### **Iteration 2: Identifying Structures to Support Primary Functionality**

### 

### **Step 2: establish iteration goal by selecting Drivers**

In this iteration, besides CRN 3, which is to allocate works to members of the development team, following primary use cases will be addressed:

* UC1: Manage Courses
* UC7: Calculate grade statistics
* UC10: Retrieve Course Information
* UC11: Subscribe/Unsubscribe to courses
* UC13: Share files and messages with team
* UC25: Email students

### **Step 3: Choose One or More elements of the system to refine**

The modules located in the different layers by the reference architectures from previous iteration will be refined in this iteration.

### 

### **Step 4: Choose One or More Design Concepts that satisfy the selected Drivers**

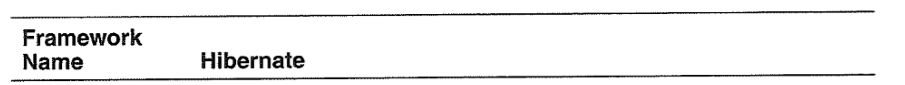
|  |  |
| --- | --- |
| Design Decisions and Location | Rationale and Assumptions |
| Create a Domain Model | It is to create an initial domain system with major entities. |
| Identify Domain Objects | CMS need to have domain objects where each distinct functional element of the application has to be encapsulated in a self-containing building block |
| Decompose Domain Objects into general and specialized components | Domain objects are complete sets of functionality supported by finer grained elements located within the layers. |
| Spring framework | The application framework allows the objects that form an application to be connected. It also supports different concerns through AOP.  Supports:   * Security (QA-5) * Publishing object interfaces so the objects can be accessed remotely |
| Hibernate Framework | Hibernate allows objects to be easily persisted in a relational database. It supports transactions and provides a query language that is used to retrieve objects from the database (UC-10). It also utilizes multi-level caching schemes to improve performance. |

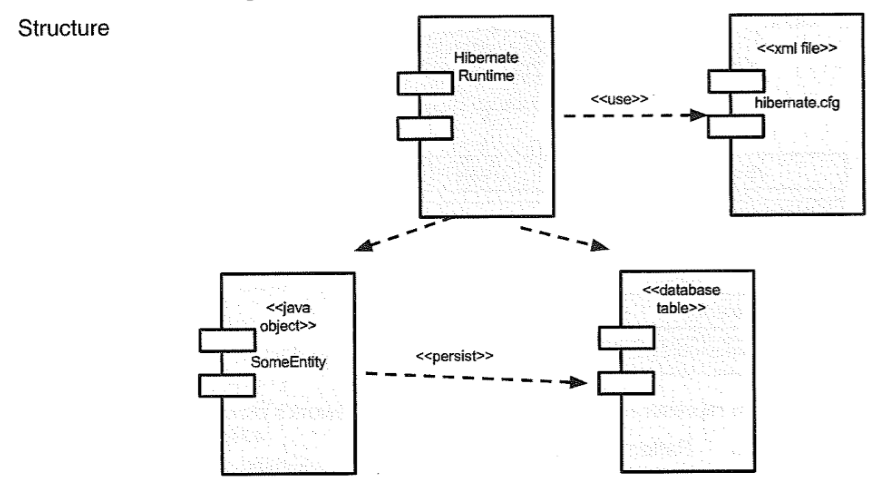
### 

### **Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces**

|  |  |
| --- | --- |
| Design Decisions and Location | Rationale |
| Map system use cases to domain objects | By analyzing system use cases, domain objects can be identified. |
| Decompose domain objects across layers to identify layer-specific modules | Through establishing set of modules, the needs to test these modules are clearer. |
| Associate components with Spring framework | This framework supports security and allows remote access. |
| Associate data layer components with Hibernate | This framework provides easy-to-use tools of handling database. |

