Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This Software Architecture Document provides an architectural overview of the C-Registration System. The C-Registration System is being developed by Wylie College to support online course registration.

This Document has been generated directly from the C-Registration Analysis & Design Model implemented in Rose. The majority of the sections have been extracted from the Rose Model using SoDA and the Software Architecture Document template.

1.3 Definitions, Acronyms and Abbreviations

See the Glossary [4].

1.4 References

Applicable references are:

1. Course Billing Interface Specification, WC93332, 1985, Wylie College Press.
2. Course Catalog Database Specification, WC93422, 1985, Wylie College Press.
3. Vision Document of the C-Registration System, WyIT387, V1.0, 1998, Wylie College IT.
4. Glossary for the C-Registration System, WyIT406, V2.0, 1999, Wylie College IT.
5. Use Case Spec - Close Registration, WyIT403, V2.0, 1999, Wylie College IT.
6. Use Case Spec – Login, WyIT401, V2.0, 1999, Wylie College IT.
7. Use Case Spec - Maintain Professor Info, WyIT407, Version 2.0, 1999, Wylie College IT.
8. Use Case Spec - Register for Courses, WyIT402, Version 2.0, 1999, Wylie College IT.
9. Use Case Spec - Select Courses to Teach, WyIT405, Version 2.0, 1999, Wylie College IT.
10. Use Case Spec - Maintain Student Info, WyIT408, Version 2.0, 1999, Wylie College IT.
11. Use Case Spec - Submit Grades, WyIT409, Version 2.0, 1999, Wylie College IT.
12. Use Case Spec - View Report Card, WyIT410, Version 2.0, 1999, Wylie College IT.
13. Software Development Plan for the C-Registration System, WyIT418, V1.0, 1999, Wylie College IT.
14. E1 Iteration Plan, WyIT420, V1.0, 1999, Wylie College IT.
15. Supplementary Specification, WyIT400, V1.0, 1999, Wylie College, IT.

2. Architectural Representation

This document presents the architecture as a series of views; use case view, logical view, process view and deployment view. There is no separate implementation view described in this document. These are views on an underlying Unified Modeling Language (UML) model developed using Rational Rose.

3. Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

1. The existing legacy Course Catalog System at Wylie College must be accessed to retrieve all course information for the current semester. The C-Registration System must support the data formats and DBMS of the legacy Course Catalog System [2].
2. The existing legacy Billing System at Wylie College must be interfaced with to support billing of students. This interface is defined in the Course Billing Interface Specification [1].
3. All student, professor, and Registrar functionality must be available from both local campus PCs and remote PCs with internet dial up connections.
4. The C-Registration System must ensure complete protection of data from unauthorized access. All remote accesses are subject to user identification and password control.
5. The C-Registration System will be implemented as a client-server system. The client portion resides on PCs and the server portion must operate on the Wylie College UNIX Server. [3]
6. All performance and loading requirements, as stipulated in the Vision Document [3] and the Supplementary Specification [15], must be taken into consideration as the architecture is being developed.

4. Use-Case View

A description of the use-case view of the software architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

The C-Registration use cases are:

- Login

- Register for Courses

- Maintain Student Information

- Maintain Professor Information

- Select Courses to Teach

- Submit Grades

- View Report Card

- Close Registration.

These use cases are initiated by the student, professor, or the registrar actors. In addition, interaction with external actors; Course Catalog and Billing System occur.

