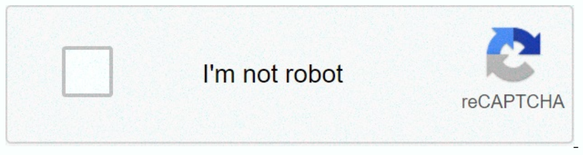

Continue

System architecture design document template

Related Information Course System Registration Software Architecture Document Version 1.0 Revision History Release Description Version Description Author 21/March/1999 1.0 Software Architecture Document generated using rational soda template and rational rose model. S. Johnson Content 1. Introduction 1.1 Purpose 1.2 Scope 1.3 Definitions, abbreviations and abbreviations 1.4 References 2. Architectural representation 3. Architectural objectives and limitations 4. View use case 4.1 Architecturally significant use cases 5. Overview of logical display architecture 5.1 - Package layering and subsystem 6. Process View 6.1 Process 6.2 Process Element Design Process 6.3 Process Model to Design Model Dependencies 6.4 Processes to Implement 7. View Deployment 7.1 External Desktop 7.2 Desktop PC 7.3 Registration Server 7.4 Course Catalog 7.5 Billing System 8. Size and power 9. Document on quality software architecture 1. Introduction This document provides a comprehensive architectural overview of the system using a number of different architectural views to display different aspects of the system. Its aim is to capture and pass on important architectural decisions that have been made in the system. This software architecture document provides an architectural overview of the C.C-Registration System being developed by Wylie College to support online course registration. This document was created directly from the C-Registration Analysis & Design Model implemented in Rose. Most sections were extracted from the Rose model using the SoDA template and software architecture document. Applicable links are: Course Billing Interface Specification, WC93332, 1985, Wylie College Press. Course Catalog Database Specification, WC93422, 1985, Wylie College Press. Vision document C-registration system, WyIT387, V1.0, 1998, Wylie College IT. Glossary for registration system C, WyIT406, V2.0, 1999, Wylie College IT. Use case specification - Close registration, WyIT403, V2.0, 1999, Wylie College IT. Use Case Specification – Login, WyIT401, V2.0, 1999, Wylie College IT. Use Case Specification - Maintain Professor Info, WyIT407, Version 2.0, 1999, Wylie College IT. Use Case Specifications - Sign up for courses, WyIT402, version 2.0, 1999, Wylie College IT. Use case specification - Select courses to teach, WyIT405, version 2.0, 1999, Wylie College IT. Use case specification - Maintain student information, WyIT408, version 2.0, 2 1999, Wylie College IT. Use Case Specification - Submit Grades, WyIT409, Version 2.0, 1999, Wylie College IT. Use Case Specification - View Report Card, WyIT410, Version 2.0, 1999, Wylie College IT. Software Development Plan for Registration System C , WyIT418, V1.0, 1999, Wylie College IT. E1 Iteration Plan, WyIT420, V1.0, 1999, Wylie College IT. Additional Specifications, WyIT400, V1.0, 1999, Wylie College, IT. 2. Architectural representation This document presents architecture as a series of views; use case view, logical view, process view, and View. No separate implementation view is described in this document. These are views of the basic model of unified modeling (UML) developed using Rational Rose. 3. Architectural objectives and limitations There are some key requirements and system limitations that have a significant impact on architecture. They are: The existing older course catalog system at Wylie College must be accessible to get all course information for the current semester. Registration system C must support data formats and recalculation of db-magis of the older course catalogue system [2]. The existing older billing system at Wylie College must be linked to student billing support. This interface is defined in the course billing interface specification [1]. All functions of students, professors and registrars must be available on both local university computers and remote computers with internet access. Registration system C shall ensure full protection of data against unauthorised access. All remote accesses are subject to user identification and password checking. Registration System C will be implemented as a client-server system. The client part is located on computers, and part of the server must work on the Wylie College UNIX server. [3] When developing architecture, account should be taken of all performance and load requirements as set out in the vision document [3] and the additional specification [15]. 4. View the use case Description of the software architecture use case view. Viewing a use case is an important input for selecting the set of scenarios or use cases that are at the center of the iteration. Describes a set of scenarios or use cases that represent some significant, central functionality. It also describes a set of scenarios and/or use cases that have substantial architectural coverage (which perform many architectural elements) or that highlight or illustrate a specific, delicate point of architecture. Cases of use of registration C are: - Login - Registration for courses - Maintain student information - Maintain information about the professor - Select courses to teach - Send grades - View report card - Close registration. These use cases are initiated by the student, professor or registrar actors. In addition, interactions with external actors; The course catalog and billing system occur. 4.1 Architecturally significant use cases Diagram name: Architecturally significant use cases 4.1.1 Close registrations Brief description: This use case allows the registrar to close the registration process. Course offers that do not have enough students are cancelled. The course offer must have at least three students. The billing system is notified to each student in each course offer that is not canceled, so the student may be charged for the course offer. The registrar is the main actor in this use case. The billing system is the actor involved in this use case. 4.1.2 Login brief description use case describes how the user logs on to the course registration system. The actors who begin this use case are the student, the professor and the registrar. 