### 

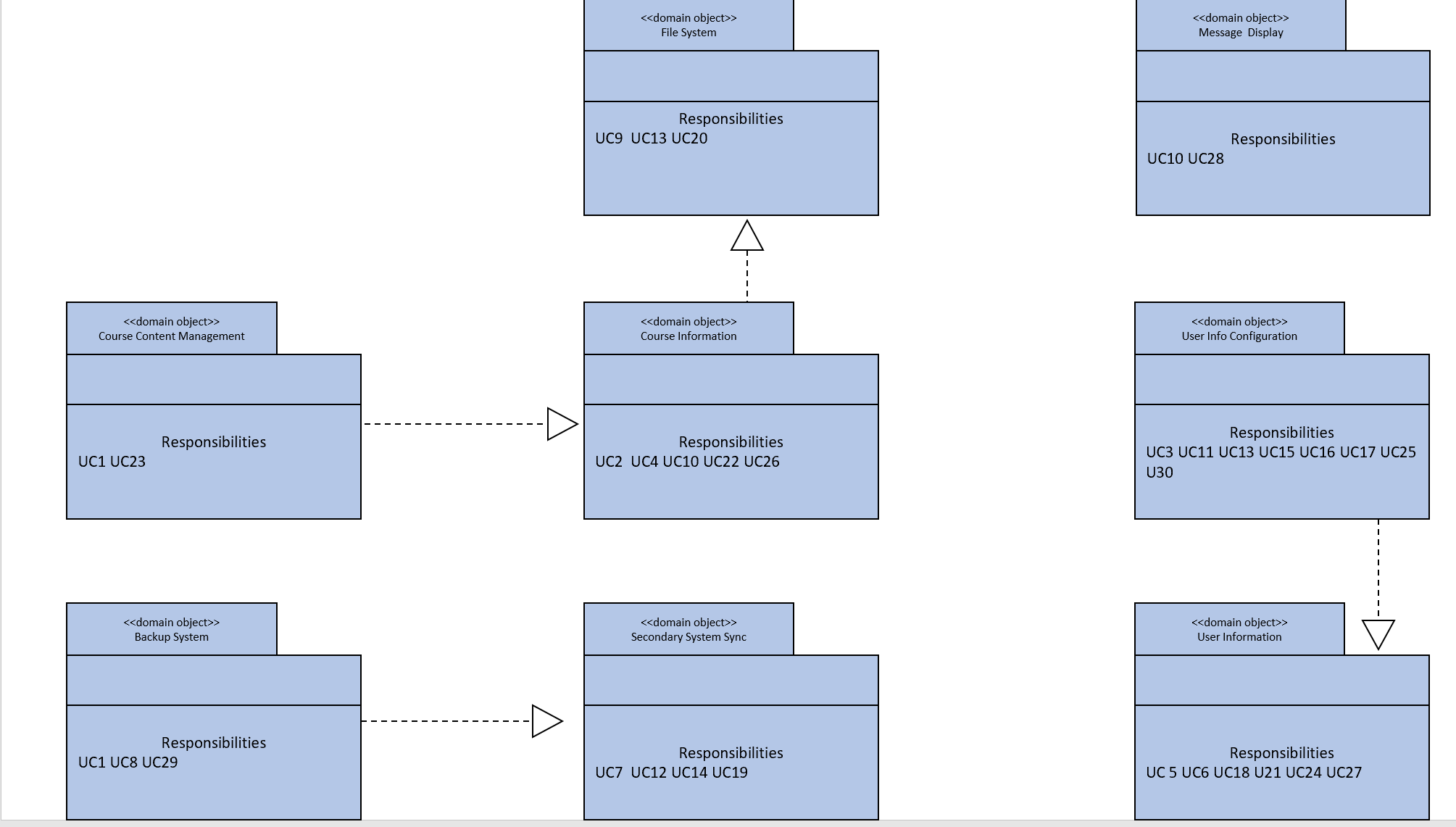




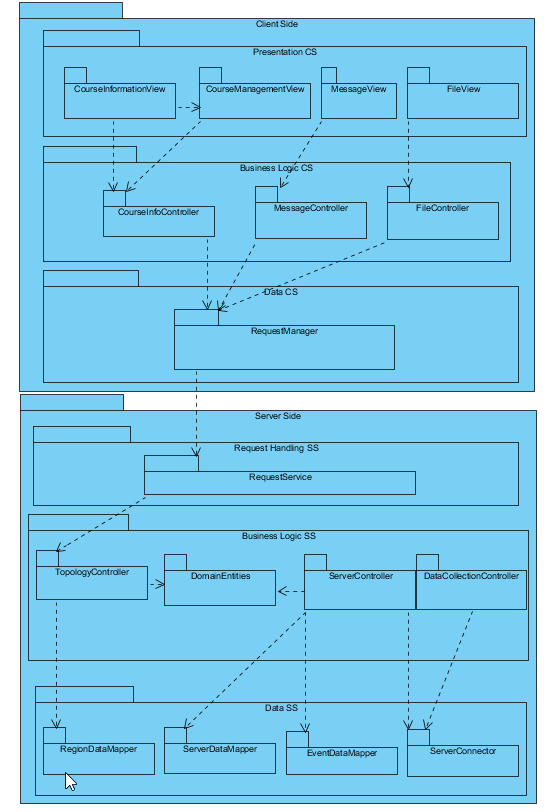
### **Step 6: Sketch Views and Record Design Decisions**

### 

**Domain Objects**



**Modules for primary use case functionality**

****

|  |  |
| --- | --- |
| **Element** | **Responsibility** |
| CourseInformationView | Displays and updates course information when receiving events |
| CourseManagementView | Displays and updates the overall course management system. Emcompasses both UI-components as well as UI process components |
| MessageView | Displays and updates information regarding the messaging system |
| FileView | Displays and updates information regarding the file sharing system |
| CourseInfoController | Responsible for providing the necessary information to the presentation layer for displaying the course information and management system information |
| MessageController | Responsible for providing the necessary information to the presentation layer for displaying messaging information |
| FileController | Responsible for providing the necessary information to the presentation layer for displaying file sharing information |
| RequestManager | Responsible for the communication with the server-side logic |
| RequestService | Provides a facade that receives request from clients |
| TopologyController | Contains business logic related to the topological information |
| DomainEntities | Contains the entities from the domain model that reside in the server side |
| ServerController | Contains business logic related to the management on server side events |
| DataCollectionController | Contains logic to perform data collection and storage |