4.1 Architecturally-Significant Use Cases

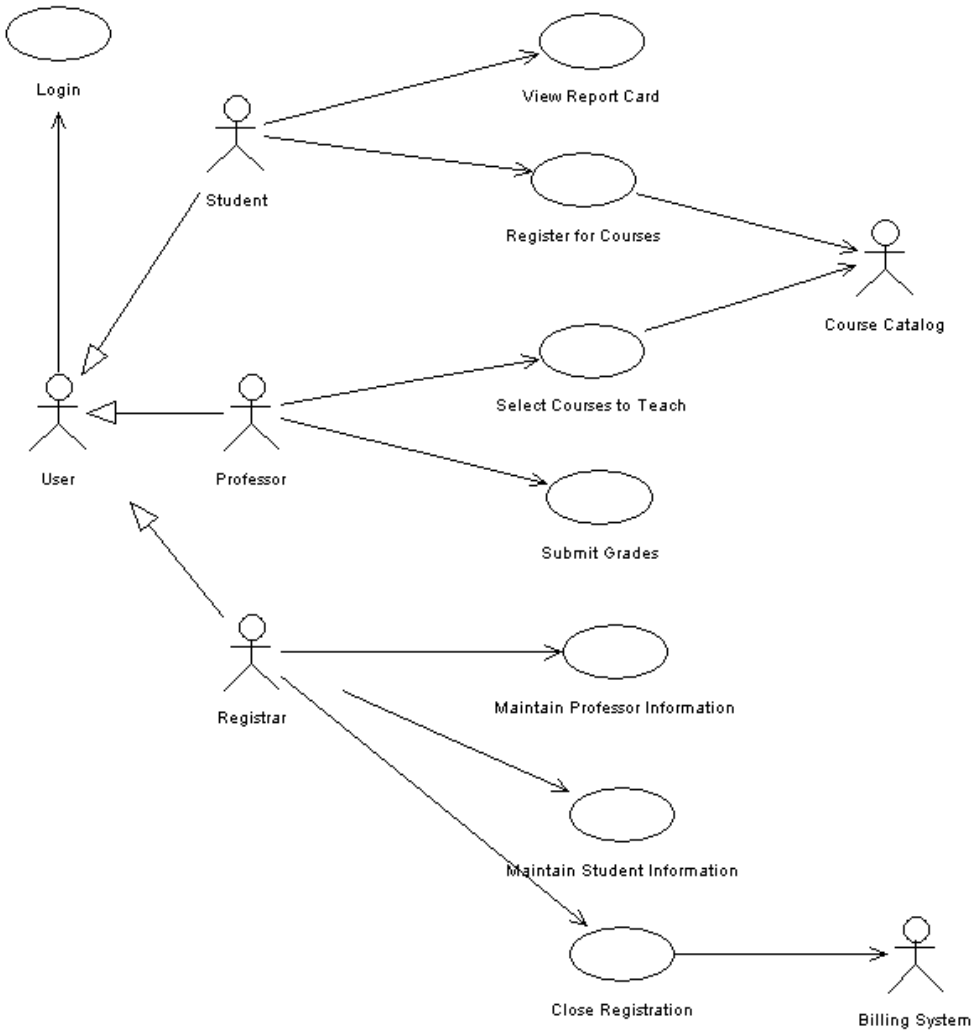


Diagram Name: Architecturally Significant Use-Cases

4.1.1 Close Registration

Brief Description:This use case allows a Registrar to close the registration process. Course offerings that do not have enough students are cancelled. Course offerings must have a minimum of three students in them. The billing system is notified for each student in each course offering that is

not cancelled, so the student can be billed for the course offering. The main actor of this use case is the Registrar. The Billing System is an actor involved within this use case.

4.1.2 Login

Brief Description:This use case describes how a user logs into the Course Registration System. The actors starting this use case are Student, Professor, and Registrar.

4.1.3 Maintain Professor Information

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.

4.1.4 Select Courses to Teach

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.

4.1.5 Register for Courses

Brief Description:This use case allows a student to register for courses 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 Billing System is notified of all registration updates. The Course Catalog 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.

4.1.6 View Report Card

Brief Description:This use case allows a student to view his/her report card for the previously completed semester. The student is the actor of this use case.

4.1.7 Submit Grades

Brief Description:This use case allows a professor to submit student grades for one or more classes completed in the previous semester. The actor in this use case is the Professor.

4.1.8 Maintain Student Information

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.

5. Logical View 

A description of the logical view of the architecture. Describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

