4.3.3 Iteration 2: Identifying Structures to Support Primary Functionatity

This section presents the results of the activities performed in each of the steps of ADD in the second iteration of the design process for the QLTV system. In this iteration, we move from the generic and coarse-grained descriptions of functionality used in iteration 1 to more detailed decisions that will drive implementation and the formation of development teams.

This movement from the generic to the specific is intentional and built into the ADD method. We cannot design everything up front, so we need to be disciplined about which decisions we make, and when, to ensure that the design is done systematically, addressing the biggest risks first and moving from there to ever finer details. Our goal for the first iteration was to establish an overall system structure. Now that this goal has been met, our new goal for this second iteration is to reason about the units of implementation, which affect team formation, interfaces, and how development tasks may be distributed, outsourced, and implemented in sprints.

4.3.3.1 Step 2: Establish Iteration Goal by Selecting Drivers The goal of this iteration is to address the general architectural concern of identifying structures to support primary functionality. Identifying these elements is useful not only for understanding how functionality is supported, but also for addressing CRN-3 that is, the allocation of work to members of the development team.

In this second iteration, besides CRN-3, the architect considers the system's primary use cases:

▪ UC-1

▪ UC-2

▪ UC-7

4.3.3.2 Step 3: Choose One or More Elements of the System to Refine The elements that will be refined in this iteration are the modules located in the different layers defined by the two reference architectures from the previous iteration. In general, the support of functionality in this system requires the collaboration of components associated with modules that are located in the different layers.

4.3.3.3 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers In this iteration, several design concepts in this case, architectural design patterns are selected from the book Pattern Oriented Software Architecture, Volume 4. The following table summarizes the design decisions. The words in bold in the following table refer to architectural patterns from this book, and can be found in Appendix A.

Driver Architectural Pattern

1 CRN-3 MVC (Model-View-Controller)

2 UC-1 Three-Tier Architecture

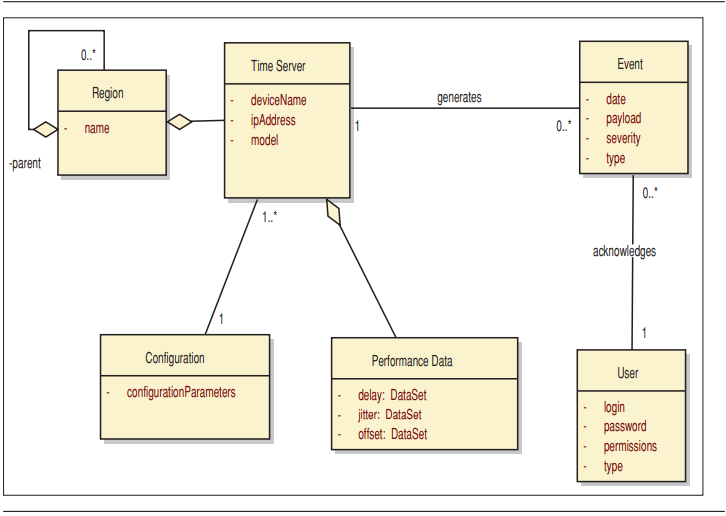
3 UC-2 Publish-Subscribe

4 UC-7 Front Controller

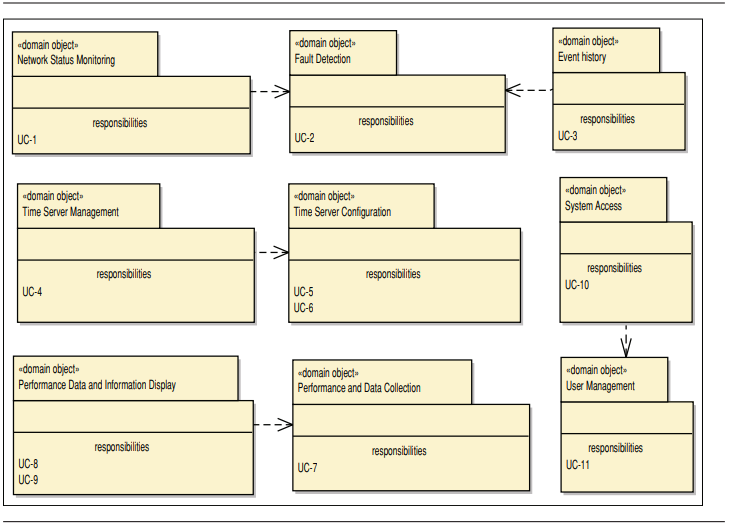
4.3.3.4 Step 5: Apply the Selected Design Concepts to the System Elements In this step, the architect applies the selected design concepts to the system elements by assigning specific responsibilities to the corresponding modules. This ensures that the modules in each layer collaborate effectively to support the system's primary functionality.

