Online Review Deliverables 1.0 Component Specification

1. Design

The Online Review application defines a set of deliverables. The Deliverable Management component defines an object model and handling capabilities for deliverables. This component provides plug-ins to the Deliverable Management component, whose logic is to verify the completion of various deliverables and load completion timestamps for those that are completed.

The design of this component is based on querying the SQL database that stores the deliverables and related information. Because SQL already contains the ability to join tables and select the maximum date among a set of maximum rows, most of the checks needed by this component can be done as a single SQL query, with no code needed other than that to execute the SQL query through JDBC. Although this approach does not provide any support for pluggable persistence stores, this approach is justified based on its simplicity, and the fact that since this is a custom component, it is incredibly unlikely that any other persistence store support will ever be needed.

1.1 Design Patterns

The SingleQuerySqlDeliverableChecker uses the **Template** pattern to allow subclasses to control the exact SQL query used and the parameters needed to execute the statement. The DeliverableChecker interface and implementations are used in a **Strategy** pattern, but this pattern is only really existent when this component is combined with the Deliverable Management component.

1.2 Industry Standards

SQL

1.3 Required Algorithms

1.3.1 Single Query SQL Statements

Most of the checks required by this component can be done by making a single SQL statement that selects all modification dates meeting the desired restrictions and then using the MAX function to select only the largest one. This component then has only to read this single value from the ResultSet and set it as the completion date of the deliverable. If no entries in the database match, the ResultSet will have no rows. This section lists the SQL statements needed.

1.3.1.1 Submission Checker SQL Statement

```
SELECT MAX(modify_date)
FROM upload
INNER JOIN upload_type_lu
      ON upload.upload_type_id = upload_type_lu.upload_type_id
INNER JOIN upload_status_lu
```

```
      ON upload.upload_status_id = upload_status.upload_status_id
WHERE upload_type_lu.name = 'Submission'
      AND upload_status_lu.name = ‘Active’
      AND upload.project_id = ?
      AND upload.resource_id = ?
```

1.3.1.2 Screening / Review / Approval Checker

```
SELECT MAX(modify_date)
FROM review
WHERE committed = TRUE
      AND resource_id = ?
      AND submission_id = ?
```

Note that a where clause for the project id is not needed, since a submission is already associated with exactly 1 project. Also note that this same SQL statement satisfies all of 1.2.2, 1.2.3, and 1.2.11.

1.3.1.3 Test Cases Checker

```
SELECT MAX(modify_date)
FROM upload
INNER JOIN upload_type_lu
      ON upload.upload_type_id = upload_type_lu.upload_type_id
INNER JOIN upload_status_lu
      ON upload.upload_status_id = upload_status.upload_status_id
WHERE upload_type_lu.name = 'Test Case'
      AND upload_status_lu.name = ‘Active’
      AND upload.project_id = ?
      AND upload.resource_id = ?
```

Note that the resource encapsulates the idea of which type of test (Stress/Accuracy/Failure) is uploaded

1.3.1.4 Aggregation Checker

```
SELECT MAX(modify_date)
FROM review
INNER JOIN resource
      ON review.resource_id = resource.resource_id
WHERE committed = TRUE
      AND review.resource_id = ?
      AND resource.project_id = ?
```

1.3.1.5 Aggregation Review Checker

```
SELECT MAX(review_comment.modify_date)
FROM review_comment
INNER JOIN comment_type_lu
      ON review_comment.comment_type_id =
            comment_type_lu.comment_type_id
INNER JOIN review
      ON review.review_id = review_comment.review_id
INNER JOIN resource
      ON resource.resource_id = review_comment.resource_id
WHERE review_comment.resource_id = ?
      AND resource.resource_id = ?
      AND comment_type_lu.name = 'Aggregation Review Comment'
```

```
      AND (review_comment.extra_info = 'Approved'
            OR review_comment.extra_info = 'Rejected')
```

1.3.1.6 Final Fixes Checker

```
SELECT MAX(modify_date)
FROM upload
INNER JOIN upload_type_lu
      ON upload.upload_type_id = upload_type_lu.upload_type_id
INNER JOIN upload_status_lu
      ON upload.upload_status_id = upload_status.upload_status_id
WHERE upload_type_lu.name = 'Final Fix'
      AND upload_status_lu.name = ‘Active’
      AND upload.project_id = ?
      AND upload.resource_id = ?
```

1.3.1.7 Submitter Comment Checker

```
SELECT MAX(review_comment.modify_date)
FROM review_comment
INNER JOIN comment_type_lu
      ON review_comment.comment_type_id =
            comment_type_lu.comment_type_id
INNER JOIN review
      ON review.review_id = review_comment.review_id
WHERE review.submission_id = ?
      AND review_comment.resource_id = ?
      AND comment_type_lu.name = 'Submitter Comment'
      AND (review_comment.extra_info = 'Approved'
            OR review_comment.extra_info = 'Rejected')
```

