TAEval

Requirements Analysis Document

Team Romero's Severed Head
Sean Benjamin
Dylan Kristolaitis
Justin Kung
Steven Wu

Submitted to:
Dr. Christine Laurendeau
COMP3004 Object-Oriented Software Engineering
School of Computer Science
Carleton University

Contents

Figures

Tables

1. Introduction

1.1 Purpose of System

In a university setting, the main purpose of attending is for increasing knowledge in a directed, focused manner. One may learn a field of study through books borrowed from the local library, but that structure is a stark contrast to a term filled with lectures from a distinguished PhD accompanied by tests, assignments, and exams that direct the student from point A to point B. Analogously, the current structure of the TA-Instructor relationship is unfocused. In communication of duties, task obligations are set at the start of the semester; but for communication of statuses and feedback for dynamically changing tasks, we still resort to e-mail. In assessing the qualities of a candidate, TA applications still rely on providing references of faculty that need to be manually contacted to receive feedback for performance that is dated or not directly applicable to the job at hand.

To allow the TAs to be successful in their job they need to have clear expectations about the tasks assigned to them for each of the courses that they TA. The instructors need to provide clear tasks and timely feedback to the TAs, to allow the TAs to complete their stated tasks at an appropriate level of satisfaction. Given that many TAs end up TA'ing repeatedly, it is invaluable for the future students in his or her section to benefit from the learning of the TAs previous errs and mistakes.

A unified system which would:

- allow the TA to know his exit criteria set by the Instructor for tasks
- allow the Instructor to know exactly if and when the task is completed from the TA
- allow the TA to receive feedback on previously completed tasks to improve upon the next instance of the same task
- allow the Administrator to be able to run reports on demand for TAs' evaluation data to judge their eligibility for future positions could solve the underlying problems with the current infrastructure.

The TAEval system is the proposed system to be used by TAs, Instructors, and Administrators that will allow Instructors to assign tasks to TAs of the course they are instructing and to provide feedback to the TAs about how they are doing on their tasks.
The scope of the TAEval system is for tasks and evaluation to be assigned, completed and evaluated over the course of the term.

The TAEval system will be comprised of the following main features for the Instructor:

Instructor can create, modify and deleted tasks.
Instructors will assign tasks to an associated TA.
Instructors will provide feedback and an evaluation rating for each task assigned to a TA.

The TAEval system will be comprised of the following main features for the Administrators:
Administrator will be to manage system data such as to courses, instructors, and TAs.
Administrators will be able to execute reports on the TAEval persistent data.

The TAEval system will be comprised of the following main features for the TAs:
TAs can view the tasks that have been assigned to them.
TAs can view evaluation on their tasks once they have been entered by the instructor.

For further details with regards to detailed system features, technical specifications, graphic user interface (GUI), data storage and inter-process communications refer to the TAEval system description.

1.2 Overview of Document

The purpose of this requirements analysis document is to provide an agreement with the client with respect to the functional and non-functional requirements of the TAEval system.

The document contains the following documentation with regards to the TAEval system:
List of functional requirements in a traceability matrix
List of non-functional requirements in a traceability matrix
Use case diagrams for the Instructor and TA actors
Detailed use case descriptions
Object model that is comprised of a data dictionary, which describes the TAEval entity, boundary and control objects, and a UML class diagram.
Dynamic model that is comprised of sequence diagrams, that map the instructor's user cases, and state machine diagrams that map only the entity objects.

2. Proposed System

2.1 Overview

In this section we outline the technical details of our proposed system, TAEval, by clearly defining functional requirements, non-functional requirements, and outlining unambiguous and complete system models.

TAEval is a client-server application that is designed to optimize the line of communication between an instructor and his or her teaching assistants by automating the issuing and tracking of tasks, task evaluations, and metrics that can quantify the TA's body of work.

2.2 Functional Requirements

Functional requirements are the concise, explicit details of what the system will be able to do with respect to functionality. These clearly and unambiguously define what objects can or cannot do in the system, and by virtue of that, they also clearly define what each object cannot do in the system. They set the expectations for object functionality for the main actors; TA, Instructor, and Administrator.

Below is a table outlining each functional requirement by traceability code and its description

Table 1 – Functional Requirements

Traceability Code	Functional Requirement
FR-00	TAs must be able to view their assigned tasks assigned by the course instructor.
FR-01	TAs must be able to view their tasks' respective evaluation evaluated by the course instructor.
FR-02	TAs must only be assigned to a maximum of one course at any given time.
FR-03	Instructors must be able to create a task at the beginning of the term for each TA for each class they are instructing.
FR-04	Instructors must be able to edit their existing delegated tasks.
FR-05	Instructors must be able to delete their existing delegated tasks.
FR-06	Instructors must be able to enter evaluation data for each existing delegated task. The evaluation scheme is 1-> 'poor', 2-> 'fair', 3-> 'good', 4-> 'very good', 5-> 'excellent'
FR-07	Instructors must be able to view a list of courses they are instructing in a specific term.
FR-08	Instructors must be able to view the list of TAs that are assigned to a specific course they are instructing.

FR-09	Instructors must be able to view a list of tasks they have created per course they are instructing.
FR-10	A course must have an existing instructor associated with it upon its creation.
FR-11	Administrators must be able to run reports on TA evaluation data, such as: TA evaluation ratings for one TA spanning all terms, TA evaluation ratings for all TAs spanning one term, TA evaluation ratings for all TAs for a particular course offering, specific TA evaluation ratings (such as only 'poor', or only 'excellent') for all TAs spanning all terms.
FR-12	Administrators must be able to view a list of courses offered in a given term.
FR-13	Administrators must be able to view a complete list of all instructors.
FR-14	Administrators must be able to view a complete list of all TAs.
FR-15	Administrators must be able to add course offerings.
FR-16	Administrators must be able to edit course offerings.
FR-17	Administrators must be able to delete course offerings.
FR-18	Administrators must be able to add instructors.
FR-19	Administrators must be able to edit instructors.
FR-20	Administrators must be able to delete instructors.
FR-21	Administrators must be able to add TAs.
FR-22	Administrators must be able to edit TAs.
FR-23	Administrators must be able to delete TAs.
FR-24	Administrators must be able to assign existing TAs to any existing course at any time.

2.3 Non-functional Requirements

Non-functional requirements are user-visible constraints on the system unrelated to the system's functional behaviour. These requirements do not pertain to the functionality of the system, but rather, they focus on the quality of our product. They are broken up into categories that describe commonality between them, such as user-friendliness, the reliability, the performance, maintainability, implementation related, how it operates, legal responsibilities, and delivery of the final product. In the context of TAEval, they represent a compromise between the client's requests and the development team's own set of standards for the same categories. Constraints can range from setting restrictions to the development of the system to self-imposed restrictions on what is deemed acceptable for commercial software.

Below is a table outlining each non-functional requirement by traceability code, category, and description.

Table 2 – Non-functional Requirements

Traceability Code	Type of NFR	Non-functional Requirement
NFR-01	Usability	TAEval user interface must be graphical in nature.
NFR-02	Usability	TAEval system must be easy to navigate via menu items and dialog boxes.
NFR-03	Usability	TAEval user interface must have a professional look and feel that is consistent with other commercial UI.
NFR-04	Usability	TAEval generated reports must be concise, consisting of summarized evaluation data, formatted as a single line per record.
NFR-05	Usability	Each client process must execute on a different machine and support a single user.
NFR-06	Usability	Data requested by user must be handled by the TAEval client which queries the central server, accessible at a configurable IP address, to populate the user's client UI.
NFR-07	Usability	All fields for user text input must have an upper limit that cannot be exceeded.
NFR-08	Usability	All save operations must be confirmed by the user.
NFR-09	Usability	All delete operations must be confirmed by the user.
NFR-10	Usability	TAEval user interface must have the same color scheme that Carleton University uses.
NFR-11	Usability	Explicit documentation on how to install and configure TAEval should be provided
NFR-12	Reliability	All exceptions should be handled gracefully with appropriately detailed error messages
NFR-13	Reliability	If TAEval crashes while an operation leading to a change in the database is occurring, the change must be halted and removed and the system should offer to restore itself to the last safe state.
NFR-14	Performance	User must be able to view up to date information on the client UI instantly.
NFR-15	Performance	There should be no duplication of data anywhere in the system.
NFR-16	Supportability	TAEval must be built to run on a lightweight client such as a mobile device in a future phase.
NFR-17	Supportability	TAEval must be able to support a minimum of four concurrent processes, each on a different host.
NFR-18	Supportability	The system should be extensible to any GUI platform with minimal work required to port over to another.
NFR-19	Implementation	All processes must work on the Linux Ubuntu 12.04 platform.
NFR-20	Implementation	Source code must be written in C++.
NFR-21	Implementation	Data storage organization must be designed for ease of

		retrieval and efficient use of storage space.
NFR-22	Implementation	Data must be stored in SQLite.
NFR-23	Implementation	Client processes must communicate with the central server using TCP/IP sockets.
NFR-24	Interface	Every user must be running a separate client process which provides the TAEval UI.
NFR-25	Operations	Client must be designed to use very little memory and must have no persistent storage.
NFR-26	Operations	All data must be stored centrally on a single host.
NFR-27	Operations	Server process must execute on central host and must manage updates and retrievals of the data.
NFR-28	Operations	Queries to the server must return only the minimum amount of necessary data.
NFR-29	Operations	Almost no data should be stored on the client when the user moves between UI screens.
NFR-30	Operations	No client processes will run on the central server host.
NFR-31	Packaging	The product must be delivered in a CD-ROM/DVD with everything necessary to install the program.
NFR-32	Legal	All administrators must agree for all sensitive information to be kept confidential.