4.3.3.5 Step 6: Verify and Validate the Applied Design Concepts The architect verifies and validates the applied design concepts by reviewing the design artifacts, conducting formal inspections, and carrying out automated testing. This helps ensure that the refined system architecture effectively supports the identified functionality and satisfies the specified drivers.

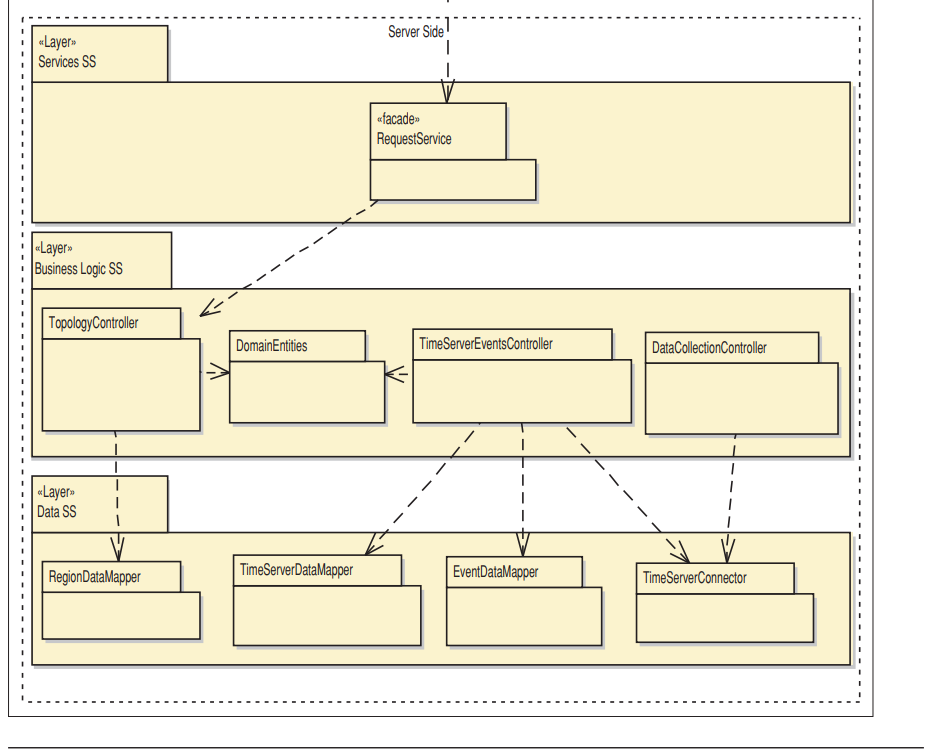
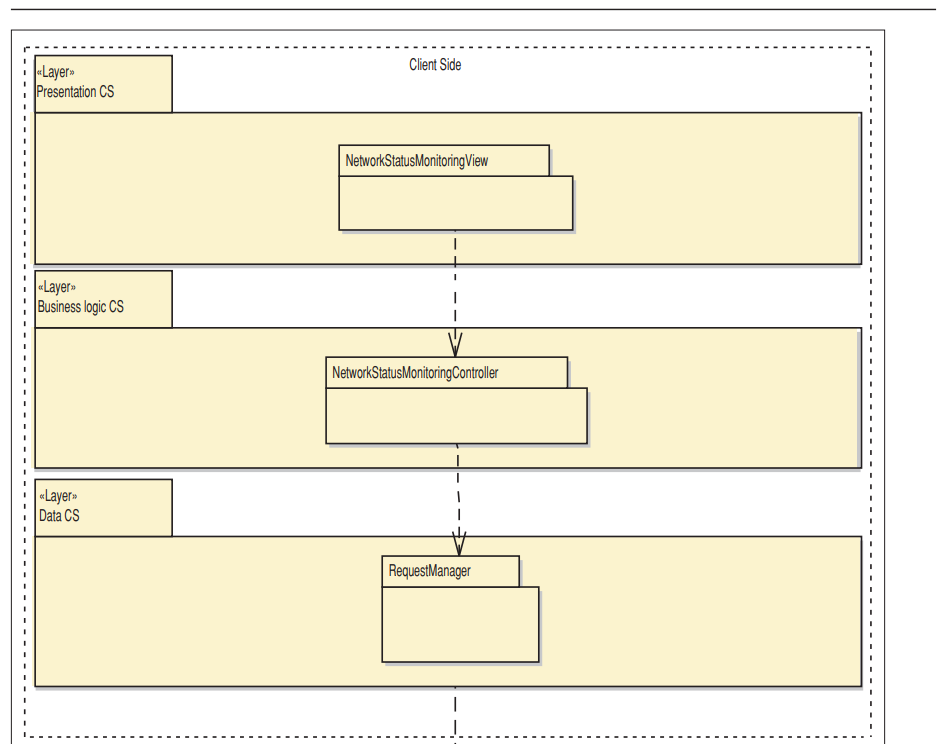
4.3.3.6 Step 7: Reflect on the Iteration The architect reflects on the outcome of the iteration by comparing the current system architecture with the one from the previous iteration. This helps identify areas of improvement and opportunities for further refinement. The architect also updates the relevant architectural models, documentation, and other artifacts to reflect the current system architecture.



