# IT 315 Final Project Part III Solution Submission Template

**Name:** Byron Laferriere

**Date:** 20 Dec 2020

1. *Generate your student information system (SIS) sequence diagram for the Register a Student for Classes use case.*

The following diagram is the Sequence Diagram for the Behavioral Model. Sequence Diagrams are desirable in systems design for their assistance with real-time specification and complex use cases because of their emphasis on time-based ordering of the activity (Dennis, Wixom, & Tegarden, 2015). This specific diagram maps out the ‘Register Students For Classes’ use case requested. This diagram is read left-right (verticality) and up-down (linearity). When working through the diagram it is read as, “Authorized User Searches By CourseID in Course Records.” This diagram begins mapping out the system in a more intricate detail than previously seen and focuses on system and object interactions. These models are meant to be revised throughout the process, as new system revelations come to light about user interaction with the system.



*Generate your SIS communication diagram for the Register a Student for Classes use case.*

The following diagram is referred to as a Communications Diagram. This diagram is similar in content to the Sequence Diagram, in that it provides a view of the dynamic aspects of an object-oriented system (Dennis, Wixom, & Tegarden, 2015). When making a communication diagram, however, it is important to maintain focus on message-passing relationships and not associations (Dennis, Wixom, & Tegarden, 2015). This diagram is meant to show process patterns and emphasize the flow of messages through a set of objects (Dennis, Wixom, & Tegarden, 2015). These diagrams are useful for systems engineers to be able to quickly evaluate the intended system design when beginning the programming process.

Diagram

Description automatically generated

**SIS Method Contract 1 template** (refer to textbook pages 306–314):

|  |  |  |
| --- | --- | --- |
| Method Name:  Verify Student ID | Class Name:  Registration | ID:  300 |
| Clients (Consumers):  Authorized Users | | |
| Associated Use Cases:  Maintain Student Record  RegisterStudentForClasses | | |
| Description of Responsibilities:  Accept user credential input and validate credentials against system of records to authorize user’s ability to manipulate the system. | | |
| Arguments Received:  String- Username  String- Password | | |
| Type of Value Returned:  String- Token representing authorization level granted via user credentials | | |
| Pre-Conditions:  Actor enters arguments (credentials) and receives access to the system. | | |
| Post-Conditions:  Actor receives token that identifies which level of clearance is permitted to the system. | | |

**SIS Method Contract 2 template:**

|  |  |  |
| --- | --- | --- |
| Method Name:  Search by CourseID | Class Name:  Registration | ID:  301 |
| Clients (Consumers):  Authorized Users (Enrollment Staff, Students) | | |
| Associated Use Cases:  Maintain Student Records  RegisterStudentForClasses | | |
| Description of Responsibilities:  Accept courseID input by authorized user and search course records for all classes matching courseID that are available after current date. Information is then relayed back to the authorized user. | | |
| Arguments Received:  CourseID - String | | |
| Type of Value Returned:  ClassID - String | | |
| Pre-Conditions:  Actor enters argument (CourseID) and searches the system for Classes that match. | | |
| Post-Conditions:  Actor receives a list of all classes that match courseID , which are available after current date | | |

**SIS Method Specification 1 template** (refer to textbook pages 314–318):

|  |  |  |
| --- | --- | --- |
| Method Name:  Verify Student ID | Class Name:  Registration | ID:  1 |
| Contract ID:  300 | Programmer:  B. Laferriere | Date Due:  06 Dec 2020 |
| Programming Language:  JAVA | | |
| Triggers/Events:  Actor attempts to login using credentials provided by university, in attempt to begin using system’s functionality. | | |

| **Arguments Received:**  **Data Type:** | **Notes:** |
| --- | --- |
| Username – String  Password – String | User’s University email address  12 Char’s Total, 1 Uppercase, 1 Lowercase, 2 Speical Char |

| **Messages Sent & Arguments Passed:**  **ClassName.MethodName:** | **Argument Data Type:** | **Notes:** |
| --- | --- | --- |
| IAM.DelegateAuthorization | String | username |
| String | password |
|  |  |

| **Argument Returned:**  **Data Type:** | **Notes:** |
| --- | --- |
| String | Authentication Token issued per credentials provided through the IAM class. |
| Algorithm Specification:  Call delegate model to perform authorization steps via the enterprise directory service. | |
| Misc. Notes:  This method uses an external source (IAM) to provide the user a token. IAM is an enterprise directory service that grants a single sign-on and universal credential management. | |

**SIS Method Specification 2 template:**

|  |  |  |
| --- | --- | --- |
| Method Name:  Search by CourseID | Class Name:  Registration | ID:  2 |
| Contract ID:  301 | Programmer:  B. Laferriere | Date Due:  06 Dec 2020 |
| Programming Language:  JAVA | | |
| Triggers/Events:  Actor successfully logs in and chooses to search for available upcoming classes by searching via courseID. | | |

