**Object Oriented Languages and Systems**

**Review Versioning in Expertiza**

E723

Design Document

****

Arjun Asok Nair (aasokna)

Davis Thomas (dthomas)

Manu Joseph (mjoseph)

Sharan Chandran (schand10)

Contents

[I. Project Description 3](#_Toc340242810)

[II. Design Overview 3](#_Toc340242811)

[III. Use Cases 5](#_Toc340242812)

[IV. Object Oriented Design 11](#_Toc340242813)

[V. Database Design 13](#_Toc340242814)

[VI. Test Plan 16](#_Toc340242815)

# Project Description

## Introduction

Expertiza is a peer-review based learning platform in which work-products by students are evaluated and scored by peer-review of other students. Review process occurs in stages. After submission of initial feedback for a work-product by the peer-group, students are allowed time to incorporate the comments and improve their work. This second version of the work product is reviewed again and scores are given. This process might be repeated again and the average score from the last review stage is considered as the final score for that work-product. This incremental development of work-products and progressive learning is the fundamental concept underlying the Expertiza system.

## Purpose

The review-versioning project for Expertiza enables the reviewers to get a new template for review during the each review stage. The work-product is evaluated again in all required aspects and new scores are given. The reviewers can also see his older reviews to check the comments he had given in earlier stages and test whether those have been accommodated. The review saved at each deadline stage would be considered as a new version of the review by Expertiza. The reviewers may edit the current version of his review anytime during the current review stage. Only the scores obtained for version of the review performed in the latest review stage are considered to calculate the final score. Older review versions are considered stale. This makes sure that the latest work product submitted by students is the one that will account for the final score for the assignment.

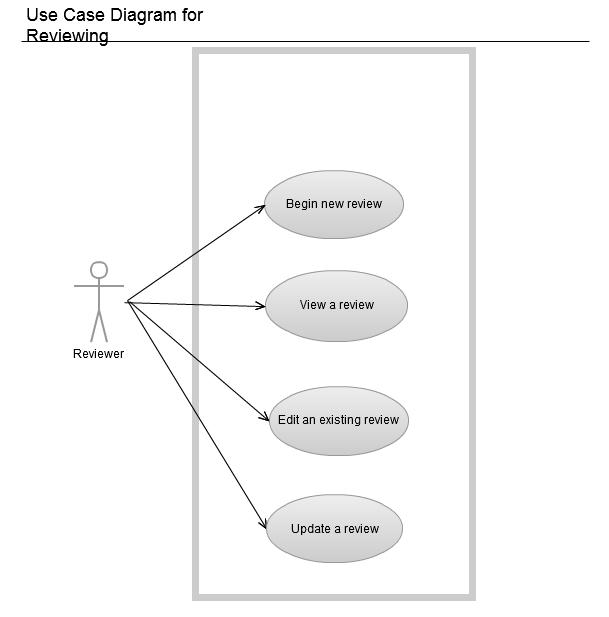
Another feature to be introduced to Expertiza as part of this project is the ability to delete work-products (i.e. links, files etc) submitted by authors. This will enable authors to change and update their work any number of times during the submission stage.

# Design Overview

Changes required to Expertiza for incorporating review versioning are as follows

1. To support multiple review versions
2. Add a new column called ‘version\_num’ to response table and model (already done by previous team that worked on the same project)
3. For a reviewer check if he has performed a review within the current review period. If yes give him an option to edit that review.
4. If the reviewer has not performed a review within the current review period then create a new review object and link it to the same review\_object\_id (assignment) but give it the next version\_num
5. Give options to view all versions of the review for the author and the reviewer, sorted in the descending order of version\_num
6. Algorithm to determine whether a review is stale for an assignment
7. Fetch all the reviews for the assignment and find the review with latest version\_num. Let it be latest\_review
8. Fetch all review deadlines for the assignment and sort by version\_num
9. If the timestamp latest\_review falls outside of the review period of the latest review the mark the review as stale
10. Calculation of total score for an assignment
11. While calculating average score for an assignment do not include the scores of the reviews that are marked stale
12. **Design Patterns used**
13. MVC pattern – We follow MVC pattern by adding creation/manipulation of review versions to the model classes (response.rb and response\_map.rb) and the decision logic (whether new review is to be created or existing review to be modified) to the controller class (response\_controller.rb). Also, the UI changes would be done to appropriate view files.
14. Strategy pattern – Two different strategies are required depending on whether a review is stale or not. If the review is not stale, it’s score has to be included in the average score for the assignment and if it is stale it should not be included. This decision would be taken by the review object in a polymorphic manner (based on above algorithm) and this will be incorporated in the final score.
15. Iterator pattern – to traverse different versions of a review at different places. Eg. to find the latest version of a review etc

