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

1. Introduction………………………………………………………#
   1. Purpose of System………………………………………#
   2. Overview of Document………………………………..#
2. Proposed System……………………………………………….#

2.1. Overview…………………………………………………….#

2.2. Functional Requirements……………………………..#

2.3. Non-functional Requirements………………………#

2.4. System Models……………………………………………..#

2.4.1. Use Case Models…………………………………#

2.4.2. Object Model……………………………………...#

2.4.3. Dynamic Model………………………………….#

1. Glossary…………………………………………………………….#

Figures

tbd

Tables

tbd

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 the 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 instructors user cases, and state machine diagrams that map only the entity objects.

1. Proposed System
   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.

* 1. Functional Requirements

Functional requirements are the concise, explicit details of what the system will be able to do with respect to functionality. For TAEval, for example, there is a distinct difference between the system allowing   
  
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. |
|  |  |

* 1. Non-functional Requirements

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. |

* 1. System Models

<explain what system models are in general, what models we provide in this section, & their purpose>  
2.4.1 Use Case Model  
<explain what a use case model is in the context of our system. Explain how this section is organized>

Use Case Overview

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

<insert detailed use case diagram for BrowseOwnTasks>

<insert detailed use case diagram for ManageTasks>

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 views a list of courses by a term that he or she chooses |
| UC-06 | ViewTaskList | The Instructor views a list of tasks they have created 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 | LackOfPrivilege | The system prompts the user to re-enter their login credentials (DUPLICATE of UC-14?) |
| UC-14 | InvalidCredentials | The system prompts the user to re-enter their login credentials |
| UC-15 | RequestTimeout | The system prompts the user of a timeout on the last request |
| UC-16 | ExceedCharLimit | The system prompts the user that the text input given exceeds the limit |
| UC-17 | SelectedTANotFound | The system prompts the Instructor that the TA entered is not found |
| UC-18 | NoTaskAvailable | The system prompts the user that there are no associated tasks to be found |
| UC-19 | NoCoursesAvailable | The system prompts the user that there are no associated courses to be found |
| UC-20 | NoEvaluationAvailable | The system prompts the user that there are no evaluations associated with the TA |

Use Case Flow of Events

<small blurb about what each of the tables below are/what they show in general>

|  |  |
| --- | --- |
| *Use Case Identifier* | UC-01 |
| *Name* | BrowseOwnTasks |
| *Participating Actors* | Initiated by TA |
| *Flow of Events* | 1. TA selects the BrowseOwnTasks option 2. TAEval responds with a list of previous terms the TA worked 3. TA selects the course that they are interested in 4. TAEval responds with the list of tasks for the selected course |
| *Entry Conditions* | User logged in to TAEval as a TA |
| *Exit Conditions* | TA receives a list of tasks for specified course |
| *Quality Requirements* | * TAEval will take no longer than 5 seconds to return list of available courses to the TA. * At any point during the flow of events, this use case can include the ViewCourseList use case. The ViewCourseList use case will be initiated when the TA invokes the ViewCourseList use case. When invoked it will list all the courses that the TA previously and currently has tasks assigned and/or evaluated. * At any point during the flow of events, this use case can include the ViewTaskList use case. The ViewTaskList use case will be initiated when the TA invokes the ViewTaskList use case. When invoked it will list all the Tasks that the TA previously and currently has tasks assigned and/or evaluated. |
| *Traceability* | FR-00 and FR-01 |

|  |  |
| --- | --- |
| *Use Case Identifier* | UC-02 |
| *Name* | ManageTasks |
| *Participating Actors* | Initiated by Instructor |
| *Flow of Events* | 1. The Instructor launches TAEval.  2. The system displays a menu with the following three login options: TA, Instructor, and Admin.  3. The Instructor selects the option to login as an Instructor.  4. The system displays the list of courses the Instructor is teaching and a menu with the following options: create a new task, edit a task, delete a task, evaluate a task, view a list of courses, view a list of TAs, and view a list of Tasks.  5. If the Instructor opts to create a new task a task creation form is displayed (include use case CreateTask).  \*6. If the Instructor opts to edit a task a list of existing tasks is displayed and the Instructor selects the task to be edited (include use case EditTask).  \*7. If the Instructor opts to delete a task a list of existing tasks is displayed and the Instructor selects the task to be deleted (include use case DeleteTask).  \*8. If the Instructor opts to evaluate a task a list of existing tasks is displayed and the Instructor selects the task to evaluate (include use case EvaluateTask).  8. If the Instructor opts to view a list of courses the system displays a list of courses the Instructor is teaching(include use case ViewCourseList).  9. If the Instructor opts to view a list of TAs the system displays a list of TAs assigned to the Instructor (include use case ViewTAList).  10. If the Instructor opts to view a list of tasks the system displays a list of tasks created by the Instructor (include use case ViewTasks). |
| *Entry Conditions* |  |
| *Exit Conditions* |  |
| *Quality Requirements* | * The list of courses displayed should be sorted alphanumerically by course code and shown grouped in descending order by term. * The list of TAs should be sorted alphabetically by their last names. * The list of tasks should be sorted alphabetically by the task name. * 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 instruct or 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. |
| *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 |
| *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 |
| *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 from the list.   --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 |

2.4.2 Object Model

2.4.3 Dynamic Model

1. Glossary  
     
   **\*THINGS TO LOOK AT:**\*  
   - Should ViewCourseList & ViewTaskList be able to be initiated by TA, if according to high level diagram, TA only accesses BrowseOwnTasks? (which only has ViewTask & ViewTaskEvaluation as sub use cases)  
   - IS UC-13 a duplicate of UC-14?

* Our 2 high level use cases differ in the flow of events; BrowseOwnTasks assumes login completed, ManageTasks doesn’t
* Does the inclusion of other use cases belong in the Quality Requirements section of BrowseOwnTasks?
* Entry condition for UC-03; should we reference what is returned from BrowseOwnTasks? Isn’t that getting too much into menu navigation? Purely as a use case, we just need A list, not necessarily to say where the list comes from?