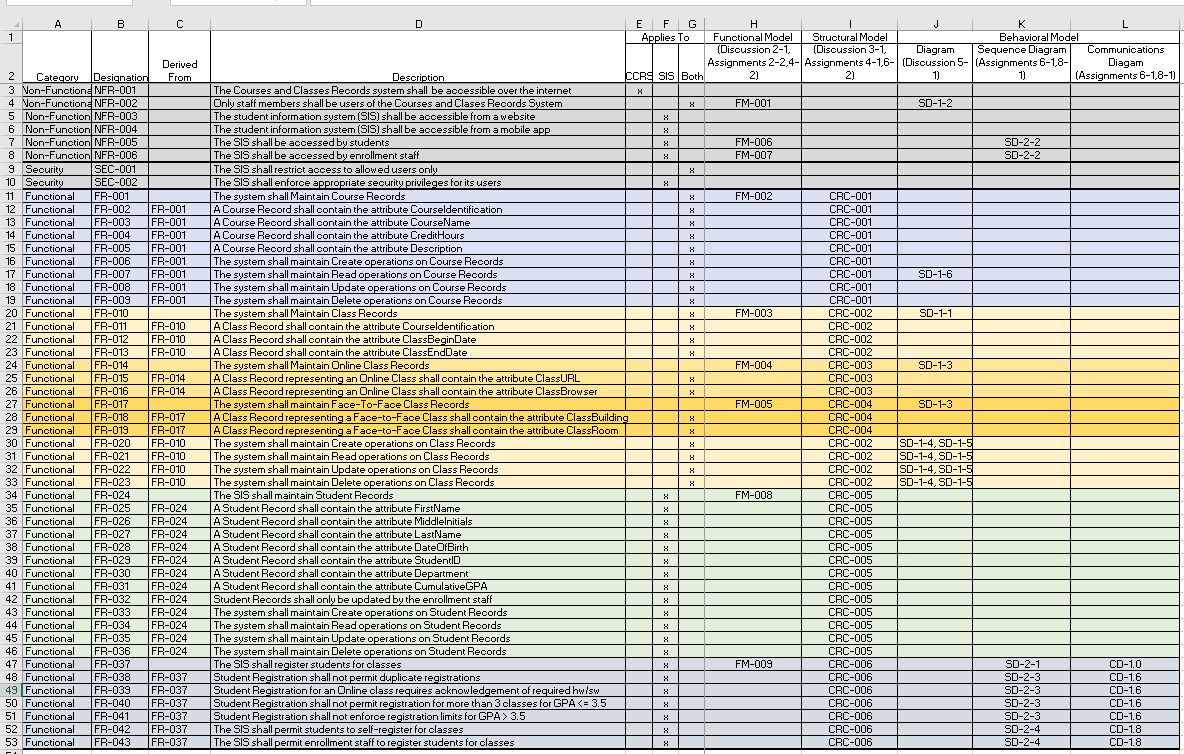
| **Arguments Received:**  **Data Type:** | **Notes:** |
| --- | --- |
| courseID - String | Sample courseID format : (SCS-300) Will usually have 3 letters followed by 3 numbers. |

| **Messages Sent & Arguments Passed:**  **ClassName.MethodName:** | **Argument Data Type:** | **Notes:** |
| --- | --- | --- |
| CourseRecords.SearchbyCourseID | String | courseID |
|  |  |
|  |  |

| **Argument Returned:**  **Data Type:** | **Notes:** |
| --- | --- |
| String | Results may contain more than one possibility for both online and face-to-face classes. |
| Algorithm Specification:  Call on CourseRecords for verification of existing CourseID. | |
| Misc. Notes:  When courseID entry matches an ID found in CourseRecords, there may be many class options per one entry search. If search does not yield results, display no match found. | |

1. *Verify and validate your sequence diagram and communication diagram against your SIS functional model and structural model.*

The best way to validate and verify that the Use Cases contained all required functionality from the Courses and Classes Requirements Definitions page, was to supply a spreadsheet containing the requested use cases. The attached Requirements Traceability Verification Matrix (RTVM) contains these use cases, as well as additional information about where these needs were met and, on which model they were satisfied within. In column H, the RTVM displays where the use cases can be found on the Functional Model. When viewing the Functional Model, look for the corresponding values assigned in this column, which can be matched back to use cases being defined and constructed on the Functional Model. In column I on the RTVM, the use cases have similar corresponding values that align with the Structural Model, which was used to help shape this model. In column J and column K, the Behavioral Model information can be located. Column J possesses the Sequence Diagram information for the use case ‘Maintain Class Records’ and column K possesses the Sequence Diagram information that corresponds to ‘Register Student For Classes.’ Lastly, in column L a reference guide for the Communications Diagram is provided as well. The Communication Diagram also corresponds to Register Student For Classes.’ This matrix was provided here to create a symbiotic presence between all the models and allows for easy location of use cases throughout each. Each use case identified in the SIS SubSystem has been met and assigned within the Models, to this point.