2.4 System Models

The system models describe the use cases, object models, and dynamic models for the TAEval system. They function as an iterative way of defining requirements that need to be satisfied by the system, identifying missing requirements, and refining the existing ones.

We are providing a use case model, object model and dynamic model. The use case model is composed of use case diagrams, use case descriptions, and a traceability matrix. The scope of the use case model is for all TA and Instructor actors use cases. The object model identifies the entities, boundary entities and controls of the system. The object model is composed of a data dictionary and a class diagram. The scope of the object model is limited to only the entity objects. The dynamic models illustrate the behaviour of the system. The dynamic model is composed of state machine diagrams for the entity objects and sequence diagrams for the instructor use cases.

2.4.1 Use Case Model

Use case models are used to model high-level functionality, at a level that is relevant to the user. For our system, we have created use cases to model actions that the Instructor and TA users would perform in conjunction with the TAEval system. Below we begin by giving an overview of all of the high-level use cases, followed by the detailed use cases with their assigned traceability codes and given names. Following the overview, we present tables containing use case descriptions that outline the how we define each use case with respect to traceability code, name, flow of events, entry and exit conditions, quality requirements, and traceability to functional or non-functional requirements that we have previously defined.

Use Case Overview

Limiting our scope to only the TA and Instructor actors, we have two high level use cases: BrowseOwnTasks and ManageTasks.

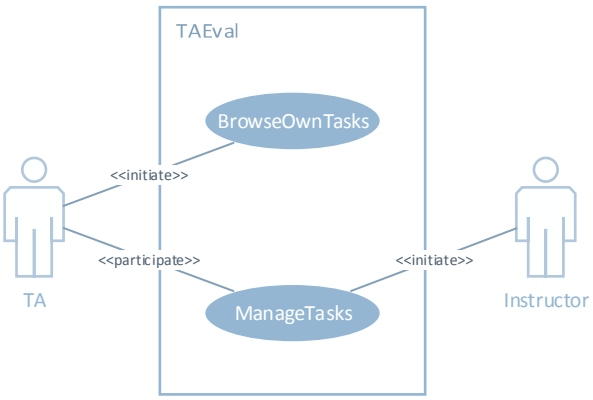

Figure 1. High-level use case diagram

Table 3 – High-level Use Case Descriptions

Traceability Code	Use Case Name	Use Case Description
UC-01	BrowseOwnTasks	The TA browses the tasks assigned to him or her by course
UC-02	ManageTasks	The Instructor manages selected properties for all tasks

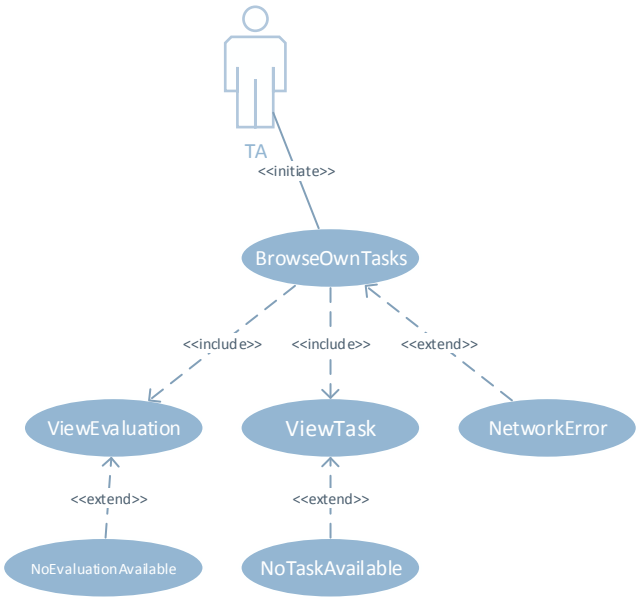

Figure 2. BrowseOwnTasks detailed use case diagram

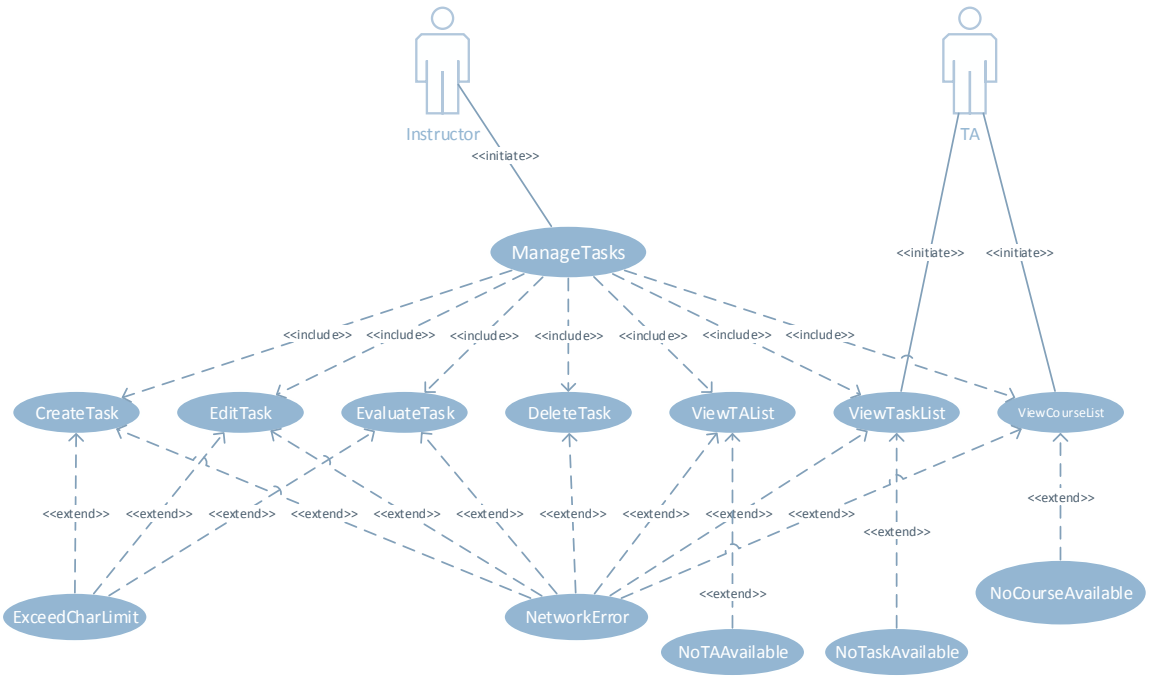

Figure 3. ManageTasks detailed use case diagram

Table 4 – Detailed Use Case Descriptions

Traceability Code	Use Case Name	Use Case Description
UC-03	ViewTask	The TA views a particular instance of a task
UC-04	ViewTaskEvaluation	The TA views the evaluation for a particular task
UC-05	ViewCourseList	The Instructor or TA views a list of courses they are registered to by a term that he or she chooses
UC-06	ViewTaskList	The Instructor or TA views a list of tasks relating to them for a course that he or she chooses
UC-07	ViewTaList	The Instructor views a list of all TAs for a particular course he or she is instructing
UC-08	CreateTask	The Instructor creates a task
UC-09	EditTask	The Instructor edits a selected task
UC-10	DeleteTask	The Instructor deletes a selected task
UC-11	EvaluateTask	The Instructor evaluates a particular existing task
UC-12	NetworkError	The system reports that the submitted form could not be received
UC-13	ExceedCharLimit	The system prompts the user that the text input given exceeds the limit
UC-14	RepositoryModificationError	The system prompts the user that the request could not be completed because the system was modified
UC-15	NoTaskAvailable	The system prompts the user that there are no associated tasks to be found
UC-16	NoCoursesAvailable	The system prompts the user that there are no associated courses to be found
UC-17	NoEvaluationAvailable	The system prompts the user that there are no evaluations associated with the TA
UC-18	NoTAAvailable	The system prompts the user that there are no TAs to select right now

Use Case Flow of Events

The following tables organize the layout for a detailed description of each use case. Each table will list the actors that participate in the use case followed by a flow of events that illustrate how the use case proceeds through each step, from being invoked to ending. The entry condition is defined as the condition(s) that must be met for the use case to be invoked. The exit condition is defined as the condition(s) that the system must be left in once the use case terminates. Finally, we list quality requirements pertaining to each use case and the functional or non-functional requirements that are fulfilled.