1.3.1.8 Final Review Checker

```
SELECT MAX(review_comment.modify_date)
FROM review_comment
INNER JOIN comment_type_lu
      ON review_comment.comment_type_id =
            comment_type_lu.comment_type_id
INNER JOIN review
      ON review.review_id = review_comment.review_id
WHERE review.submission = ?
      AND review_comment.resource = ?
      AND comment_type_lu.name = 'Final Review Comment'
      AND (review_comment.extra_info = 'Approved'
            OR review_comment.extra_info = 'Rejected')
```

1.3.2 Appeal Responses Checker

For the check of Appeal Responses, the technique used for all the other checks can not be used, because for each “Appeal” comment, there might or might not be an “Appeal Response” comment. This means that doing the correct join of tables followed by using a MAX function will not work, because some entries in the table may be NULL, and the MAX function ignores NULL (it returns the max over all non-null values). A single SQL statement can still be used to select the correct entries, but manual iteration of the selected dates will be needed.

The following SQL statement will select each “Appeal” comment, and then do an outer join looking for the corresponding “Appeal Response”, selecting the modification date of the “Appeal Response” comment, if it exists, and NULL otherwise.

```
SELECT appeal_response_comment.modify_date
FROM review_item_comment appeal_comment
INNER JOIN comment_type_lu appeal_comment_type
      ON appeal_comment.comment_type_id =
            appeal_comment_type.comment_type_id
INNER JOIN review_item
      ON appeal_comment.review_item_id = review_item.review_item_id
INNER JOIN review
      ON review_item.review_id = review.review_id
LEFT OUTER JOIN (review_item_comment appeal_response_comment
      ON appeal_response_comment.review_item_id =
      appeal_comment.review_item_id
      INNER JOIN comment_type_lu appeal_response_comment_type
      ON appeal_response_comment.comment_type_id =
            appeal_response_comment_type.comment_type_id)
WHERE review.resource_id = ?
      AND appeal_comment_type.name = 'Appeal'
      AND appeal_response_comment_type.name = 'Appeal Response'
```

Once this query is executed, it will be necessary to iterate over the returned dates in this column. If any entry is NULL, the deliverable is not complete. If no entry is NULL, the deliverable is complete, and the latest date should be used as the completion date of the deliverable.

1.4 Component Class Overview

SqlDeliverableChecker:

The SqlDeliverableChecker class is the base class for any DeliverableChecker whose checking process involves querying a database to determine the completion date. Although it implements the DeliverableChecker interface, the actual checking method and logic must be provided by each subclass. This class simply holds the DBConnectionFactory and connection name used, and provides subclasses a Connection to the database when requested. The only complicated thing to implement in this class is opening the database connection; this is simply delegated to the DB Connection Factory component, so it is not complicated either.

This class is immutable.

SingleQuerySqlDeliverableChecker:

The SingleQuerySqlDeliverableChecker class templates the process of checking a deliverable whose completion date can be found by executing a single SQL query which returns a single row containing a single column of the Date type. (Or Timestamp or Time type, the conversion is transparently handled by the Database Abstraction component.) This class handles most of the process of

running the query: opening the database connection, executing the query, extracting the results from the query, and setting the completion date on the deliverable, and closing the database connection. Abstract methods are provided to allow subclasses to template the SQL query to execute to set any needed parameters or the query.

This class is immutable.

SubmissionDeliverableChecker:

The SubmissionDeliverableChecker class subclasses the SingleQuerySqlDeliverableChecker class. The SQL query that it executes checks whether a given resource (in this case a designer/developer) has uploaded a submission for the project. If so, it marks the deliverable complete, using the date of the last upload. This class, like all of the SingleQuerySqlDeliverableChecker classes is simple to implement, as all it has to do is provide the correct SQL query and set a couple parameters of the PreparedStatement, using the properties of the Deliverable.

This class is immutable.

CommittedReviewDeliverableChecker:

The CommittedReviewDeliverableChecker class subclasses the SingleQuerySqlDeliverableChecker class. The SQL query that it executes checks whether a given resource (in this case a reviewer) has completed a review for the project and submission. If so, it marks the deliverable as complete, using the date the review was last changed. Note that this class satisfies requirements 1.2.2, 1.2.3, and 1.2.11, as the SQL query that needs to be executed in all of these instances is identical. This class, like all of the SingleQuerySqlDeliverableChecker classes is simple to implement, as all it has to do is provide the correct SQL query and set a couple parameters of the PreparedStatement, using the properties of the Deliverable.

This class is immutable.

TestCasesDeliverableChecker:

The TestCasesDeliverableChecker class subclasses the SingleQuerySqlDeliverableChecker class. The SQL query that it executes checks whether a given resource (in this case a reviewer) has uploaded a test suite. If so, it marks the deliverable as completed, using the date of the last test suite upload. This class, like all of the SingleQuerySqlDeliverableChecker classes is simple to implement, as all it has to do is provide the correct SQL query and set a couple parameters of the PreparedStatement, using the properties of the Deliverable.

This class is immutable.