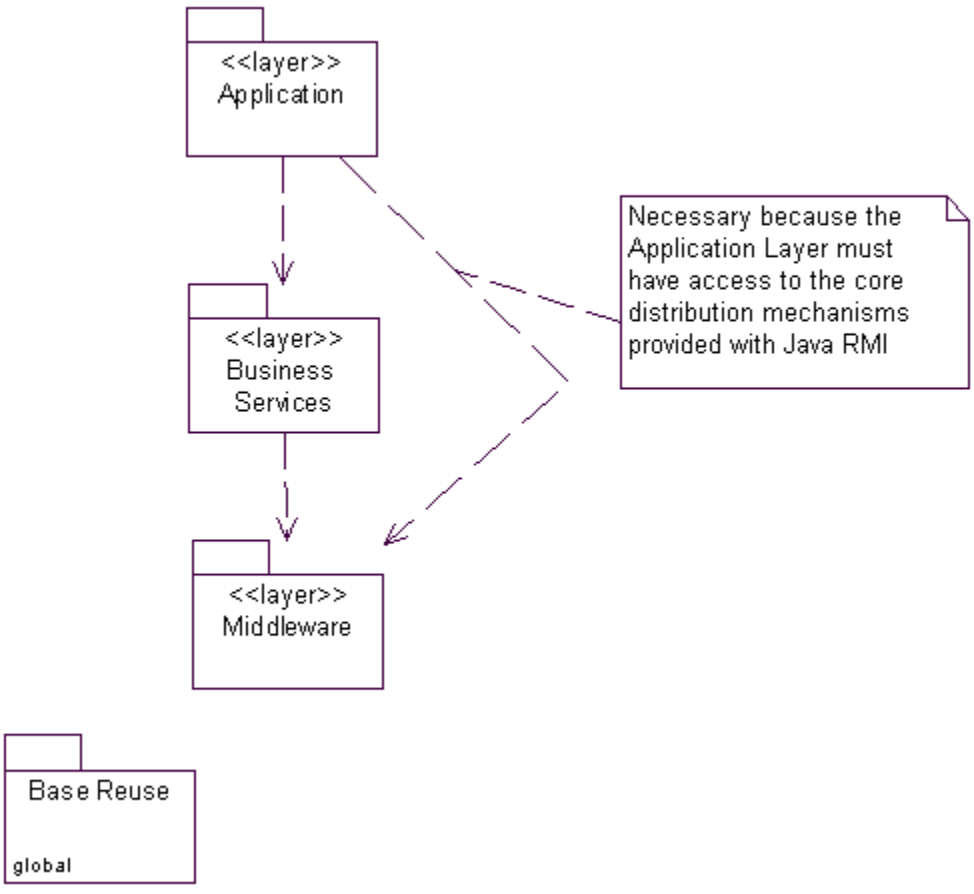
The logical view of the course registration system is comprised of the 3 main packages: User Interface, Business Services, and Business Objects.

The User Interface Package contains classes for each of the forms that the actors use to communicate with the System. Boundary classes exist to support login, maintaining of schedules, maintaining of professor info, selecting courses, submitting grades, maintaining student info, closing registration, and viewing report cards.

The Business Services Package contains control classes for interfacing with the billing system, controlling student registration, and managing the student evaluation.

The Business Objects Package includes entity classes for the university artifacts (i.e. course offering, schedule) and boundary classes for the interface with the Course Catalog System.

5.1 Architecture Overview – Package and Subsystem Layering



5.1.1 Application

layer

This application layer has all the boundary classes that represent the application screens that the user sees. This layer depends upon the Process Objects layer; that straddles the separation of the client from mid-tier.

5.1.2 Business Services

layer

The Business Services process layer has all the controller classes that represent the use case managers that drive the application behavior. This layer represents the client-to-mid-tier border. The Business Services layer depends upon the Process Objects layer; that straddles the separation of the client from mid-tier.

5.1.3 Middleware

layer

The Middleware layer supports access to Relational DBMS and OODBMS.

5.1.4 Base Reuse

The Base Reuse package includes classes to support list functions and patterns.

6. Process View 

A description of the process view of the architecture. Describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.

The Process Model illustrates the course registration classes organized as executable processes. Processes exist to support student registration, professor functions, registration closing, and access to the external Billing System and Course Catalog System.