Use Case Identifier	UC-01
Name	**BrowseOwnTasks**
Participating Actors	Initiated by TA
Flow of Events	1. The system displays the list of all courses the TA is or has been involved in with a selectable option to filter by a specific semester and a menu with the following options: view a list of courses, view a list of tasks. 2. If the TA opts to view a list of courses the system displays a list of courses optionally filtered by semester (initiate use case **ViewCourseList**). 3. The TA selects a course from the course list. 4. If the TA opts to view a list of tasks the system displays a list of tasks associated with the TA's selected course (initiate use case **ViewTaskList**).
Entry Conditions	User logged in to TAEval as a TA
Exit Conditions	
Quality Requirements	• TAEval will take no longer than 5 seconds to return list of available courses to the TA.
Traceability	FR-00 and FR-01

Use Case Identifier	UC-02
Name	**ManageTasks**
Participating Actors	Initiated by Instructor Participated by TA
Flow of Events	1. The system displays the list of all courses the Instructor is teaching and has taught with a selectable option to filter by a specific semester and a menu with the following options: create a new task, view a list of courses, view a list of TAs, and view a list of tasks. 2. If the Instructor opts to create a new task a task creation form is displayed (include use case **CreateTask**). 3. If the Instructor opts to view a list of courses the system displays a list of courses the Instructor is teaching optionally

	filtered by a specific semester (include use case **ViewCourseList**). 4. The Instructor selects a course from the list of courses. 5. If the Instructor opts to view a list of TAs the system displays a list of TAs assigned to the Instructor of the Instructor's selected course (include use case **ViewTAList**). 6. If the Instructor opts to view a list of tasks the system displays a list of all tasks the Instructor has created for the selected course (include use case **ViewTaskList**).
Entry Conditions	User logged in to TAEval as an Instructor
Exit Conditions	
Quality Requirements	• The system should respond to requests in no more than 10 seconds.
Traceability	FR-03, FR-04, FR-05, FR-06, FR-07, FR-08, FR-09

Use Case Identifier	UC-03
Name	**ViewTask**
Participating Actors	Initiated by TA
Flow of Events	1. TA selects a single task they want to view. 2. TAEval will return the requested task. 3. TA reviews task.
Entry Conditions	TA has received list of tasks for a specified course from BrowseOwnTasks
Exit Conditions	TA has received and reviewed their task
Quality Requirements	TAEval will take no longer than 5 seconds to return requested task to the TA.
Traceability	FR-00

Use Case Identifier	UC-04
Name	**ViewTaskEvaluation**
Participating Actors	Initiated by TA
Flow of Events	1. TA selects a single task they want to view that has been evaluated 2. TAEval will return the evaluated requested task. 3. TA reviews evaluated task.
Entry Conditions	TA has received list of tasks for a specified course from BrowseOwnTasks
Exit Conditions	TA has received and reviewed their evaluation
Quality Requirements	TAEval will take no longer than 5 seconds to return requested evaluation to the TA.
Traceability	FR-01

Use Case Identifier	UC-05
Name	**ViewCourseList**
Participating Actors	Initiated by Instructor or TA
Flow of Events	1. The user requests to view the course list that they currently instructor have previously instructed, for the instructor, or have tasks currently or previously assigned, for the TA. 2. TAEval will return the list of courses for the specific user. The TA will receive the list of courses that they have tasks currently or previously assigned. The instructor will receive the course that they have currently or previously instructed. 3. The Instructor or TA reviews listed courses.
Entry Conditions	User is logged in as Instructor (or TA?) to the TAEval system
Exit Conditions	User has received and reviewed their listed courses
Quality Requirements	• TAEval will take no longer than 5 seconds to return requested course list to the user. • The list of all courses displayed should be sorted alphanumerically by course code and shown grouped in descending order by term.
Traceability	FR-07

Use Case Identifier	UC-06
Name	**ViewTaskList**
Participating Actors	Initiated by Instructor or TA
Flow of Events	1. The user requests to view the list of tasks that they have created, for the instructor, or list of tasks that are assigned, for the TA. 2. TAEval will return the list of tasks for the specific user. The TA will receive the list of task they have been assigned. The instructor will receive the list of tasks that they have created. 3. The Instructor or TA reviews listed tasks.
Entry Conditions	User is logged in as an Instructor or TA to TAEval
Exit Conditions	User has received and reviewed their list of tasks
Quality Requirements	• TAEval will take no longer than 5 seconds to return the requested task list • The list of tasks should be sorted alphabetically by task name.
Traceability	FR-00, FR-09

Use Case Identifier	UC-07
Name	**ViewTAList**
Participating Actors	Initiated by Instructor
Flow of Events	1. The user requests to view the TA list 2. The TA list is returned and displayed to the user
Entry Conditions	User is logged in as an Instructor and a course for which to view the TA list has been selected
Exit Conditions	The list of TAs for the specific course is displayed to the user
Quality Requirements	• The TA list is displayed within 5 seconds of the request • The list of TAs should be sorted alphabetically by their last names.
Traceability	FR-08

Use Case Identifier	UC-08
Name	**CreateTask**
Participating Actors	Initiated by Instructor
Flow of Events	1. Instructor selects the create task option. 2. If one or more TA's are selected from TA List, then their names are added to the assigned TA's. 3. Instructor is prompted to input task name, task description, and assign additional TA's 4. Instructor submits the form. Instructor waits for confirmation. 5. TAEval receives form submission and notifies instructor.
Entry Conditions	User is logged into TAEval as an Instructor
Exit Conditions	• Instructor receives notification of task creation success. • Task list updated with the created task name. --OR-- • Instructor cancels form submission and nothing is updated.
Quality Requirements	Instructor's form submission is received and the Instructor is taken to the previous menu after no longer than 5 seconds.
Traceability	FR-03

Use Case Identifier	UC-09
Name	**EditTask**
Participating Actors	Initiated by Instructor
Flow of Events	1. Instructor requests the editing of a task. 2. The system presents the Instructor with a form for the task, with all of the current attributes of the task in place. 3. The instructor changes one or more of the task name, task description, or task assignee. 4. Instructor submits the form and waits for a response. 5. If a task name or task description is entered, the system checks that they have not both crossed their respective upper limit for number of characters. If the input is verified as acceptable, the system edits the task as the Instructor requested and notifies the instructor.
Entry Conditions	User is logged into TAEval as an Instructor User has selected a task from the task list
Exit Conditions	• Instructor receives notification of task editing success. • Task list updated with the edited task name. --OR-- • Instructor cancels form submission and nothing is updated.
Quality Requirements	• Instructor's form submission is receive • Instructor is taken to the previous menu after no longer than 5 seconds.
Traceability	FR-04

Use Case Identifier	UC-10
Name	**DeleteTask**
Participating Actors	Initiated by Instructor
Flow of Events	1. Instructor requests the deletion of a task. 2. The system prompts the Instructor with a confirmation box, asking if the Instructor is sure they want to delete the task and that the changes cannot be reverted. 3. If the Instructor selects 'OK', then the system removes the entire task and any associations to it from the database and notifies the Instructor that the deletion was successful. 4. If the Instructor selects 'Cancel', then the system doesn't act further.
Entry Conditions	User is logged in to TAEval as an Instructor. User has selected a task from the task list
Exit Conditions	• Instructor receives notification of task deletion success. • Task list updated with the deleted task removed

	--OR-- • Instructor cancels upon prompt and nothing is updated.
Quality Requirements	Instructor's form submission is received and the Instructor is taken to the previous menu after no longer than 5 seconds.
Traceability	FR-05

Use Case Identifier	UC-11
Name	**EvaluateTask**
Participating Actors	Initiated by Instructor
Flow of Events	1. The Instructor selects the evaluate task option. 2. The system displays a task evaluation form to the Instructor. 3. The Instructor specifies a rating from 1-5 -- with 1 being the worst and 5 being the best -- then leaves textual feedback and submits the form. 4. The system updates the task with the Instructor's evaluation, sends a notification of success, and returns the Instructor to the previous menu.
Entry Conditions	The Instructor has selected a task to evaluate from a list of tasks.
Exit Conditions	• The selected task has its evaluation data updated to reflect the Instructor's evaluation OR • The Instructor has cancelled the evaluation and the task's evaluation data is left unchanged.
Quality Requirements	The system should respond to requests in no more than 5 seconds.
Traceability	FR-06

Use Case Identifier	UC-12
Name	**NetworkError**
Participating Actors	Communicates with Instructor and TA
Flow of Events	1. The TAEval system encounters a network error (timeout, no connection, etc.). 2. The TAEval client will notify user about network error and will allow user to try their request again.
Entry Conditions	• This use case extends the BrowseOwnTasks, ManageTasks, ViewTaskList, ViewTaskEvaluation, ViewCourseList, and ViewTAList, CreateTask, EditTask, DeleteTask and EvaluateTask use cases. • It is initiated whenever a network error is encountered between the client and server.
Exit Conditions	• User receives a notification about network error
Quality Requirements	The user receives a notification in no more than 5 seconds after the network error has affected the system.
Traceability	NFR-12

Use Case Identifier	UC-13
Name	**ExceedCharLimit**
Participating Actors	Communicates with Instructor and TA
Flow of Events	1. The user tries to input more characters into a text field than the maximum allowed amount for that particular field. 2. The TAEval system prevents the user from inputting any more characters into the field unless the number of characters exceeding the upper limit is removed.
Entry Conditions	• This use case extends the CreateTask, EditTask, EvaluateTask use cases. • It is initiated whenever the user attempts to input more than the maximum number of characters allowed into a text field
Exit Conditions	• The TAEval system prevents the user from inputting any more characters
Quality Requirements	The user will not be interrupted by this exception
Traceability	NFR-12, NFR-07