Initial domain model (Key: UML)



Domain objects associated with the use case model (Key: UML)



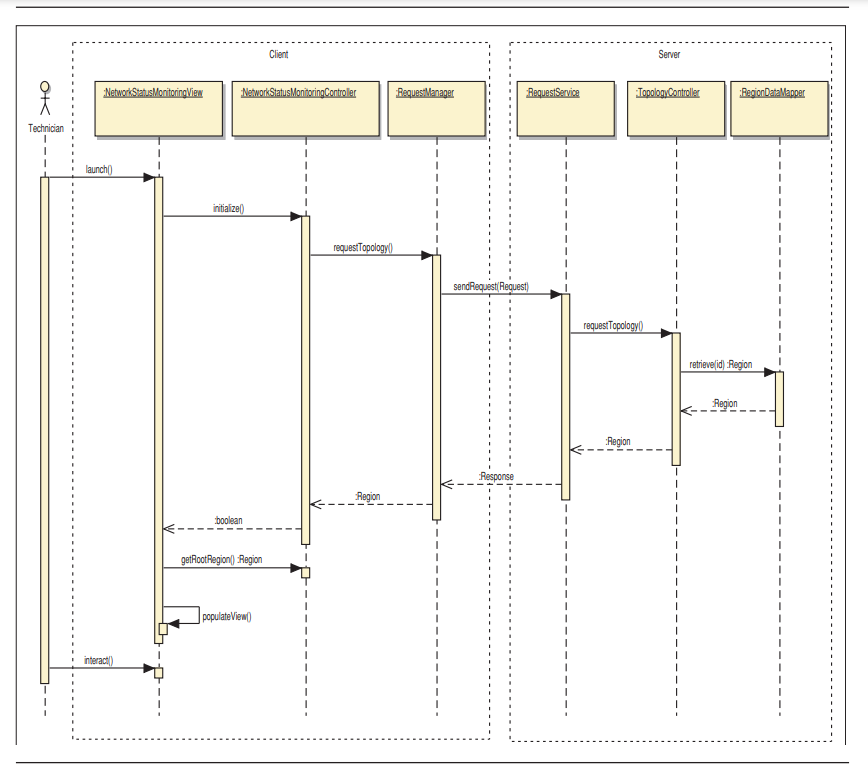
Modules that support the primary use cases (Key: UML)

|  |  |
| --- | --- |
| **Element** | **Responsibility** |
| NetworkStatusMonitoringView | Displays the network representation and  updates it when events are received.  This component embodies both UI components and UI process components from the reference architecture. |
| NetworkStatusMonitoringController | Responsible for providing the necessary information to the presentation layer for displaying the network representation. |
| RequestManager | Responsible for communication with the server-side logic. |
| RequestService | Provides a facade that receives requests from the clients. |
| TopologyController | Contains business logic related to the topological information. |
| DomainEntities | Contains the entities from the domain model (server side). |
| TimeServerEventsController | Contains business logic related to the management of events. |
| DataCollectionController | Contains logic to perform data collection and storage. |
| RegionDataMapper | Responsible for persistence operations (CRUD) related to the regions. |
| TimeServerDataMapper | Responsible for persistence operations  (CRUD) related to the time servers. |
| EventDataMapper | Responsible for persistence operations  (CRUD) related to the events. |
| TimeServerConnector | Responsible for communication with the  time servers. It isolates and abstracts  operations with the time servers to  support communication with different  types of time servers (see QA-2). |

For UC-1 (monitor network status), I will implement a dashboard to display information about the regions. I will provide a functionality that allows users to add new regions or edit existing regions. Additionally, I will provide a feature to display the links between regions.