1. *Explain your approach to the problem, the decisions you made to arrive at your solution, and how you completed it.*

In the planning and design for the Sequence Diagram included, it was important to notice communications between objects from the initially provided documentation. I chose to design this phase of the system to represent that the user has already been authorized and, instead start with the system checking that the entered StudentID matches a record. I showed this step for clarification purposes when dealing with Enrollment Staff being the authorized user, to ensure they had the correct student identified to work on. After which, I chose to show an interaction between the authorized user and the course records system. The requirements definitions document detailed that a user should be able to search for upcoming classes that are available to them, by entering a courseID that returns all matching entries, both online and face-to-face. The next step then progresses to returning the available classes I previously generated with a return message, with which the Enrollment Staff can make available to the SIS Subsystem for either students or enrollment staff to choose from for up to one semester out. From here I used an alternative fragment to iterate through three possible selection conditions provided by the use cases. I used the alternative fragment because of how useful they can be when meeting multiple conditions. These fragments will check the Class\_Selected by the actor against the Student Records to prevent a duplicate registration, provide additional information on hardware and software if an online class is selected and ensure the student will be able to meet all of the requirements listed, and lastly it is going to be used to deny registration if there are already two classes a student with a GPA less than 3.5 out of 4 is registered for. If all of these conditions were met and agreed upon by the actor, the system then progresses to enrolling the student in the selected class. I set up this section to allow for the failure of a condition to progress to the IF/THEN/ELSE segment listed below. From here, the system will check for condition satisfaction first and then choose how to proceed. If the conditions are met, the registration process will accept the selected class and register the student. From here, the actor will be able to decide to continue or finish enrolling in future classes. If the conditions were not met, the IF statement will force the system to proceed to the ELSE statement and prompt the user with what conditional violation occurred. The user can either choose to exit the system or return to entering a courseID.

The communication diagram was similar in design, in that both the Sequence Diagram and Communication Diagram provide a view of the dynamic aspects of an object-oriented system (Dennis, Wixom, & Tegarden, 2015). A communication diagram is only meant to display the interactions between the objects in a less detailed manner than the Sequence Diagram. I chose the design to ensure that only Student Registration was in contact with the actor. From within Student Registration, the process for class registration can be completed by the user. Starting at the top of my Sequence Diagram, the Communication Diagram follows the messages in order and creates its own numerical system for identifying these messages. I progressed through the messages singularly, until I reached the alternative fragment that checked for three separate conditions. First, for the communication between the actor and Student Registration checking that the studentID entered is correct. This was given the label 1.01 as it is the start of the communication diagram. After that, the Student Registration system communicates with the Course Records System to retrieve all classes associated with an entered CourseID. The Course Records System responds with the 1.03 case and classes available. After Enrollment Staff updates the SIS with available classes in the upcoming semester that match the CourseID, both student or enrollment staff actors can select a class. The next step, once a class is selected, was an intriguing design task. In the Communication Diagram, since it is intended to be a simplified version of the Sequence Diagram, I grouped the three conditions into one message between the actor and the Student Record System. Creating these into a condition by using brackets ([ ]). I chose to represent the message in this style because of the fact that the system will not progress through the following bracketed messages if the condition within the bracket is not met or true (Dr. F. Manning, 2018). Therefore, in my communications model, if the student’s record does not satisfy the conditions in any way, it will automatically progress from step 1.07 to step 1.10, which states exactly why the conditions were not satisfied back to the actor. If the student’s record does satisfy the conditions, then the system will progress from 1.07 through 1.09 before skipping step 1.10. The Communication Diagram ends on 1.11, where I set this up to either exit the system or return to step 1.02. I did this because of the specification of this step in the SIS Requirements Documentation.

1. *Reflect on this experience and the lessons you learned from it.*

Initially, I really struggled with the conceptualization of the method contracts and specification cards. I found that during this process, it was easier to complete the Sequence Diagram first, which then gave me a better insight of how to shape the rest of the project. Moving from the Sequence Diagram to the Communication Diagram, to me, was the next logical step to take in the creation of this. Completing the communication diagram gave me a better idea of how to write the method contracts and specifications, by verbally working through the diagram out loud. During the revision process, however, I was able to clarify some of the questionable design decisions that I made previously and determine whether or not those decisions were correct. This seemed to become true after the final week’s reading and learning about the proper ways to transition from analysis to design. The thing that stands out most to me through this process, is that over-complication of object definition is a system designer’s worst enemy. To this point, I have consistently attempted to create and embellish detail in the project, which has benefitted me exactly zero times. Another thing to note, re-reading through the initial documentation that was provided to us earlier in the semester, has seemed to really help in the final stages of the development process. I am noticing a much better understanding of how to dissect that information and turn it into valuable data that can be used in the design process. In summation, I have come to realize that this process truly is more iterative and incremental than I the book can portray. As I look back through the process, I see opportunity for growth through repetition of the process. Object-Oriented Analyses and Design is undoubtedly something that is perfected over time.

*References*

Dennis, A., Wixom, B. H., & Tegarden, D. (2015). VitalSource Bookshelf Online. Retrieved November 07, 2020, from <https://mbsdirect.vitalsource.com/>

Manning, F., Dr. (2018, April 10). IT 315: Communications Diagram CC. Retrieved December 06, 2020, from <https://www.youtube.com/watch?v=6r5GHBXTtpM&feature=youtu.be>