# Use Cases



Assignment Review

|  |  |
| --- | --- |
| **Actor** | **Description** |
| Reviewer | The reviewer is a student/instructor who has the permission to and wants to review an assignment that he/she selects or has been assigned to review. |

**Name:** Begin a new review

**Actor:** Reviewer

**Other Participants:** None

**Preconditions:**

1. The reviewer should be logged in to Expertiza and should have an assignment to review.
2. No previous reviews were performed by this reviewer on a particular submission of the assignment.
3. The assignment should be in "Review" state

**Primary Sequence:**

1. Log in to Expertiza
2. Select the assignment
3. Click on other's work
4. Select/request a new submission to review
5. Click on "Begin" to start the review
6. Fill the review form
7. Save the review

**Alternate Flow:** None

**Name:** Edit an existing review

**Actor:** Reviewer

**Other Participants:** None

**Preconditions:**

1. The reviewer should be logged in to expertiza and should have an assignment to review.
2. A review of the submitted assignment should be already performed in the current review state (review/re-review).
3. The assignment should be in "Review" or "re-review" state

**Primary Sequence:**

1. Log in to Expertiza
2. Select the assignment
3. Click on other's work
4. Click on "Edit" to perform the edit operation on the latest version of the review
5. The review form should be loaded with the values that were previously saved.
6. Correct the review form to needed values
7. Save the review

**Alternate Flow:** None

**Name:** View an existing review

**Actor:** Reviewer

**Other Participants:** None

**Preconditions:**

1. The reviewer should be logged in to Expertiza and should have an assignment to review.
2. A review of the submitted assignment should be already performed.
3. The assignment should be in "Review" or "Re-review" state.

**Primary Sequence:**

1. Log in to Expertiza
2. Select the assignment
3. Click on other's work
4. Click on "View" to view the latest version of performed review while the assignment.
5. The review should be loaded with the values that were previously saved without the option to edit.
6. Click back to go back to the previous page.

**Alternate Flow:**

1. Log in to Expertiza
2. Select the assignment
3. Click on other's work
4. Click on the link to the previous version to view the performed review while the assignment has crossed the "Review" state and if there is a previous version.
5. The review should be loaded with the values that were previously saved without the option to edit.
6. Click back to go back to the previous page.

**Name:** Update a Review

**Actor:** Reviewer

**Other Participants:** None

**Preconditions:**

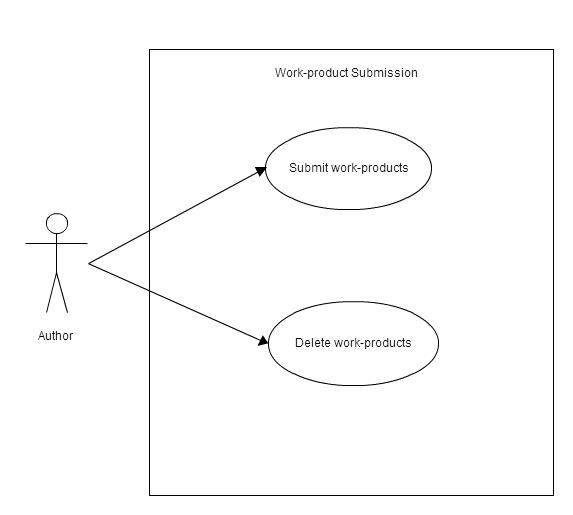
1. The reviewer should be logged in to expertiza and should have an assignment to review.
2. A review of the submitted assignment should be already performed during "Review" state.
3. The assignment should be in "Re-Review" state. No previous reviews were performed in the current state.