Use Case Identifier	UC-14
Name	**RepositoryModificationError**
Participating Actors	Communicates with Instructor and TA
Flow of Events	1. The user is performing a task involving some data they have received from the TAEval repository on their last request. 2. The user performs a new request dependent on the data they received from their last request. 3. The new request tries to access the repository referencing the dependent data after it has already been modified by another user. 4. The TAEval system alerts the user that their request could not be completed because of a recent modification to the repository that left it in a state different to what they are expecting.
Entry Conditions	• This use case extends the ViewTask, ViewTaskEvaluation, ViewCourseList, ViewTaskList, ViewTAList, CreateTask, EditTask, DeleteTask and EvaluateTask use cases. • It is initiated whenever the repository no longer exists in the state that the user is expecting.
Exit Conditions	• User receives notification that their request could not be completed because the system was modified
Quality Requirements	• The user receives the notification in no more than 5 seconds • The user will not lose any client-side changes they have made
Traceability	NFR-12

Use Case Identifier	UC-15
Name	**NoTaskAvailable**
Participating Actors	Communicates with TA
Flow of Events	1. The TAEval system searches repository to find tasks associated with logged in TA. 2. The TAEval server notifies TAEval client about zero tasks being associated with the TA. 3. The TAEval client receives notification that no tasks are associated with the TA.
Entry Conditions	• This use case extends the ViewTask. • It is initiated whenever no tasks are found for a particular TA
Exit Conditions	• TA receives a notification that they do not have any tasks associated with them.
Quality Requirements	• The user receives the notification in no more than 5 seconds
Traceability	NFR-12

Use Case Identifier	UC-16
Name	**NoCoursesAvailable**
Participating Actors	Communicates with Instructor and TA
Flow of Events	1. The TAEval system searches repository to find courses associated with logged in user. 2. The TAEval server notifies TAEval client about zero courses being associated with the user. 3. The TAEval client receives notification that no courses are associated with the user.
Entry Conditions	• This use case extends the ViewCourseList. • It is initiated whenever no courses are found for a particular user
Exit Conditions	• User receives a notification that they do not have any tasks associated with them.
Quality Requirements	• The user receives the notification in no more than 5 seconds
Traceability	NFR-12

Use Case Identifier	UC-17
Name	**NoEvaluationAvailable**
Participating Actors	Communicates with TA
Flow of Events	1. The TAEval system searches repository to find evaluations associated with logged in TA. 2. The TAEval server notifies TAEval client about zero evaluations being associated with the TA. 3. The TAEval client receives notification that no tasks are associated with the TA.
Entry Conditions	• This use case extends the ViewEvaluation. • It is initiated whenever no evaluations are found for a particular TA
Exit Conditions	• TA receives a notification that they do not have any evaluations associated with them.
Quality Requirements	• The user receives the notification in no more than 5 seconds
Traceability	NFR-12

Use Case Identifier	UC-18
Name	**NoTAAvailable**
Participating Actors	Communicates with Instructor
Flow of Events	1. The TAEval system searches repository to find the selected TA. 2. The TAEval server notifies TAEval client about zero TAs in the database. 3. The TAEval client receives notification that no TAs are in the system to be selected.
Entry Conditions	• This use case extends the ViewTAList. • It is initiated whenever no TA are found for a particular course
Exit Conditions	• Instructor receives a notification that the system could not find any TAs associated with them.
Quality Requirements	• The user receives the notification in no more than 5 seconds
Traceability	NFR-12

2.4.2 Object Model

The object model details the tangible things from the real world that are modeled by the system and how they are associated with either each other or intangible things from the system. The distinction is made here between which objects are:

- based on the application domain -- known as entity objects
- which are necessary for user-system interaction -- boundary objects, and
- which are used to manipulate them -- control objects.

Data Dictionary

The data dictionary is a table that contains a formal name of the object, its respective attributes and associations, the definition of the object in the context of our system, and the use cases that the object traces back to.

Entity Object	Attributes and Associations	Definition	Traceability
Admin	- name	The highest level of user supported by the TAEval system. Administrators are responsible for adding, removing and editing courses, instructors and TAs in the system.	
Course	- name - term - code - tasks - TAs - instructor	A course is an instructional period for students in a given term. A course is taught by an instructor and may also have TAs assigned to it.	UC-01, UC-02, UC-03, UC-04, UC-05,UC-07, UC-12, UC-14, UC-16
Evaluation	- rating - feedback	An evaluation is the communication between an instructor and a TA for a completed task. An evaluation is created by the instructor for a completed task and consists of a numerical rating and written feedback.	UC-01, UC-02, UC-03, UC-04, UC-05,UC-07, UC-12, UC-14, UC-16

Instructor	- name - department - courses	An instructor is responsible for teaching one or more courses.	UC-01, UC-02, UC-03, UC-04, UC-05,UC-07, UC-12, UC-14, UC-16, UC-18
TA	- name - year - degree - major - course - studentId	A TA (Teaching Assistant) is assigned to a course and performs duties to help the instructor. TAs may be responsible for grading assignments, administering tests and meeting with students to offer help pertaining to course material.	UC-01 to UC-08, UC-12 to UC-18
TAEval	- admins - instructors - TAs - courses	TAEval is a system that facilitates and improves upon Instructor-TA communication by tracking tasks assigned and courses taught, administered by Administrators.	UC-01 to UC-18
Task	- name - description - TA - evaluation	A task is an assigned piece of work for a Course, given by an Instructor to a TA.	UC-01 to UC-15
User	- name	A user is someone who has credentials in the TAEval system to log in and use its functionality.	all UC

Class Diagram

The UML class diagram's main purpose is to graphically show the relationships shared between classes within the system. Because the object model's scope is limited to entity objects only, boundary and control objects' relationships will not be found in this diagram. Classes, which are definitions, within a system have many possible relationships: some may inherit from each other; some may have another class as an attribute. By including multiplicity and directionality, we can quickly arrive at a detailed big picture glance of the system's set of objects' interactions.

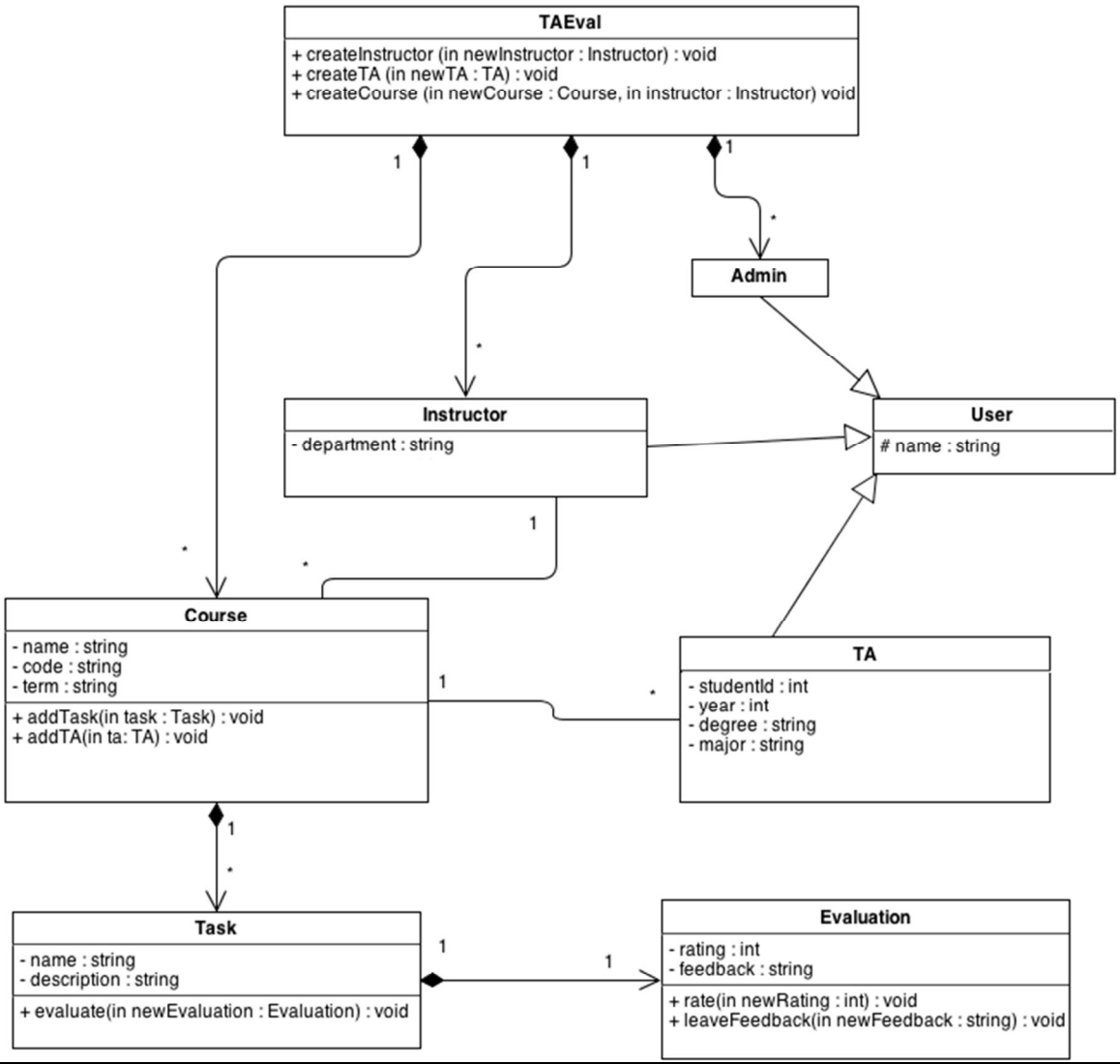

Figure 4. Class diagram

2.4.3 Dynamic Model

The dynamic model clarifies system behavior from an external point of view. The purpose of these models is to show the dynamic, as in ever-changing, behavior of our classes of objects (entity, boundary, control) within the system from a non-developers' perspective.

