[TopCoder]

Review Selection 1.0 Component Specification

1. Design

TopCoder attempts to improve the competition review process and is in need for updating current systems for integrating with new review process outlined in separate Competition Review Process Conceptualization document.

Currently there are two applications involved in review process. TopCoder Website application is used by the reviewers for signing up for the review positions and listing the statistics for projects and users. Online Review application is actually used for managing the review process providing the users with abilities to upload submissions for contests, fill the screening and review scorecards, submitting and resolving appeals, performing contest results aggregation, submitting final fixes and approving the results of contests.

This component will implement the algorithms and the data structures required to implement the review selection algorithm. The ReviewSelection interface is used to select primary and secondary reviewers, the current implementation for this interface is the RatingBasedSelectionAlgorithm class, which select reviewers based on their past performance statistics and current workload. The ReviewApplicationProcessor interface is used by the RatingBasedSelectionAlgorithm class to restore eligibility review points.

This component also provides a ReviewBoardApplicationListener interface, this interface is used for the cases when the review positions are not filled at the completion of the review registration phase. This interface listens for review registration events and tests if the reviewer applications are sufficient in such an event, in which case it closes the review registration phase.

1.1 Design Patterns

1.1.1 Strategy pattern

The RatingBasedSelectionAlgorithm class uses pluggable WorkloadFactorCalculator and ReviewApplicationProcessor implementations to calculate workload factor and process reviewer selection result. The WorkloadFactorCalculator interface is implemented by the DefaultWorkloadFactorCalculator class, and the ReviewApplicationProcessor interface is implemented by the ReviewApplicationProcessor class.

The ReviewSelection, ReviewApplicationProcessor and ReviewBoardApplicationListener interfaces and their implementations provided in this component can be possibly used in some external strategy context.

1.1.2 DTO pattern

DTO classes like ReviewApplication, ReviewerStatistics and Project are used by this component.

1.2 Industry Standards

None.

1.3 Required Algorithms

1.3.1 Select reviewer based on performance statistics and current workload

The RatingBasedSelectionAlgorithm class implements the ReviewSelection interface. It will select primary reviewer and secondary reviewers based on the reviewer's past performance statistics and current workload. The following steps should be performed:

1. Get the project type for given project by using ProjectManager.

```
// retrieve the project object

Project project = projectManager.getProject(applications[0].getProjectId());

// get project type
```

```
long projectTypeId = project.getProjectCategory().getProjectType().getId();
```

2. For each reviewer application, calculate the reviewer's coefficient:
    2.1. Retrieve performance statistics for the reviewer and project type using ReviewerStatisticsManager.

```
ReviewerStatistics statistics =
reviewerStatisticsManager.getReviewerStatisticsByCompetitionType(application.getRevi
ewerId(), projectTypeId);
```

    2.2. Retrieve the number of concurrent review for the reviewer using RboardApplicationBean.

```
// get the application delay for this reviewer

long delay = rboardApplicationBean.getApplicationDelay(rboardDataSourceName,
application.getReviewerId())

// calculate concurrent review count using the application delay

int activeReviewCount = delay /
RBoardApplicationBean.APPLICATION_DELAY_PER_ACTIVE_PROJECT;
```

    2.3. Calculate the workload factor from concurrent review number using WorkloadFactorCalculator.

    2.4. Calculate the coefficient for the reviewer using the following formula, note that the number 0.5 is configurable through the timelineReliabilitySubtractor field.

```
Average Coverage * Average Accuracy * (Average Timeline Reliability – 0.5) *
Workload Factor
```

3. Compute best average statistics among the primary applications to find the primary reviewer.
4. Compute the best 2 (or other configurable value) average statistics among the unselected primary reviewers to find secondary reviewers.
5. Assign reviewer to application using RBoardApplicationBean.
6. Pass the assigned applications to ReviewApplicationProcessor
7. Passed the unassigned applications to ReviewApplicationProcessor
8. Return the reviewer selection result.