6.1 Processes 

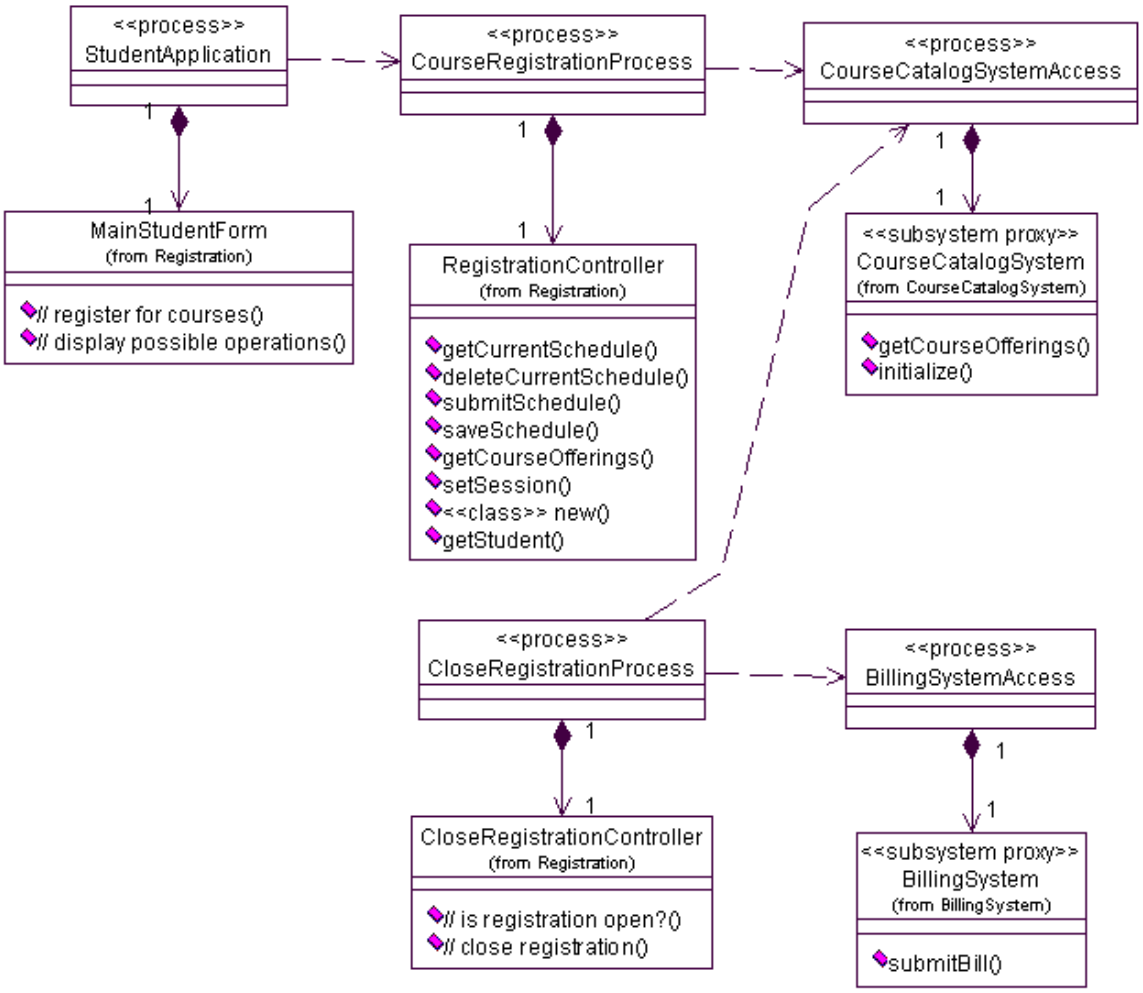


Diagram Name: Processes

6.1.1 CourseCatalogSystemAccess

This process manages access to the legacy Course Catalog System. It can be shared by multiple users registering for courses. This allows for a cache of recently retrieved courses and offerings to improve performance.

The separate threads within the CourseCatalog process, CourseCache and OfferingCache are used to asynchronously retrieve items from the legacy system.

Analysis Mechanisms:

- Legacy Interface

Requirements Traceability:

- Design Constraints: The system shall integrate with existing legacy system (course catalog database).

6.1.2 CourseCatalog

The unabbridged catalog of all courses and course offerings offered by the university including those from previous semesters.

This class acts as an adapter (see the Gamma pattern). It works to makes sure the CourseCatalogSystem can be accessed through the ICourseCatalog interface to the subsystem.

6.1.3 CourseRegistrationProcess

There is one instance of this process for each student that is currently registering for courses.

6.1.4 RegistrationController

This supports the use case allowing a student to register for courses 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.

Analysis Mechanisms:

- Distribution

6.1.5 StudentApplication

Manages the student functionality, including user interface processing and coordination with the business processes.

There is one instance of this process for each student that is currently registering for courses.

6.1.6 MainStudentForm

Controls the interface of the Student application. Controls the family of forms that the Student uses.

6.1.7 BillingSystemAccess

This process communicates with the external Billing System to initiate student billing.

6.1.8 CloseRegistrationProcess

The Close Registration process is initiated at the end of the registration time period. This process communicates with the process controlling access to the Billing System.

6.1.9 BillingSystem

The Billing System supports the submitting of student bills for the courses registered for by the student for the current semester.

Analysis Mechanisms:

- Legacy Interface

6.1.10 CloseRegistrationController

The Close Registration Controller controls access to the Billing System.