State Machines

State machine diagrams specifically show all possibly distinct states that a single object can go through, from its beginning state to its end state, where the end state is defined as the state at which, upon arriving, the object will never re-enter another state. They allow us to formally and visually map out the distinct states of an object while also allowing us to identify new behavior. A state is defined as a unique set of attributes the object maintains with respect to the system. For example, two distinct states that an Instructor may have is that, given he or she is defined with a name and department, he or she is either teaching or not teaching. The scope of our state machine diagrams is limited to the entity objects only, which were declared and defined in the data dictionary.

Below are the state machine diagrams for the following entities: Course, TA, Instructor, Task, Evaluation.

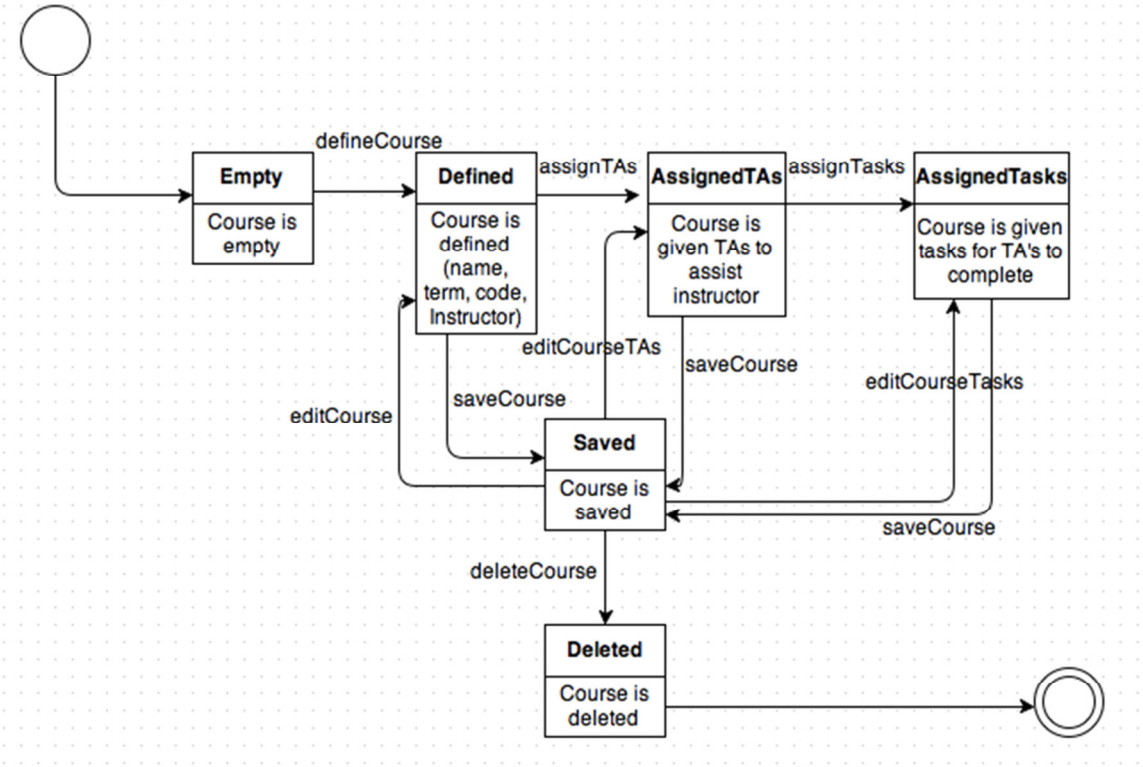

Figure 5. Course state machine

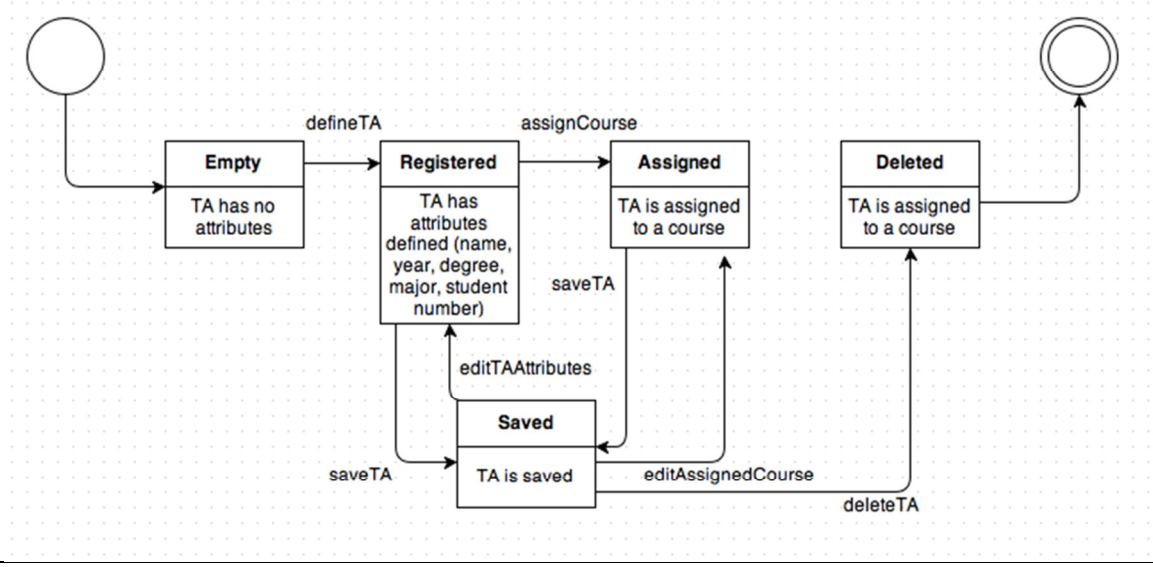

Figure 6. TA state machine

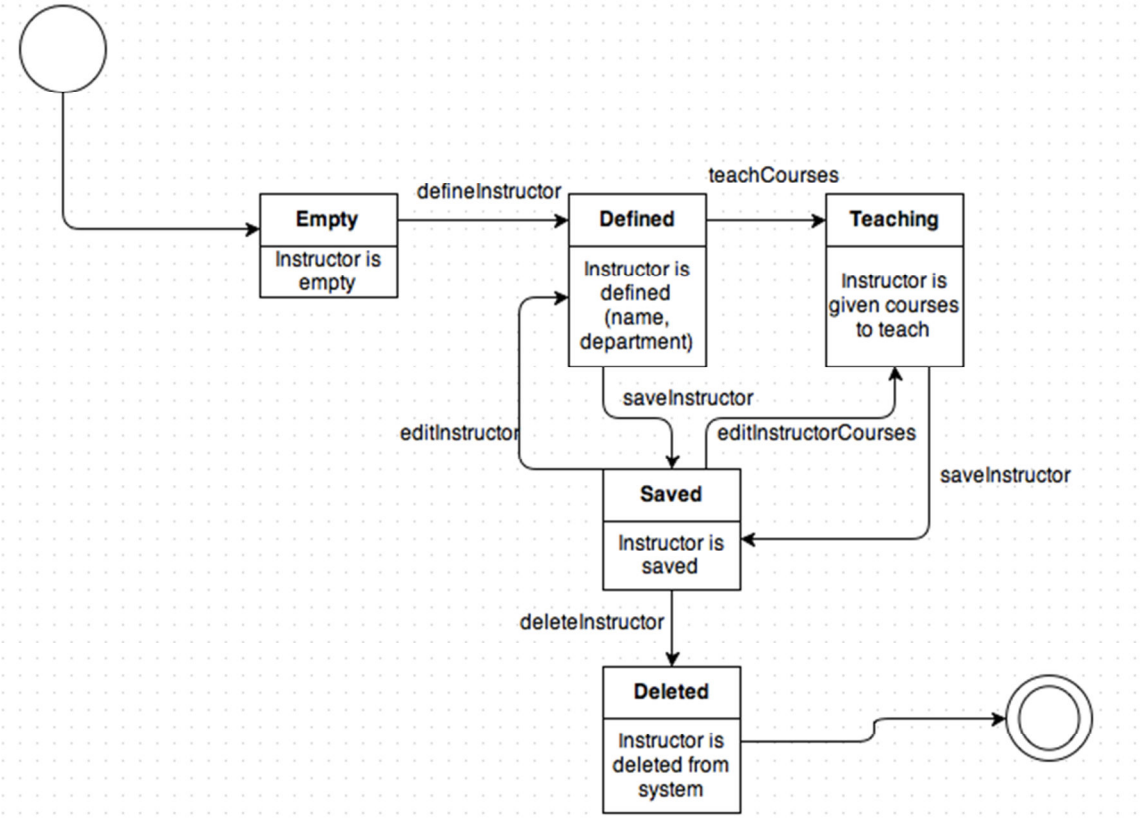

Figure 7. Instructor state machine

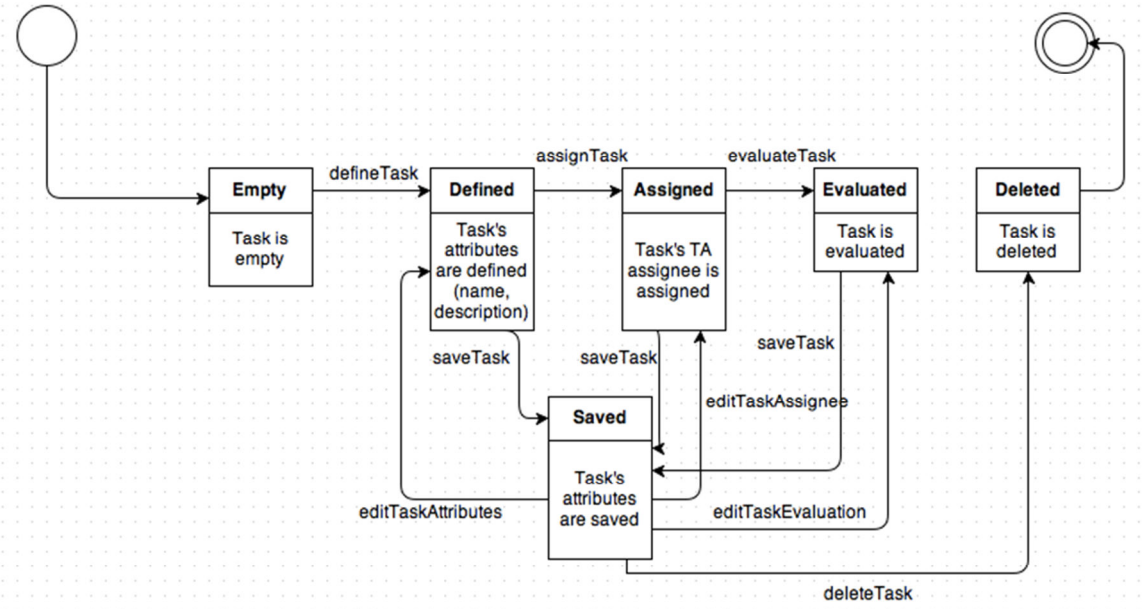

Figure 8. Task state machine

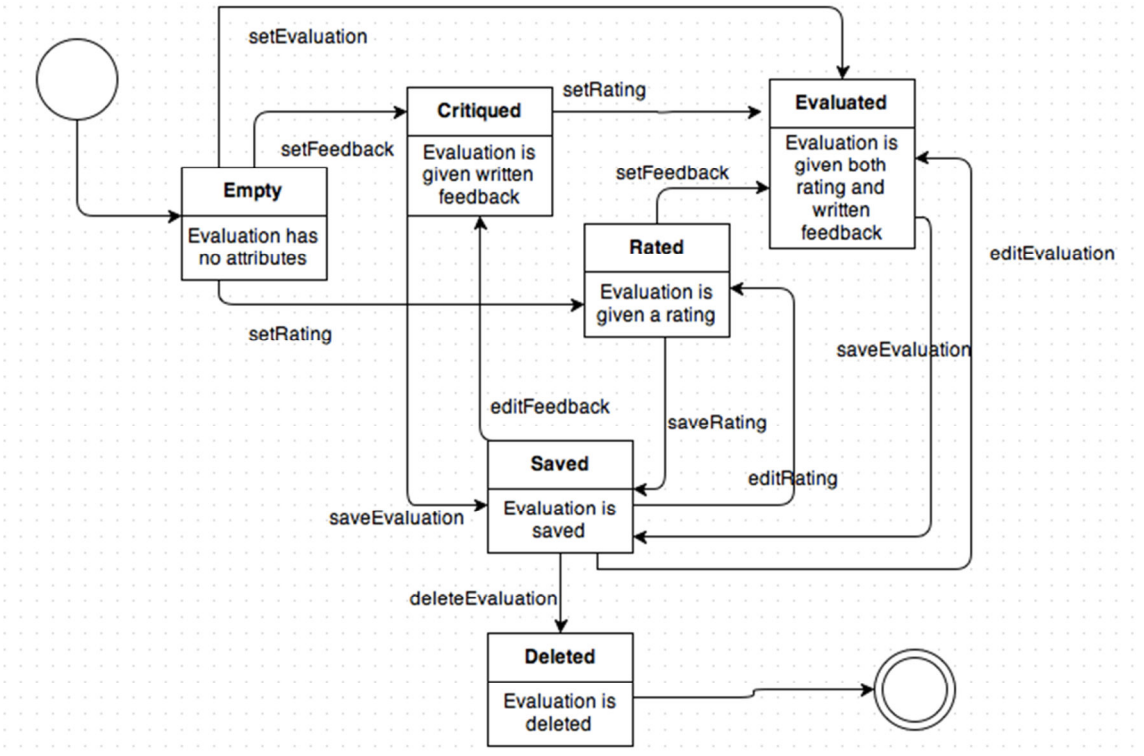

Figure 9. Evaluation state machine

Sequence Diagrams

Sequence diagrams also assist with capturing the system behavior by analyzing and visually diagramming how a particular use case sparks interaction between one or many objects within the system. This is accomplished by having the y-axis representing time elapsed as you descend from the top-down, showing the introductions and interactions of objects with the initial initiating actor in order of appearance. It allows us to see which objects create other objects, and when or whether certain objects terminate before the use case ends or not.