4.1.3 Maintain Professor Information Brief Description: This use case allows the registrar to maintain information about the professor in the registration system. This includes adding, modifying and removing professors from the system. The registrar is a participant in this use case. 4.1.4 Select courses to teach a brief description: This use case allows the professor to select course offers (date and time-specific courses) from the course catalogue to which he is entitled and intends to teach in the upcoming semester. The actor who starts this use case is a professor. The course catalog system is an actor within a use case. 4.1.5 Registration for courses Brief description: This use case allows the student to register for courses in the current semester. The student can also edit or delete the course selection if the changes are made within the addition/drag period at the beginning of the semester. The billing system is notified of all registration updates. The course catalogue contains a list of all course offers for the current semester. The main actor in this use case is the student. The course catalog system is an actor within a use case. 4.1.6 View report card Brief description: This use case allows the student to view his report card for a previously completed semester. The student is an actor in this use case. 4.1.7 Send Degrees Brief Description: This use case allows the professor to submit student grades for one or more classes completed in the previous semester. The actor in this case use is a 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, editing, and removing students from the system. The actor for this use case is the registrar. 5. Logical view Description of the logical view of the architecture. Describes the most important classes, their organization in service packages and subsystems, and the arrangement of these subsystems into layers. It also describes the most important implementations of a use case, such as dynamic aspects of architecture. Class diagrams can be included to illustrate relationships between architecturally significant classes, subsystems, packages, and layers. A logical view of the course registration system consists of 3 main packages: User Interface, Business Services, and Business Objects. The UI package contains classes for each form that actors use to communicate with the system. There are boundary classes that support signing up, maintaining plans, maintaining professor information, selecting courses, sending grades, maintaining student information, closing enrollment, and viewing report card. The business services package contains control classes for linking to billing checking student registration and managing student evaluations. The business object package contains entity classes for university artifacts (i.e. course offering, schedule), and boundary classes for interfaces running the course catalog. 5.1 Architecture Overview - Package layering and subsystem 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 on the Process Objects layer; that steps away from separating the client from the middle class. 5.1.2 Business Services Business Services Process Layer has all controller classes that represent use case manager that controls application behavior. This layer represents client-to-middle-tier boundaries. The Business Services layer depends on the Process Objects layer; that steps away from separating the client from the middle class. 5.1.3 Middleware middleware supports access to The Relation DBMS and OODBMS. 5.1.4 Basic reuse The basic reuse package contains classes to support the list of features and patterns. 6. Process view Description of the architecture process view. Describes the tasks (processes and threads) involved in system execution, interaction, and configuration. It also describes the allocation of objects and classes to tasks. The process model illustrates the course registration classes organized as executable processes. There are processes to support student registration, professor function, closure of registration, and access to external billing system and course catalog system. 6.1 Processes 6.1.1 CourseCatalogSystem Access This process manages access to an older system of course catalogues. It can be shared by multiple users who register for courses. This allows you to cache recently loaded courses and bids to improve performance. Separate threads within the CourseCatalog, CourseCache, and OfferingCache process are used to asynchronously retrieve items from an older system. Analytical mechanisms: - Legacy traceability interface requirements: - Design limitations: The system integrates with the existing legacy system (course catalog database). 6.1.2 KurzCatalog A shameless catalogue of all courses and courses offered by the university, including courses from previous semesters. This class acts as an adapter (see gamma pattern). This works by ensuring that CourseCatalogSystem is accessible through the ICourseCatalog interface to the subsystem. 6.1.3 Course registrationProcess is one instance of this process for each student who is currently enrolling in courses. 6.1.4 RegistrationController This supports the use case, which allows the student to register for courses in the current semester. The student can also edit or delete the course selection if the changes are made within the addition/drag period at the beginning of the semester. Analytical mechanisms: - Distribution 6.1.5 StudentApplication Manages the Student including user interface processing and coordination with business processes. There is one instance of this process for each student who is currently registering for courses. 