| RegionDataMapper | Responsible for the persistence operations (CRUD) related to the regions |
| ServerDataMapper | Responsible for the persistence operations (CRUD) related to the server |
| EventDataMapper | Responsible for the persistence operations (CRUD) related to the events |
| ServerConnector | Responsible for the communication with the servers. It isolates and abstracts operations with the servers to support communication with different types of servers |

### **Step 7: Perform Analysis of current Design and review iteration Goal and Achievement of Design Purpose**

|  |  |  |  |
| --- | --- | --- | --- |
| Not Addressed | Partially Addressed | Completely Addressed | Design Decisions Made During the Iteration |
|  |  | UC1 | Modules to manage courses are identified |
|  |  | UC7 | Modules to manage grades have been identified |
|  |  | UC8 | By removing local data, admin can manage backups of the system |
|  | UC10 |  | No relevant design decisions |
|  |  | UC11 | Modules to manage subscription have been identified. |
|  |  | UC13 | Modules to manage file system have been identified |
|  |  | UC25 | Modules to manage message system have been identified |
|  | QA-1 |  | No relevant decision. |
| QA-2 |  |  | No relevant decision |
|  | QA-5 |  | No relevant decision |
|  | QA-10 |  | Identified domain models make possible easy maintenance of the system |
| CON-1 |  |  | No relevant decision |
| CON-2 |  |  | No relevant decision |
|  | CON-4 |  | Modules that handle file and course system partially address this issue. |
|  |  | CRN-1 | No relevant decision. |
|  |  | CRN-2 | OODP concept has been applied to draw the picture |
|  |  | CRN-3 | Works are delegated according to identified domain objects. |

### 