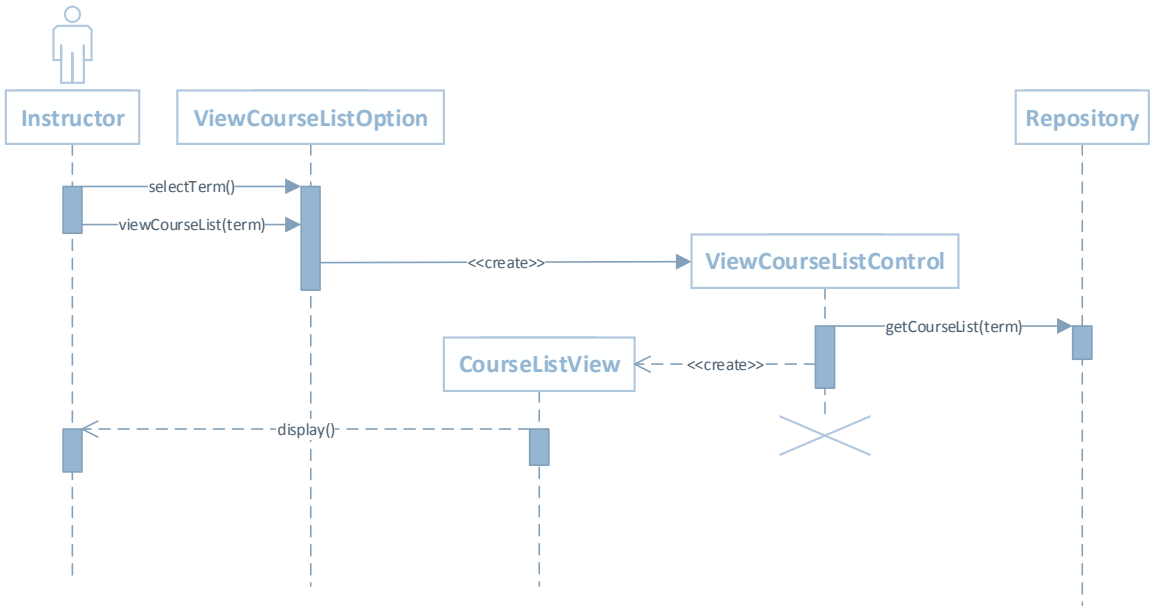

Figure 10. ViewCourseList sequence diagram
Traceability: UC-05

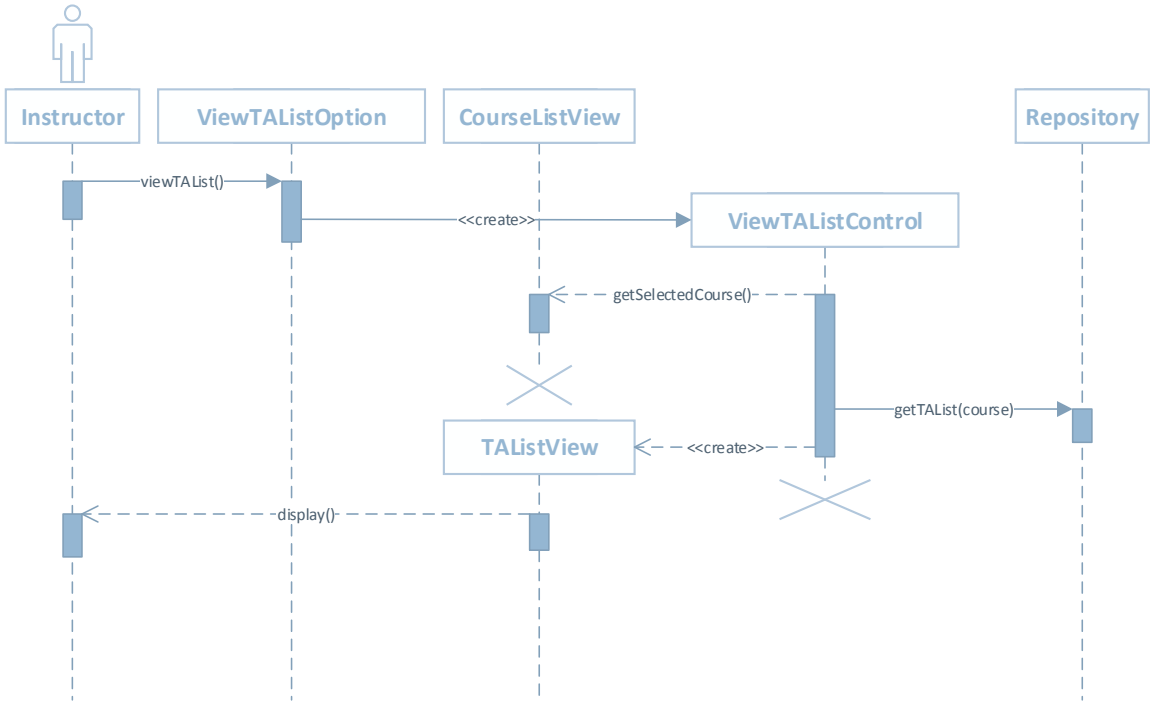

Figure 11. ViewTAList sequence diagram
Traceability: UC-07

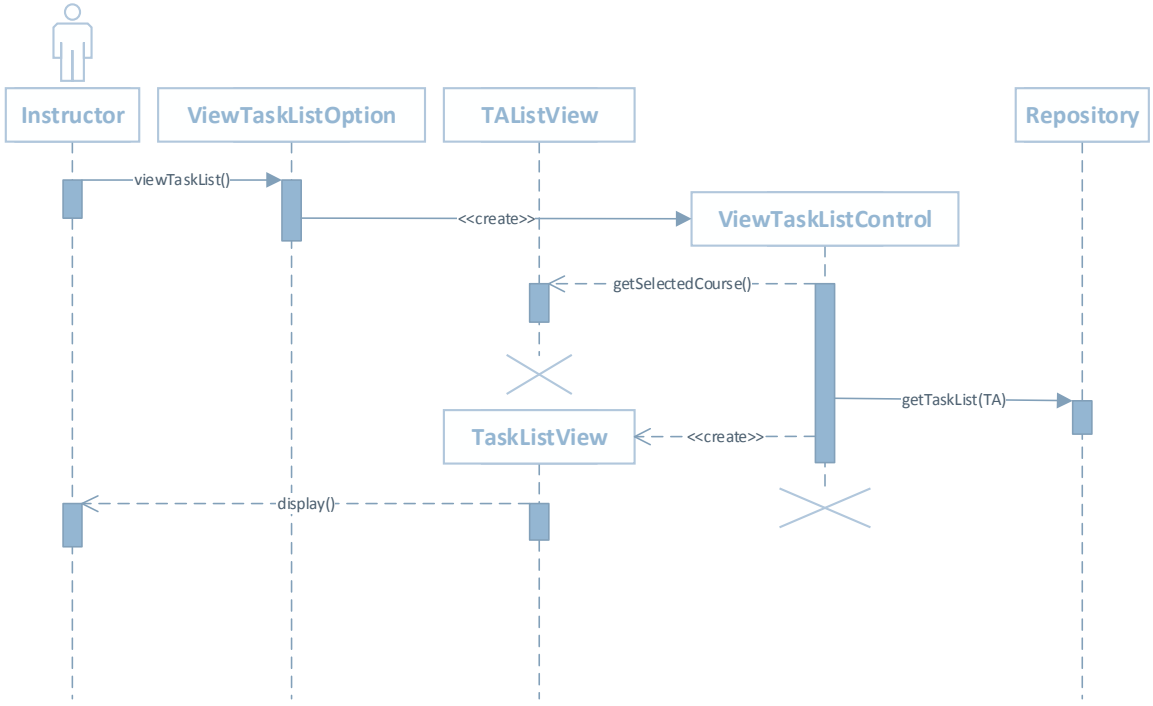

Figure 12. ViewTaskList sequence diagram
Traceability: UC-07

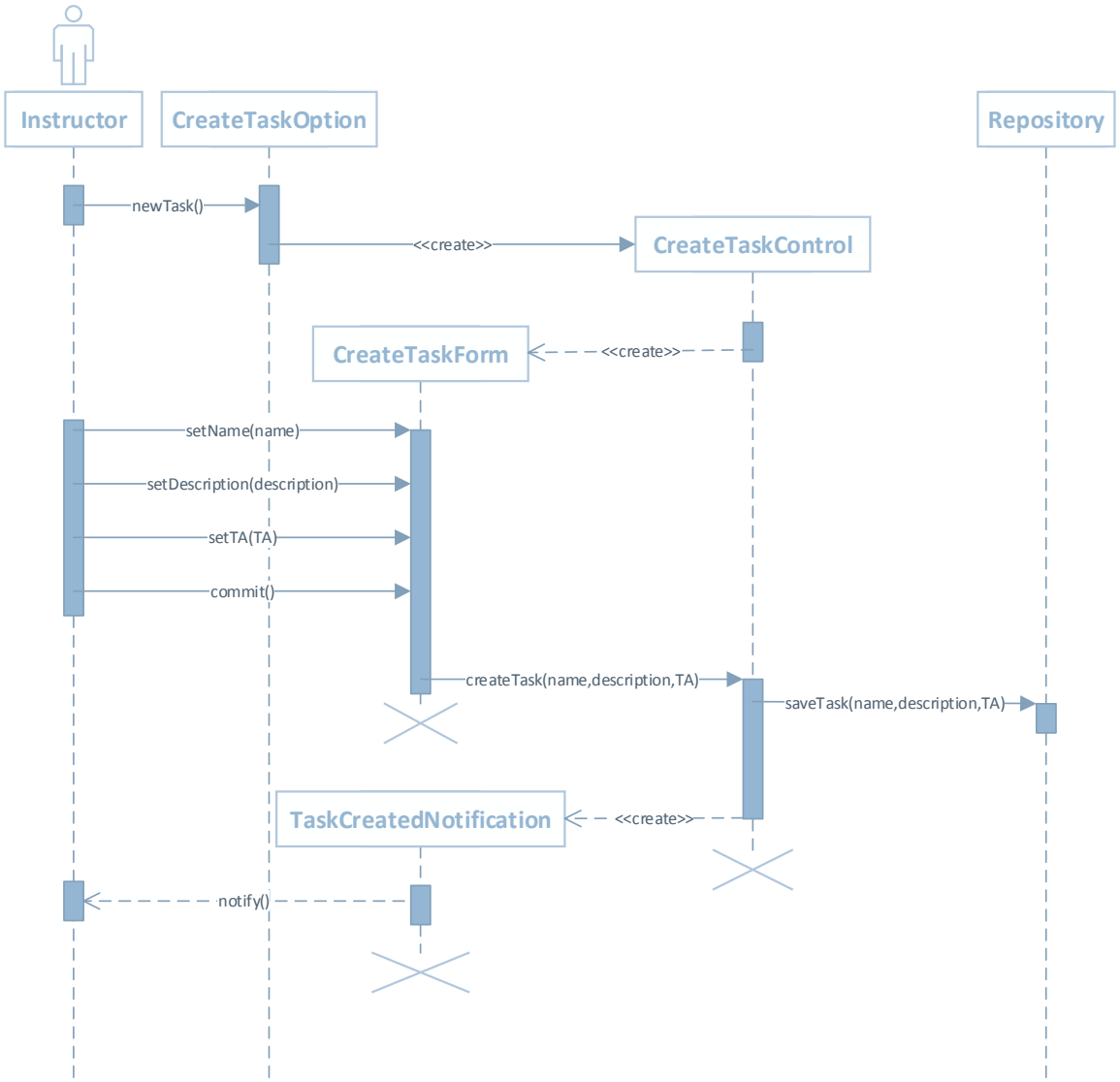