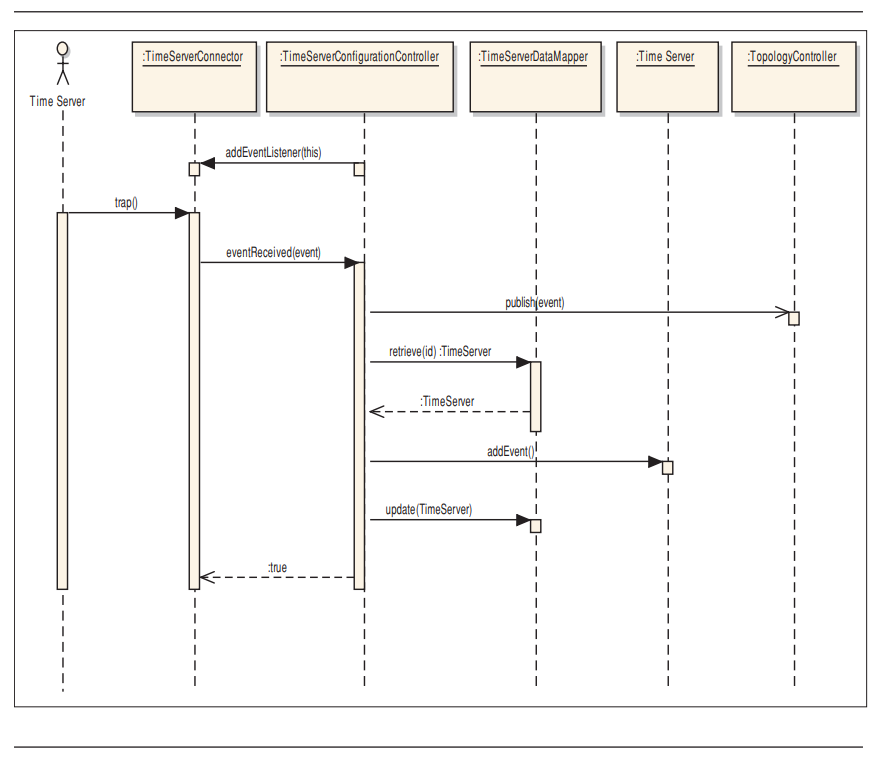
For UC-2 (configure region details), I will set up a form to allow users to enter information about a specific region. Alongside, I will provide a dropdown list to let users select a region type from a provided list.

For UC-7 (validate configuration changes), I will implement a warning dialog to notify users that they are making changes to the configuration that may not be allowed. I will provide a "Continue" button and a "Cancel" button to allow users to decide whether to proceed with the changes or not.



Sequence diagram for use case UC-1

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| **Element:** | NetworkStatusMonitoringContoller |
| boolean initialize() | Opens up the network representation so that users can interact with it. |
| Region getRootRegion() | Returns a reference to the root region and the neighbors of this object (excluding traps). |
| **Element:** | RequestManager |
| Region requestTopology() | Requests the topology. This method returns a reference to the root region from which it is possible  to navigate through the complete topology. |
| **Element:** | RequestService |
| Response  sendRequest(Request req) | This method receives a request. Only this method is exposed in the service interface. This simplifies  the addition of other functionality in the future without having to modify the existing service interface. |
| **Element:** | TopologyController |
| Region requestTopology() | Requests the topology. This method returns a reference to the root region from which it is possible  to navigate through the complete topology. |
| **Element:** | RegionDataMapper |
| Region retrieve(int id) | Returns a Region from its id. |



Sequence diagram for use case UC-2 (Key: UML)

Shows an initial sequence diagram for UC-2 (detect fault) shows only the components on the server side. The interaction starts with a TimeServer sending a trap, which is received by the TimeServerConnector. The trap is transformed into an Event and sent to the TimeServerConfigurationController. The Event is sent asynchronously to the TopologyController for publication to the clients and isthen persisted.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| **Element:** | TimeServerConnector |
| boolean addEventListener  (EventListener el) | This method allows components from the business logic to register themselves as listeners to events that are received from the time servers. |
| **Element:** | TimeServerConfigurationController |
| boolean eventReceived(Event  evt) | This callback method is invoked when an event is received. |
| **Element:** | TopologyController |
| publish(Event evt) | This method notifies the clients that a new event has occurred. |
| **Element:** | TimeServerDataMapper |
| TimeServer retrieve(int id)  boolean update(TimeServer ts) | Retrieves a TimeServer identified by its id.  Persists changes in a TimeServer. |

Step 7: Validate configuration changes

Goal and Achievement of Design Purpose

The goals and achievements of this iteration were to:

Identify the modules associated with the primary use cases of the Library Management System.

Identify the modules associated with the rest of the functionality of the system.

Create a work assignment table to address CRN-3.

Refine the architecture to accommodate new concerns and constraints.