AggregationDeliverableChecker:

The AggregationDeliverableChecker class subclasses the SingleQuerySqlDeliverableChecker class. The SQL query that it executes checks whether a given resource (in this case a reviewer) has committed a review. (Note that, unlike the CommittedReviewDeliverableChecker class, the committed review is per project and not per submission). If there is a committed aggregation review,

it marks the deliverable as completed, using the date the review was last modified. This class, like all of the SingleQuerySqlDeliverableChecker classes is simple to implement, as all it has to do is provide the correct SQL query and set a couple parameters of the PreparedStatement, using the properties of the Deliverable.

This class is immutable.

AggregationReviewDeliverableChecker:

The AggregationReviewDeliverableChecker class subclasses the SingleQuerySqlDeliverableChecker class. The SQL query that it executes checks whether a given resource (in this case an aggregation reviewer) has attached an approved or rejected comment to an aggregation review. If so, it marks the deliverable as completed, using the last update time of the comment. This class, like all of the SingleQuerySqlDeliverableChecker classes is simple to implement, as all it has to do is provide the correct SQL query and set a couple parameters of the PreparedStatement, using the properties of the Deliverable.

This class is immutable.

FinalFixesDeliverableChecker:

The FinalFixesDeliverableChecker class subclasses the SingleQuerySqlDeliverableChecker class. The SQL query that it executes checks whether a submitter has uploaded the final fixes for the component. If so, it marks the deliverable as completed, using the date of the last final fixes upload. This class, like all of the SingleQuerySqlDeliverableChecker classes is simple to implement, as all it has to do is provide the correct SQL query and set a couple parameters of the PreparedStatement, using the properties of the Deliverable.

This class is immutable.

FinalReviewDeliverableChecker:

The FinalReviewDeliverableChecker class subclasses the SingleQuerySqlDeliverableChecker class. The SQL query used checks to see whether the final review for the submission is complete. If so, it marks the deliverable as completed, using the date of the approval/rejection comment. This class, like all of the SingleQuerySqlDeliverableChecker classes is simple to implement, as all it has to do is provide the correct SQL query and set a couple parameters of the PreparedStatement, using the properties of the Deliverable.

This class is immutable.

SubmitterCommentDeliverableChecker:

The SubmitterCommentDeliverableChecker class subclasses the SingleQuerySqlDeliverableChecker class. The SQL query that it executes checks whether the resource (i.e. the submitter) has attached an approved or rejected comment to the review. If so, it marks the deliverable as completed, using the date that the comment was updated. This class, like all of the SingleQuerySqlDeliverableChecker classes is simple to implement, as all it has to

do is provide the correct SQL query and set a couple parameters of the PreparedStatement, using the properties of the Deliverable.

This class is immutable.

AppealResponsesDeliverableChecker:

The AppealResponsesDeliverableChecker class subclasses the SqlDeliverableChecker class and checks that every "Appeal" comment has been responded to with an "Appeal Response". If this is the case, the deliverable is marked as complete. Unlike the other classes in the component, a simple SQL query with a MAX function will not suffice, and manual iteration of the results will be needed. This makes this class somewhat (but not significantly) harder to develop than the SingleQuerySqlDeliverableChecker class. All the complication is in the check method (see its documentation for more detail).

This class is immutable.

1.5 Component Exception Definitions

No custom exceptions are declared in this component, as all classes must adhere to an interface definition in another component. Hence, the only exception thrown is the DeliverableCheckingException declared in that component. (The exact exception used will depend on the winning Deliverable Management submission. The use of the DeliverableCheckingException is a "best guess" at what this exception will be, but is subject to change during final fixes.)

1.6 Thread Safety

Although thread-safety is not required for this component, this design is thread-safe. This is due to the inherent nature of a DeliverableChecker as a stateless class. The only state needed in the implementations is the database connection factory. Since the connection factory and connection name are immutable, there is no mutable state that could cause thread safety issues. Furthermore, the information used on each check is provided solely through the method parameters, so there is no way that calls on multiple threads can even access the information used in the calls in other threads.

2. Environment Requirements

2.1 Environment

Java 1.4+ is required for compilation, testing, or use of this component.

2.2 TopCoder Software Components

- DB Connection Factory 1.0: Used for configuration of the DB Connection factory component, and can also be used with Object Factory. There is no direct use in this component.
- Database Abstraction 1.1: Allows the SQL queries that are executed to return Date or Timestamp or Time values. The Database Abstraction component

allows these values to be transparently mapped to the Date type needed to set the completion date of the deliverable.
- Configuration Manager 2.1.4: Used for configuration of the DB Connection factory component. Can also be used for Object Factory configuration.
- Object Factory 2.0: Can be used to create the various DeliverableCheckers from a configuration, if this is desired. There is no enforced compile time or runtime dependency on this component.

2.3 Third Party Components

None

3. Installation and Configuration

3.1 Package Name

com.cronos.onlinereview.deliverables

3.2 Configuration Parameters

This component does not provide any direct configuration parameters. The Object Factory component can be used to create the various deliverable checkers from a configurable specification.

3.3 Dependencies Configuration

None.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

Follow the above instructions. To use Object Factory for configuration, see the Object Factory component documentation.

4.3 Demo

None needed.

5. Future Enhancements

As this is a custom component that simply provides specific interface implementations for an interface declared in another component, no future enhancements (beyond bug fixes) are expected.