**Primary Sequence:**

1. Log in to Expertiza
2. Select the assignment
3. Click on other's work
4. Click on "Update" to perform a re-review.
5. The review form should be loaded with default values (values as in a new review)
6. Save the review.

**Alternate Flow:** None

**Use Cases for Deletion of work-products**

****

**Name:** Deletion of work-products

**Actor:** Author

**Other Participants:** Administrator / Instructor

**Precondition:** The assignment should be in “Submission” or “Re-submission” phase while the author tries to delete work-products.

**Primary Sequence:**

1. Go to “Your Work” page on Expertiza for a particular assignment. The work-products for this assignment are listed here.
2. Select the work-products (hyperlinks, files etc.) which the user wants to delete by clicking the check-boxes next to them.
3. Click on delete button.
4. The selected work-products will be deleted and the user will be re-directed to “Your Work” page.

**Alternate Sequence:** None

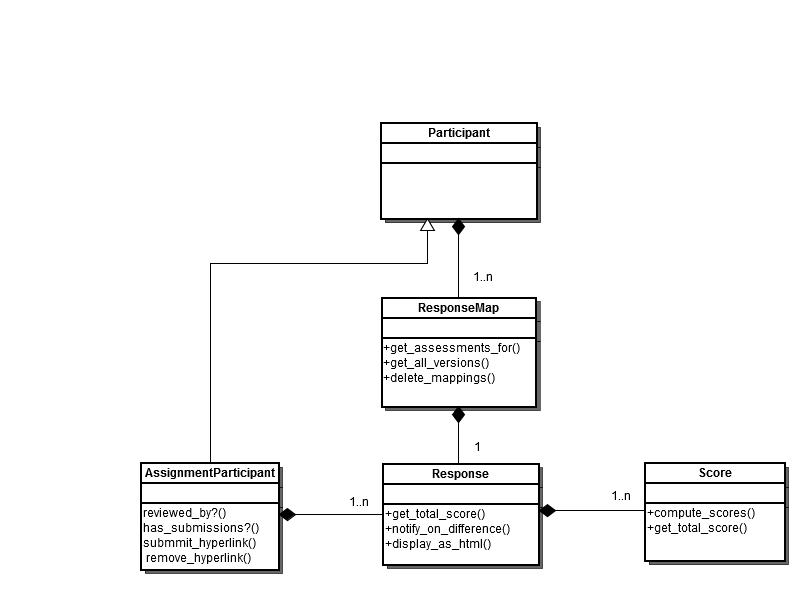
# Object Oriented Design

Following are some of the files which will contain major part of implementation changes:

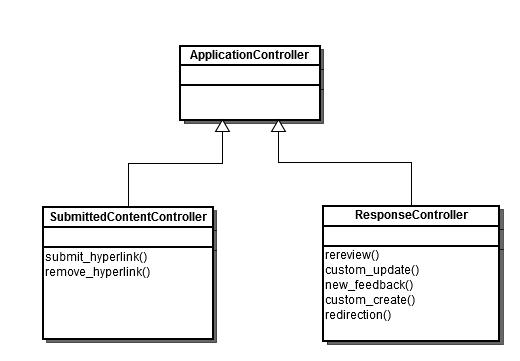
1. app/models/participant.rb
2. app/models/response\_map.rb
3. app/models/assignment\_participant.rb
4. app/models/response.rb
5. app/models/score.rb
6. app/controllers/submitted\_content\_controller.rb
7. app/controllers/response\_controller.rb

As with many web applications, Model-View-Controller architecture is used to realize this solution.

