : short
1
1
1
1
TransactionMgr
(from Database)
saveCourse (aCourse : Course) : Boolean
1
DBCourse
(from Database)
saveCourse (aCourse : Course) : Boolean

Figure 12 Class Diagram "Add a Course"

The code for the iteration is completed and the iteration is tested and documented. The completed iteration is integrated with any previous iterations.

The Transition Phase 4

The system was successfully transitioned to the University community in two releases--beta and the final system. During the beta period, bugs were discovered, reported and fixed by the development staff. After using the beta version of the system, professors added the requirement to view a class roster on-line. This requirement was successfully implemented and available in the final release of the system. Students and professors were pleased with the time savings provided by the paperless system.

Due to the success of the Registration System it was decided that another version of the system should be developed to provide an on-line catalogue of course offerings. Budgets and staff were approved and the process began again.

APPENDIX A

Course Registration System

Use-Case Specification

Register for Courses Use Case

Version: Draft

Revision History

Date	Version	Description	Author
21/Dec/02	Draft	Draft Version	The Author

Table of Contents

Register for Courses Use Case

4. Brief Description

This use case allows a Student to register for course offerings in the current semester. The Student can also modify or delete course selections if changes are made within the add/drop period at the beginning of the semester. The Course Catalog System provides a list of all the course offerings for the current semester.

The main actor of this use case is the Student. The Course Catalog System is an actor within the use case.

5. Flow of Events

The use case begins when the Student selects the "maintain schedule" activity from the Main Form.

1. Basic Flow – Create a Schedule

The Student selects "create schedule."

The system displays a blank schedule form.

The system retrieves a list of available course offerings from the Course Catalog System.

The Student selects 4 primary course offerings and 2 alternate course offerings from the list of available offerings. Once the selections are complete the Student selects "submit."

Courses are added for each selected course offering.

The system saves the schedule.

2. Alternative Flows

1. *Modify a Schedule*

TBD.

2. *Delete a Schedule*

TBD.

3. *Course Catalog System Unavailable*

If, the system is unable to communicate with the Course Catalog System after a specified number of tries, the system will display an error message to the Student. The Student acknowledges the error message and the use case terminates.

3. Special Requirements

Special requirements will be determined during the next iteration.

4. Pre-Conditions

Login

Before this use case begins the Student has logged onto the system.

5. Post-Conditions

Post-conditions will be determined during the next iteration.

6. Extension Points

Extension points of the business use case will be identified during the Elaboration Phase.

APPENDIX B

Course Registration System

Use-Case Specification

Select Courses to Teach Use Case

Version: Draft

Revision History

Date	Version	Description	Author
21/Dec/02	Draft	Draft Version	Author name

Table of Contents

Select Courses to Teach Use Case

6. Brief Description

This use case allows a professor to select the course offerings (date- and time-specific courses will be given) from the course catalog for the courses that he/she is eligible for and wishes to teach in the upcoming semester.

The actor starting this use case is the Professor. The Course Catalog System is an actor within the use case.

7. Flow of Events

The use case begins when the professor selects the "select courses to teach" activity from the Main Form.

1. Basic Flow – Select Courses to Teach

1. The system retrieves and displays the list of course offerings the professor is eligible to teach for the current semester. The system also retrieves and displays the list of courses the professor has previously selected to teach.

2. The professor selects and/or de-selects the course offerings that he/she wishes to teach for the upcoming semester.

3. The system removes the professor from teaching the de-selected course offerings.

4. The system verifies that the selected offerings do not conflict (i.e., have the same dates and times) with each other or any offerings the professor has previously signed up to teach. If there is no conflict, the system updates the course offering information for each offering the professor selects.

1. Alternative Flows

Issues: Add flows to deal with following conditions:

- Handling of course scheduling conflicts
- Registration period is ended
- Professor is not eligible to teach the course.

2. Special Requirements

Special requirements will be determined during the next iteration.

3. Pre-Conditions

1. Login

Before this use case begins the Professor has logged onto the system.

4. Post-Conditions

Post-conditions will be determined during the next iteration.

5. Extension Points

1. Extension points of the business use case will be identified during the Elaboration Phase.

APPENDIX C

Course Registration System

Use-Case Specification

Maintain Professor Information Use Case

Version: Draft

Revision History

Date	Version	Description	Author
21/Dec/02	Draft	Draft version.	Author name

Table of Contents

Maintain Professor Information Use Case

1. Brief Description

This use case allows the Registrar to maintain professor information in the registration system. This includes adding, modifying, and deleting professors from the system.

The actor of this use case is the Registrar.

2. **Flow of Events**

The use case begins when the Registrar selects the "maintain professor" activity from the Main Form.

1. **Basic Flow – Add a Professor**

The Registrar selects "add a professor."

The system displays a blank professor form.

The Registrar enters the following information for the professor: name, date of birth, social security number, status, and department.

The system validates the data to insure the proper data format and searches for an existing professor with the specified name. If the data is valid the system creates a new professor and assigns a unique system-generated id number. This number is displayed, so it can be used for subsequent uses of the system.

Steps 2-4 are repeated for each professor added to the system. When the Registrar is finished adding professors to the system the use case ends.

2. **Alternative Flows**

1. *Modify and Delete a Professor*

2. TBD

3. **Special Requirements**

Special requirements will be determined during the next iteration.

4. Pre-Conditions

Log In

Before this use case begins the Registrar has logged onto the system.

5. **Post-Conditions**

Post-conditions will be determined during the next iteration.

6. **Extension Points**

Extension points of the business use case will be identified during the Elaboration Phase.

Course Registration System

Use-Case Specification

Maintain Student Information Use Case

Version: Draft

Revision History

Date	Version	Description	Author
21/Dec/02	Draft	Draft version	Author Name

Table of Contents

Maintain Student Information Use Case

8. Brief Description

This use case allows the Registrar to maintain student information in the registration system. This includes adding, modifying, and deleting students from the system.

The actor for this use case is the Registrar.

9. **Flow of Events**

The use case begins when the Registrar selects the "maintain student" activity from the Main Form.

1. **Basic Flow – Add Student**

The Registrar selects "add student."

The system displays a blank student form.

The Registrar enters the following information for the student: name, date of birth, social security number, status, and graduation date.

The system validates the data to insure the proper format and searches for an existing student with the specified name. If the data is valid the system creates a new student and assigns a unique system-generated id number.

Steps 2-4 are repeated for each student added to the system. When the Registrar is finished adding students to the system the use case ends.

2. **Alternative Flows**

1. *Modify a Student*

Issue: Must ensure the flows for modifying and deleting students are similar to the flows for modifying and deleting professors.

2. *Delete a Student*

Issue: Must ensure the flows for modifying and deleting students are similar to the flows for modifying and deleting professors.

3. **Special Requirements**

Special requirements will be determined during the next iteration.

4. **Pre-Conditions**

Log In

Before this use case begins the Registrar has logged onto the system.

5. **Post-Conditions**

Post-conditions will be determined during the next iteration.

6. Extension Points

Extension points of the business use case will be identified during the Elaboration Phase.