Figure 13. CreateTask sequence diagram
Traceability: UC-08

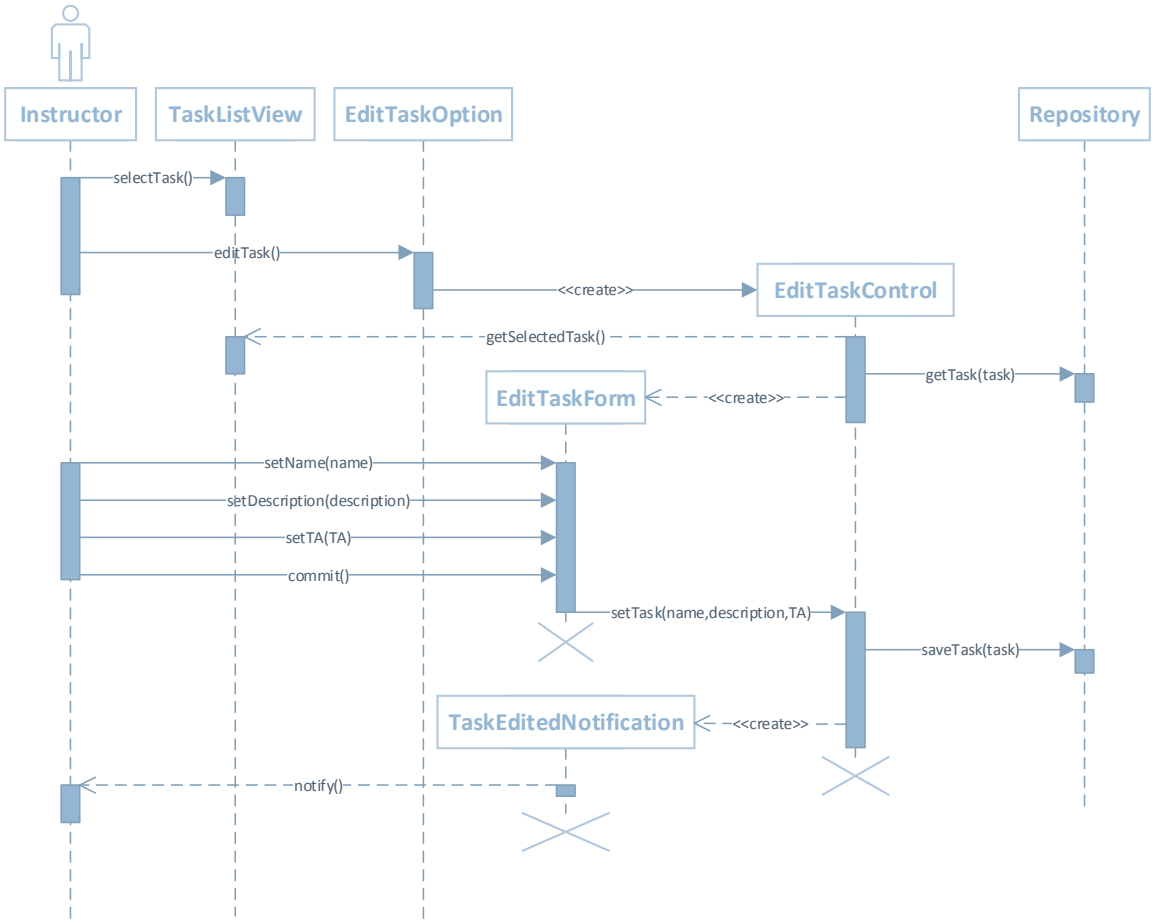

Figure 14. EditTask sequence diagram
Traceability: UC-09

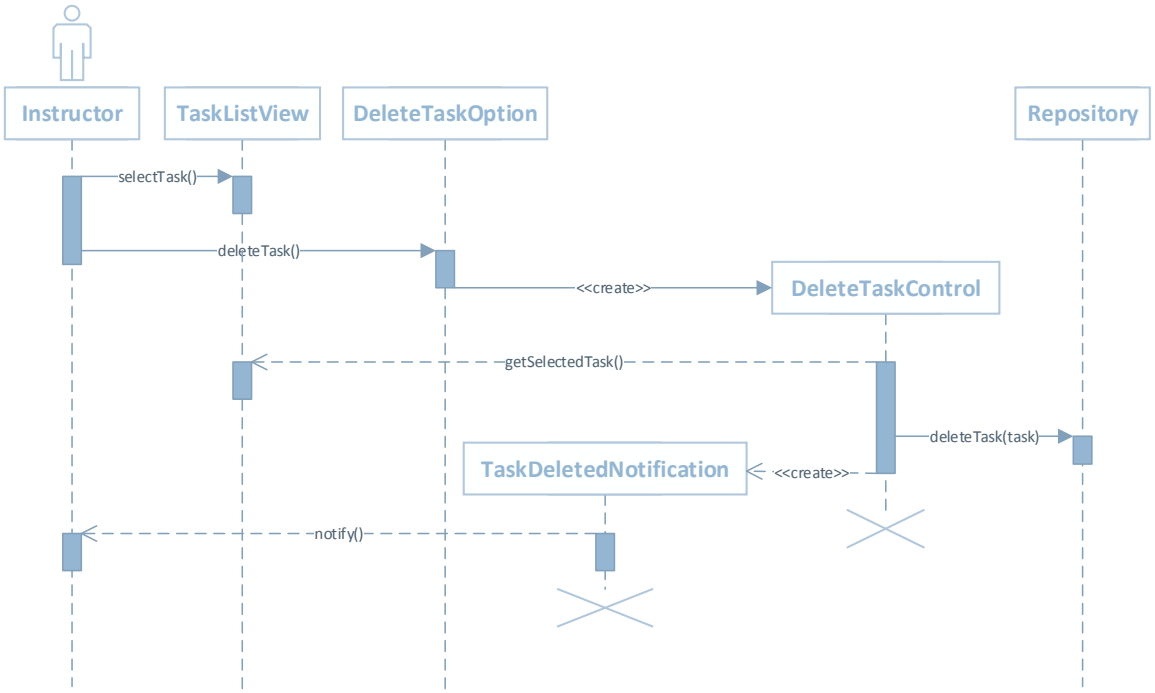

Figure 15. DeleteTask sequence diagram
Traceability: UC-10

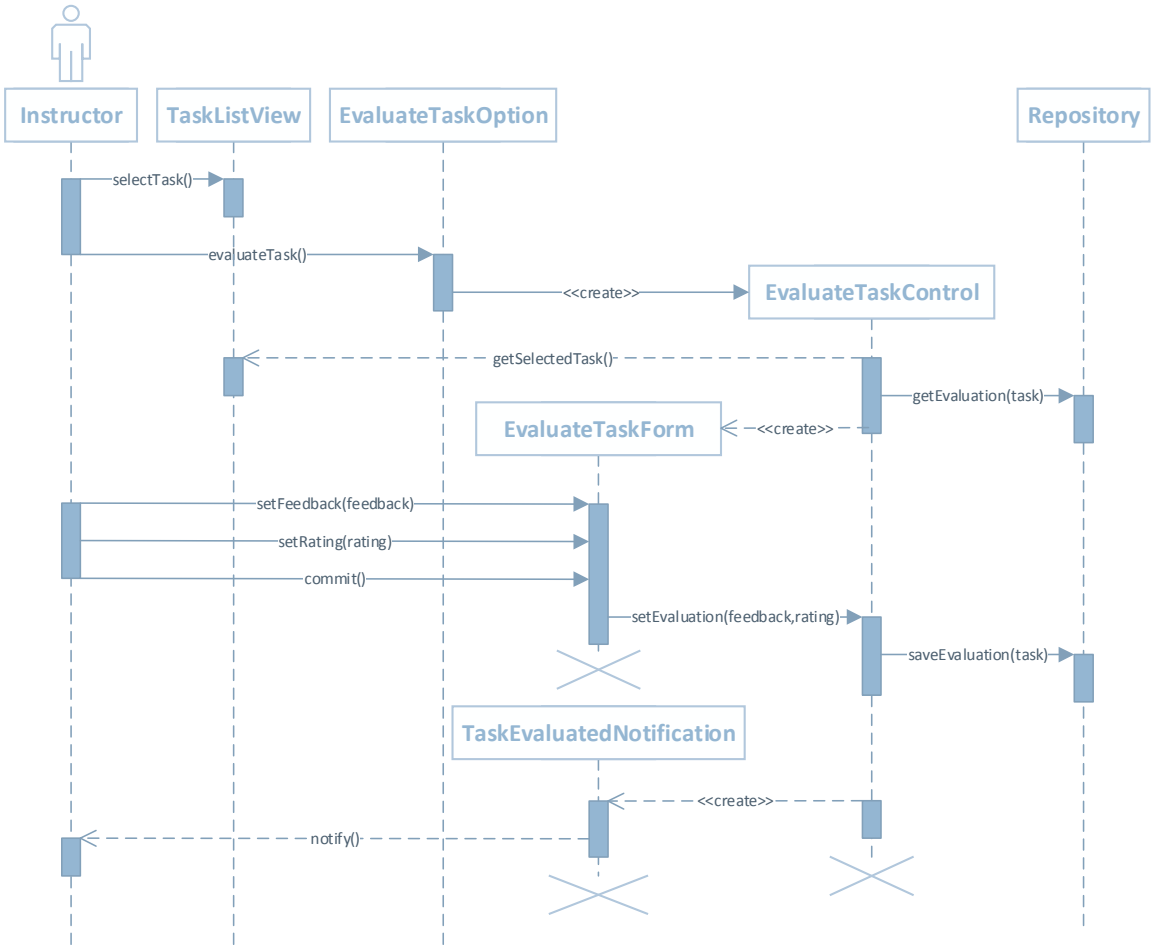

Figure 16. EvaluteTask sequence diagram
Traceability: UC-11