Analysis Mechanisms:

- Distribution

6.2 Process to Design Elements 

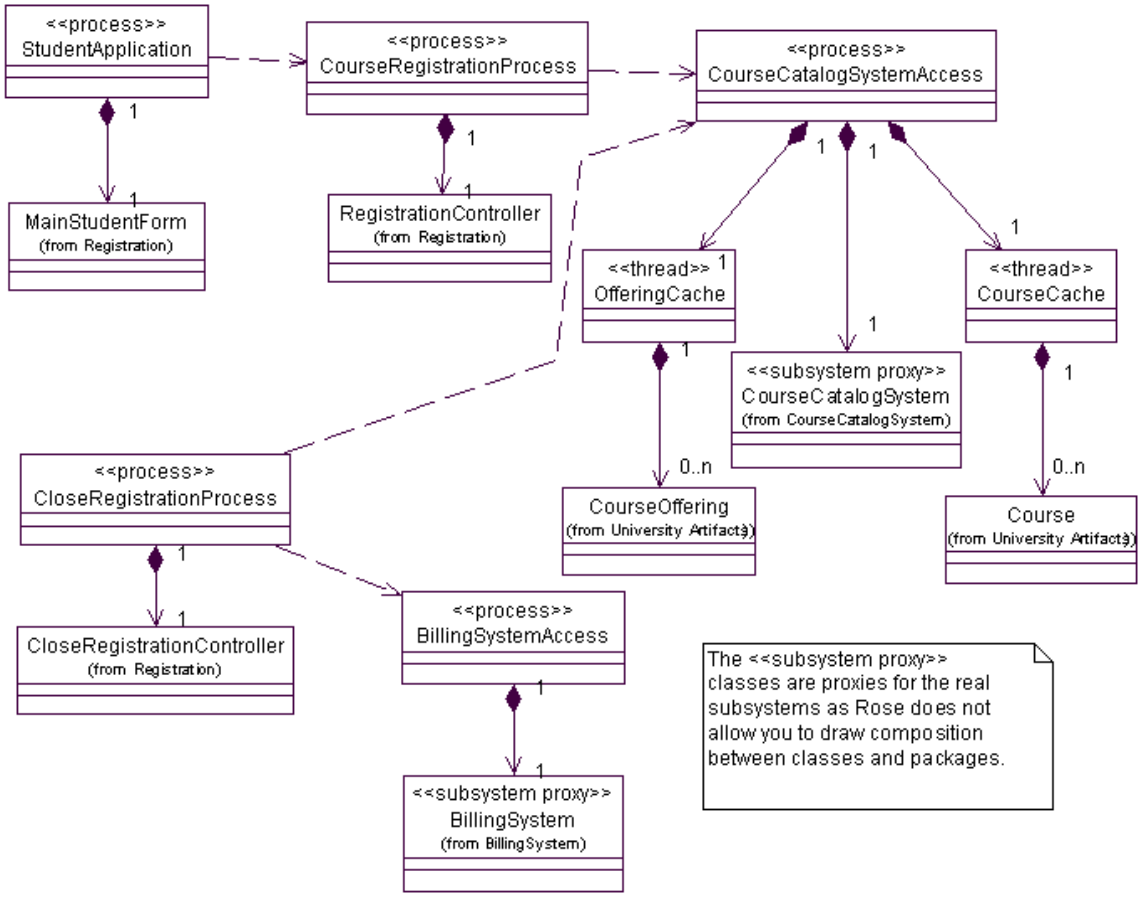


Diagram Name: Process to Design Elements

6.2.1 CourseCache

The Course Cache thread is used to asynchronously retrieve items from the legacy Course Catalog System.

6.2.2 OfferingCache

The OfferingCashe thread is used to asynchronously retrieve items from the legacy Course Catalog System.

6.2.3 Course

A class offered by the university.

Analysis Mechanisms:

- Persistency

- Legacy Interface

6.2.4 CourseOffering

A specific offering for a course, including days of the week and times.