6.1.6 MainStudentForm Controls the Student application interface. Manages 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 starts at the end of the registration period. This process communicates with the billing system access control process. 6.1.9 BillingSystem Billing system supports the presentation of student accounts for courses registered by the student for the current semester. Analysis mechanisms: - Legacy interface 6.1.10 CloseRegistrationController Close registration controller controls access to the billing system. Analytical mechanisms: - Distribution 6.2 Process to design element diagram name: The process of designing elements 6.2.1 CourseCache Thread cache courses is used to asynchronously retrieve items from an older course catalog system. 6.2.2 OfferingCache The OfferingCashe thread is used to asynchronously retrieve items from an older course catalog system. 6.2.3 Course A class offered by the University. Analytical mechanisms: - Persistence - Legacy Interface 6.2.4 CourseOffering Specific offer for the course, including days of the week and times. Analysis mechanisms: - Persistence - Legacy interface 6.3 Process model to design model dependencies Diagram name: Process model to design dependency model 6.4 Processes to implement diagram name: Implementation processes 6.4.1 Remote * Remote interface is used to identify all remote objects. Each object that is a remote object must directly or indirectly implement this interface. Only methods specified in the remote interface are remotely available. * Implementation classes can implement any number of remote interfaces and can extend other remote implementation classes. 6.4.2 Runnable * Runnable should be implemented by any class whose instances are intended to be run by a thread. The class must define a method for any arguments called run. * This interface is designed to provide a common protocol for objects that want to run code when they are active. For example, Runnable is implemented by the Thread class. * Being active simply means that the thread has started and has not yet been stopped. 6.4.3 Thread * Thread is a thread implementation in a program. The Java VM allows an application to have multiple threads running at the same time. * Each thread has priority. Higher priority threads are executed preferably over lower priority threads. Each thread may or may not be marked as a daemon. When code running on a thread creates a new Thread object, the new thread has a priority that initially equals Creating a thread and being a thread daemon if and only if the thread creation is a daemon. 7. Deployment View Description view deployment architecture describes the different physical nodes for the most typical platform configurations. It also describes how to allocate tasks (from the view process) to physical nodes. This section is organized according to the physical configuration of the network. each such configuration is represented by a deployment diagram followed by process mapping to each processor. Diagram Name: View deployment 7.1 External desktop computer Students sign up for courses using external desktop computers that are connected to College Server over an Internet dial-up connection. 7.2 Desktop students register for courses through local desktop computers that are connected directly to the College Server over a LAN. These local computers are also used by professors to select a course and submit student grades. The registrar uses these local computers to maintain information for students and professors. 7.3 Registration Server Registration Server is the main campus of UNIX Server. All faculties and students have access to the server through the LAN campus. 7.4 Course catalogue The course catalogue system is an older system that contains a complete catalogue of courses. Access to it is available through a college server and LAN. 7.5 Billing system Billing system (also called financial system) is an older system that generates student accounts every semester. 8. Size and Performance The chosen software architecture supports key size and timing requirements as specified in the additional specification [15]: The system must support up to 2000 current users against a central database at any given time and up to 500 simultaneous users against local servers at any one time. The system provides access to an older course catalog database with a maximum of 10-second latency. The system must be able to complete 80% of all transactions within 2 minutes. The client part must require less than 20 MB of disk space and 32 MB of RAM. The selected architecture supports size and timing requirements through client-server architecture implementations. The client part is implemented on local school computers or on remote dialing computers. Components have been designed to ensure that minimum disk and memory requirements are needed for part of the computer client. 9. Quality The software architecture supports quality requirements as set out in the additional specification [15]: The desktop user interface must meet the requirements for Windows 95/98. The user interface of registration system C must be designed for ease of use and must be suitable for a community of computer literate users without further training in the system. Each feature of registration system C must have built-in online help for the user. Online Help provides step-by-step instructions on how to use the system. Online Help must contain definitions of terms and abbreviations. available 24 hours a day, 7 days a week. There must be no more than 4% dream. The mean time between failures must exceed 300 hours. Upgrades to the C-Registration pc client part must be downloaded from the UNIX server over the Internet. This feature gives students easy access to system upgrades. Upgrades.