1. Class diagram showing the relationships among the various Model classes:



1. Class diagram showing the relevant Controller classes:



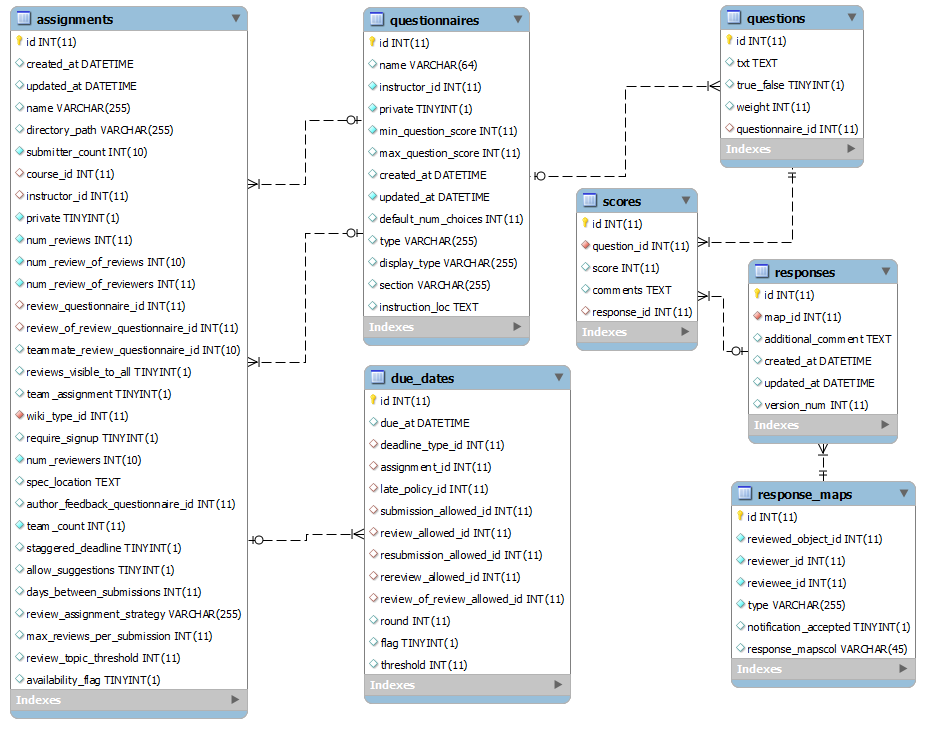
# Database Design

The review feature of Expertiza will be used by reviewers to review an assignment, by assignment participants to view their scores and by instructors to assign a review questionnaire to an assignment. This feature is also used for performing meta-reviews. The system maintains information about all the reviews for each assignment. This information is then used to calculate the score of an assignment. We describe here only a part of the system, i.e. tables and their corresponding fields, that is used to implement the review functionality of Expertiza.

Classes of Users

1. **Reviewer** is responsible for reviewing an assignment and assigning scores for each question of a review questionnaire.
2. **Instructor**is responsible for assigning a review questionnaire to an assignment.
3. **Assignment Participant**can view the scores assigned by the reviewer for an assignment that a user is participating in.

E-R Diagram



Major entities in the system

The major entities in the database and their respective fields that are considered for the for review feature are:-

1. **Assignments**- represents an assignment that a user participates in and is reviewed by a reviewer. The system stores the following:-

|  |  |  |
| --- | --- | --- |
| *Field* | *Type* | *Description* |
| id | INT(11) | Unique id assigned to an assignment |
| review\_questionnaire\_id | INT(11) | Id of the questionnaire assigned for the reviewof an assignment. |
| review\_of\_review\_questionnaire\_id | INT(11) | Id of the questionnaire assigned for the review of review of an assignment. |

1. **Due dates** - represents all due dates of an assignment. An assignment can have multiple due dates of different types such as assignment due date, review due date etc. The system stores the following information :-

|  |  |  |
| --- | --- | --- |
| *Field* | *Type* | *Description* |
| id | INT(11) | Unique id assigned to a due date |
| due\_at | DATETIME | Time of the deadline for an assignment. |
| deadline\_type\_id | INT(11) | Id of the type of deadline. There are multiple deadlines for an assignment, such as submission deadline, review deadline etc. |
| assignment\_id | INT(11) | Id of the assignment for which this deadline applies. |