Analysis Mechanisms:

- Persistency

- Legacy Interface

6.3 Process Model to Design Model Dependencies 

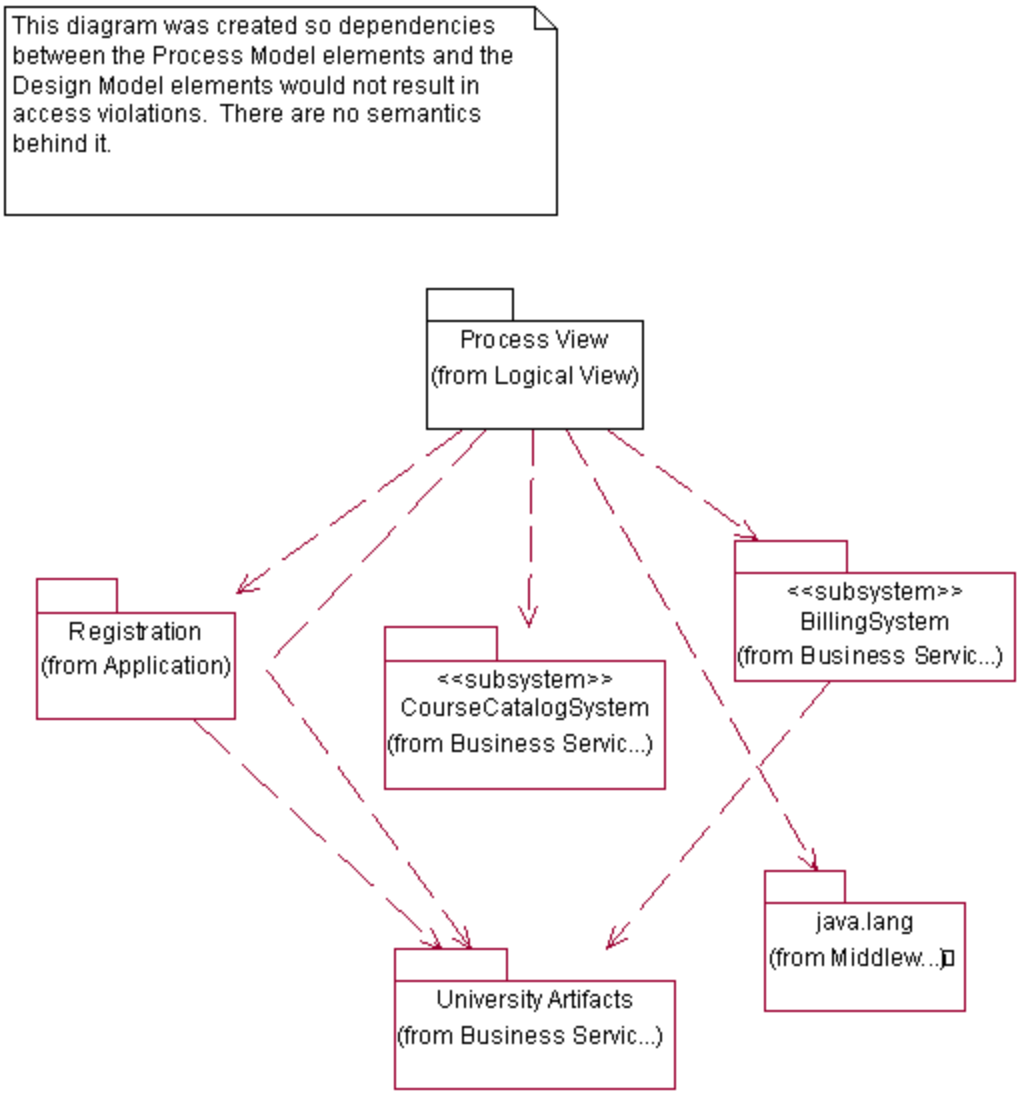


Diagram Name: Process Model to Design Model Dependencies

6.4 Processes to the Implementation

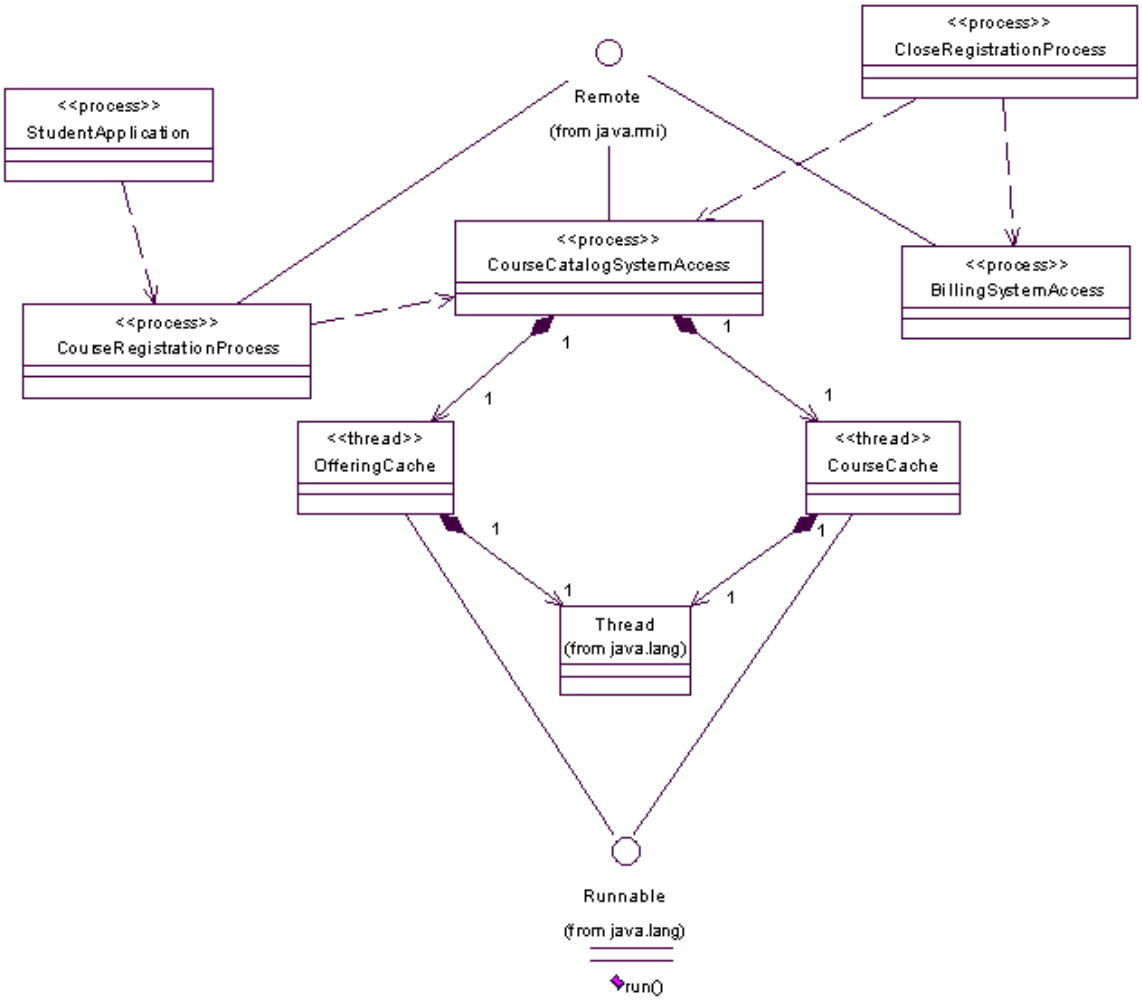


Diagram Name: Processes to the Implementation

6.4.1 Remote

* The Remote interface serves to identify all remote objects. Any object that is a remote object must directly or indirectly implement this interface. Only those methods specified in a remote interface are available remotely.

* Implementation classes can implement any number of remote interfaces and can extend other remote implementation classes.

6.4.2 Runnable

* The *Runnable* interface should be implemented by any class whose instances are intended to be executed by a thread. The class must define a method of no arguments called *run*.

* This interface is designed to provide a common protocol for objects that wish to execute code while they are active. For example, *Runnable* is implemented by class *Thread*.

* Being active simply means that a thread has been started and has not yet been stopped.

6.4.3 Thread

* A *thread* is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.

* Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority. Each thread may or may not also be marked as a daemon. When code running in some thread creates a new Thread object, the new thread has its priority initially set equal to the priority of the creating thread, and is a daemon thread if and only if the creating thread is a daemon.

7. Deployment View

A description of the deployment view of the architecture Describes the various physical nodes for the most typical platform configurations. Also describes the allocation of tasks (from the Process View) to the physical nodes.

This section is organized by physical network configuration; each such configuration is illustrated by a deployment diagram, followed by a mapping of processes to each processor.

