Review Management Requirements Specification

1. Scope

1.1 Overview

Reviews are produced based on scorecards. A review holds a collection of items which address each of the questions on the scorecard. It also consists of the author that produced the review, the submission it addresses and the scorecard template it is based on. Various types of comments can be attached to the review or to each review item. A committed review must address all questions on the corresponding scorecard, and will have its overall score available.

The component provides the management functionalities to create, update or search reviews. The review persistence logic is pluggable.

1.2 Logic Requirements

1.2.1 Review Management Operations

Despite manipulation of the review structure defined above, this component will support the following operations.

1.2.1.1 Create Review

A new review can be created. The identifiers will be provided with ID Generator. Review should be validated.

The operator needs to be provided for auditing purpose.

1.2.1.2 Update Review

An existing review can be updated. The identifiers of any new entities will be provided with ID Generator. Review should be validated.

The operator needs to be provided for auditing purpose.

1.2.1.3 Add Review Comment

A review comment can be added either to the review level or item level.

The operator needs to be provided for auditing purpose.

1.2.1.4 Get Review

A review can be retrieved by specified identifier.

1.2.1.5 Search Reviews

An array of reviews can be retrieved by specified search criteria. An option should be able to specify whether to retrieve the complete review hierarchies or only the review instances.

1.2.1.5.1 Review Search Criteria

At minimum, the criteria should be capable of specifying any combination of scorecard type, submission, reviewer, project and committed flag.

1.2.1.6 All Comment Types

There should be a way to retrieve all comment types in the system.

1.2.2 Search Builder Usage

Search Builder should be used with the searching functionality. Only the identifiers of the entities should return from the Search Builder. Convenient methods should be provided to create the applicable filters.

1.2.3 Auditing Fields

Review must also include auditing fields of creation/modification operator and timestamp. These fields will not be provided by component users.

1.2.4 Persistence

Persistence needs to be pluggable. For this release an Informix plug-in will be developed. The SQL scripts will be provided.

1.2.4.1 Persistence Implementation

The persistence implementation needs to be designed in this component, but will be separated into a second development project. Please put all persistence implementation related information into a separate sub-package and clearly mark the responsibilities of the two development projects.

1.3 Required Algorithms

No specific algorithms are required.

1.4 Example of the Software Usage

A scorecard/review application can use the component as a model layer. Application user can create and modify reviews on the web interface.

1.5 Future Component Direction

Separate component will be developed to calculate review scores. XML persistence can be developed to facilitate offline review.

2. Interface Requirements

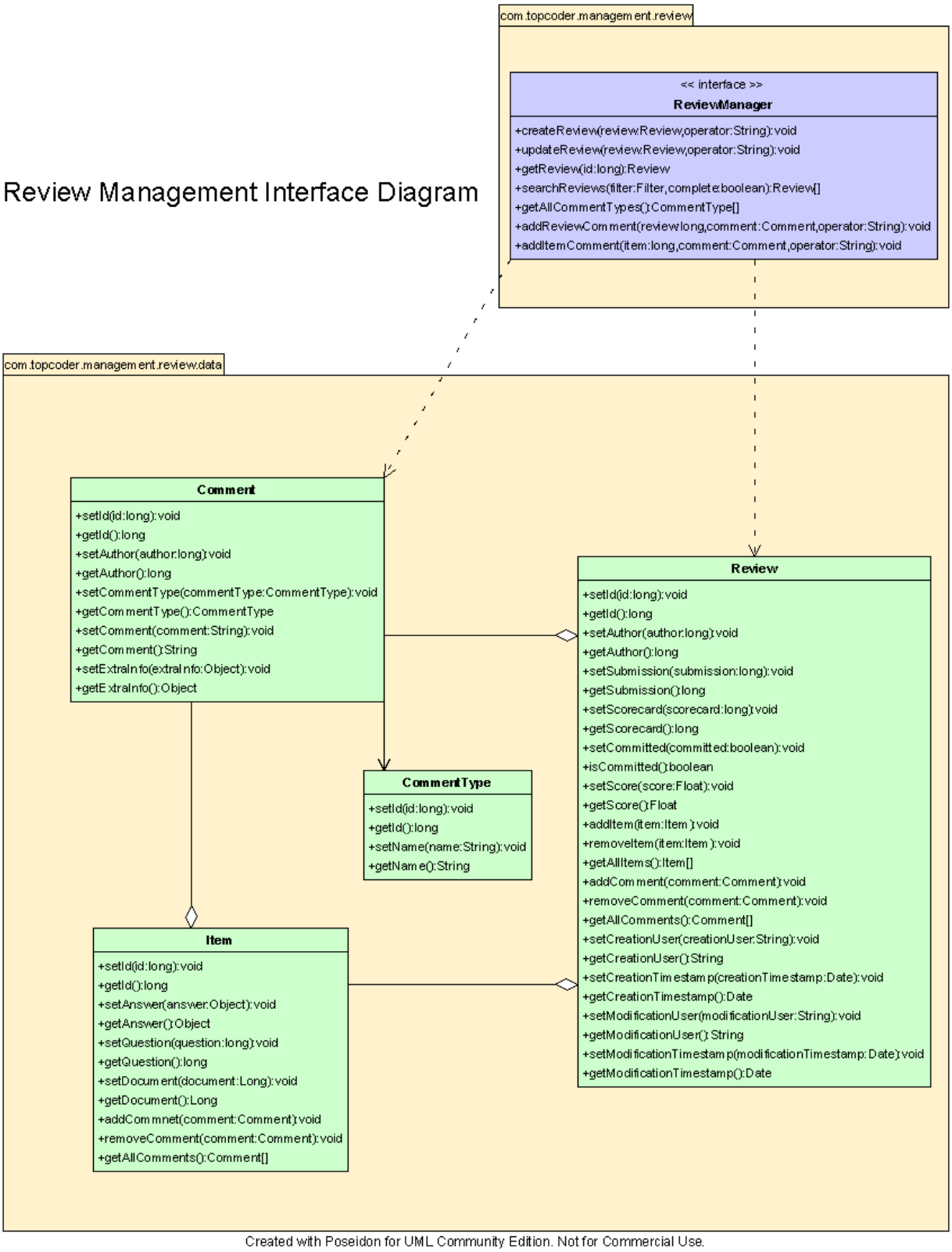
2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Design must adhere to the interface diagram definition. Designer can choose to add more methods to the classes/interfaces, but must keep the ones defined on the diagram as a minimum. Source files can be found in the distribution.

Review Management Interface Diagram



2.1.3 Environment Requirements

- Development language: Java1.4

- Compile target: Java1.4

2.1.4 Package Structure

com.topcoder.management.review

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Database connection

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

JDBC

3.2.2 TopCoder Software Component Dependencies:

- Review
- Configuration Manager
- DB Connection Factory
- ID Generator
- Search Builder

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None.

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003
- Informix 10.0

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

3.3.1 Database Connections

Database connections must not be cached within the component. Connections should be created for each operation and closed afterwards.

3.3.2 Component Scalability

The component needs to be scalable. Running multiple instances in the same JVM or in multiple JVM's concurrently should not cause any problem.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of Poseidon.