1. **Questionnaires** - represents a set of questions assigned to the review of an assignment. The system stores the following information :-

|  |  |  |
| --- | --- | --- |
| *Field* | *Type* | *Description* |
| id | INT(11) | Unique id assigned to a questionnaire. |

1. **Questions**- represents a question that can be part of any questionnaire. System stores the following information :-

|  |  |  |
| --- | --- | --- |
| *Field* | *Type* | *Description* |
| id | INT(11) | Unique id assigned to a question. |
| weight | INT(11) | The weight assigned to a question. |
| questionnaire\_id | INT(11) | Id of the type of questionnaire to which this question belongs. |

1. **Scores** - represents the scores assigned for a question in the response by a reviewer. The system stores the following information:-

|  |  |  |
| --- | --- | --- |
| *Field* | *Type* | *Description* |
| id | INT(11) | Unique id assigned to a score. |
| question\_id | INT(11) | The id of the question for the score is given. |
| score | INT(11) | The value of the score given. |
| comments | TEXT | The comments given for the question. |
| response\_id | INT(11) | The id of the response of a reviewer. |

1. **Responses**– represents the response to a review of a questionnaire of an assignment. The system stores the following information :-

|  |  |  |
| --- | --- | --- |
| *Field* | *Type* | *Description* |
| id | INT(11) | Unique id assigned to a response of a reviewer. |
| map\_id | INT(11) | The id to the mapper table that maps the response to the reviewer, reviewee and assignment. |
| additional\_comment | TEXT | Any additional comments given in the response of the questionnaire. |
| created\_at | DATETIME | The create time of the response. |
| updated\_at | DATETIME | Theupdate timeof the response. |
| version\_num | INT(11) | The id of previous version of the response if a rereview was submitted. |

1. **Response Maps**– represents the mapping of response to its reviewer, reviewee and the assignment. The system stores the following information :-

|  |  |  |
| --- | --- | --- |
| *Field* | *Type* | *Description* |
| id | INT(11) | Unique id assigned to a response of a reviewer. |
| reviewed\_object\_id | INT(11) | The id of the assignment for the response is given. |
| reviewer\_id | INT(11) | The id of the reviewerwho gave the response. |
| reviewee\_id | INT(11) | The id of the reviewee. |
| type | DATETIME | Type of review, such as, Metareview, rereview etc. |

# Test Plan

**Test - Add a new review**

1. The user should be allowed to review only within the review date. So review date should be validated.
2. The link "Begin" should be available for a new review.
3. A new review should get added to the database when it is saved.

**Test - Edit an existing review**

1. The user should be allowed to edit only within the review date for the current version.
2. A link "Edit" should be visible for the user.
3. When "Edit" form is displayed, it should populate the fields with the existing values from the database.
4. The changes saved should be updated to the database.

**Test - View an existing review**

1. Viewing a review should be allowed only after a review is performed.
2. The link "View" should be visible for the review in the current review period.
3. All previous versions of reviews which passed review dates should be displayed with links as the review names.

**Test - Update an existing review**

1. "Update" should be visible only if a review was previously performed.
2. The assignment state should be in re-review.
3. Review form should be populated with default values like a new review
4. The updated review should get added to the database.

**Test** - **Validate the reviews fetched for calculating score are not stale**

1. Fetch all non-stale reviews for the assignment, that has both reviews that are not redone and also that are redone.
2. Validated that the review date of all fetched reviews does not fall outside of the review period of the latest review.

**Test** - **Validate score calculation**

1. Do re-review an existing review for an assignment.
2. Validate that the score of the assignment got updated correctly, by including the new re-review into its score calculation.

**Test - Validate work-product deletion**

1. Select the work-product(s) to be removed and click ‘Delete’.
2. Verify that the user is re-directed to the “Your Work” page and the work-products marked for deletion are not displayed on the page.

**Test - Validate deletion of work-products from database**

1. Select the work-product(s) to be removed and click ‘Delete’.
2. Verify that the entries corresponding to the work-products marked for deletion are removed from the database.