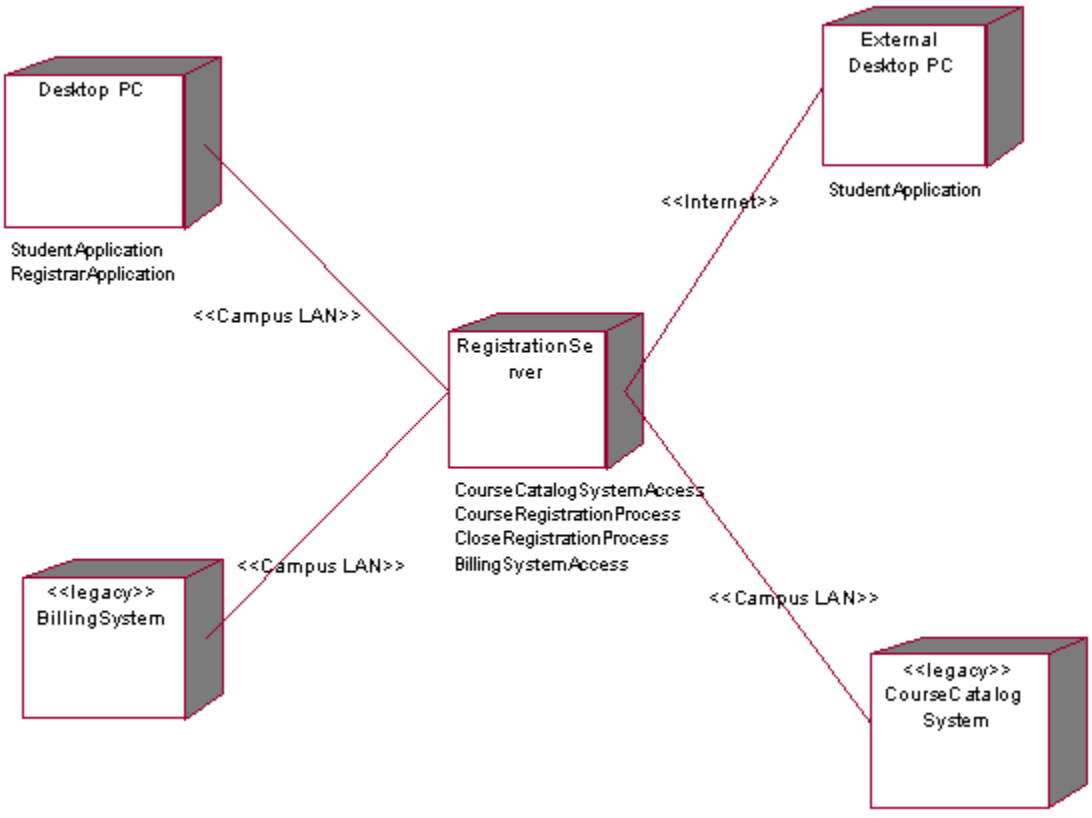


Diagram Name: Deployment View

7.1 External Desktop PC

Students register for courses using external desktop PCs which are connected to the College Server via internet dial up.

7.2 Desktop PC

Students register for courses via local Desktop PCs that are connected directly to the College Server via LAN. These local PCs are also used by professors to select course and submit student grades. The Registrar uses these local PCs to maintain student and professor information.

7.3 Registration Server

The Registration Server is the main campus UNIX Server. All faculty and students have access to the Server through the campus LAN.

7.4 Course Catalog

The Course Catalog System is a legacy system that contains the complete course catalog. Access to it is available via the College Server and LAN.

7.5 Billing System

The Billing System (also called the Finance System) is a legacy system that generates the student bills each semester.

8. Size and Performance

The chosen software architecture supports the key sizing and timing requirements, as stipulated in the Supplementary Specification [15]:

1. The system shall support up to 2000 simultaneous users against the central database at any given time, and up to 500 simultaneous users against the local servers at any one time.
2. The system shall provide access to the legacy course catalog database with no more than a 10 second latency.
3. The system must be able to complete 80% of all transactions within 2 minutes.
4. The client portion shall require less than 20 MB disk space and 32 MB RAM.

The selected architecture supports the sizing and timing requirements through the implementation of a client-server architecture. The client portion is implemented on local campus PCs or remote dial up PCs. The components have been designed to ensure that minimal disk and memory requirements are needed on the PC client portion.

9. Quality

The software architecture supports the quality requirements, as stipulated in the Supplementary Specification [15]:

1. The desktop user-interface shall be Windows 95/98 compliant.
2. The user interface of the C-Registration System shall be designed for ease-of-use and shall be appropriate for a computer-literate user community with no additional training on the System.
3. Each feature of the C-Registration System shall have built-in online help for the user. Online Help shall include step by step instructions on using the System. Online Help shall include definitions for terms and acronymns.
4. The C-Registration System shall be available 24 hours a day, 7 days a week. There shall be no more than 4% down time.
5. Mean Time Between Failures shall exceed 300 hours.
6. Upgrades to the PC client portion of C-Registration shall be downloadable from the UNIX Server over the internet. This feature enables students to have easy access to system upgrades.