Refer to the javadoc of the RatingBasedSelectionAlgorithm#selectReviewers() method for detail implementation notes.

1.3.2 Calculate workload factor from concurrent review number

The DefaultWorkloadFactorCalculator class provides default algorithm to calculate workload. The following formula should be used, note that the number 1 and 0.05 is configurable through the workloadFactorBase and reviewNumberMultiplier fields respectively.

```
1 - 0.05 * Concurrent Review Number
```

Refer to the javadoc of the DefaultWorkloadFactorCalculator#caculateWorkloadFactor() method for detail implementation notes.

1.3.3 Process reviewer selection result

The DefaultReviewApplicationProcessor class will process reviewer selection result by updating the reviewer's statistics. For reviewer applications which apply for primary reviewer and eventually not selected as primary reviewer, the primary eligibility review points will be restored. The ReviewerStatisticsManager will be used to update reviewer statistics.

Refer to the javadoc of the DefaultReviewApplicationProcessor#updateUnassignedReviewersStatistics() method for detail implementation notes.

1.3.4 *Handle review registration event*

The DefaultReviewBoardApplicationListener class implements the ReviewBoardApplicationListener interface. It is used for the cases when the review positions are not filled at the completion of the review registration phase. This listener listens for review registration events and tests if the review board is full in such an event, in which case it closes the review registration phase. Note that in order to get sufficient reviewers, there need to have at least 3 reviewers (the number is configurable using the reviewerNumber field), and at least one of them would like to be primary reviewer. The PhaseManager will be used the end the review registration phase.

Refer to the javadoc of the DefaultReviewBoardApplicationListener#applicationReceived() method for detail implementation notes.

1.3.5 *Logging*

This component will use logging wrapper component to write log. Method entrance and exit should be logged using DEBUG level, and all errors should be logged using ERROR level. Refer to the method's javadoc for detail.

1.4 Component Class Overview

ReviewSelection [interface]

This interface defines the contract for different review selection algorithms. It provides method to choose one primary reviewer and several secondary reviewers from a list of review applications. This component provides the RatingBasedSelectionAlgorithm class as an implementation of this interface.

Implementation of this interface should be thread-safe.

ReviewSelectionResult

This class is used to represent the reviewer selection result returned by ReviewSelection, it contains a primary reviewer and a list of secondary reviewers.

This class is immutable and thread-safe.

RatingBasedSelectionAlgorithm

This class implements the ReviewSelection interface. It will select primary reviewer and secondary reviewers based on the reviewer's past performance statistics and current workload. This class uses the ReviewApplicationProcessor interface to update reviewer statistics and the WorkloadFactorCalculator interface to calculator workload factor.

This class is immutable and thread-safe as long as the configure() method will be called just once right after instantiation.

ReviewApplicationProcessor [interface]

This interface is used by RatingBasedSelectionAlgorithm class to process the reviewer selection result. This component provides the DefaultReviewApplicationProcessor as an implementation of this interface.

Implementation of this interface should be thread-safe.

DefaultReviewApplicationProcessor

This class implements the ReviewApplicationProcessor interface. It will restore eligibility points for unassigned primary reviewer applications.

This class is immutable and thread-safe as long as the configure() method will be called just once right after instantiation.

ReviewBoardApplicationListener [interface]

This interface is used for the cases when the review positions are not filled at the completion of the review registration phase. This listener listens for review registration events and tests if the reviewer applications are sufficient in such an event, in which case it closes the review registration phase. This component provides the DefaultReviewBoardApplicationListener as an implementation of this interface.

Implementation of this interface should be thread-safe.

DefaultReviewBoardApplicationListener

This class implements the ReviewBoardApplicationListener interface. It will check to see whether the reviewer applications are sufficient, if such case happens, it will use the PhaseManager to close the review registration phase.

This class is immutable and thread-safe as long as the configure() method will be called just once right after instantiation.

ReviewerCoefficient

This class implements the Comparable interface, so that it can be sorted based on coefficient value and application date. It has the coefficientValue and reviewApplication fields. The coefficientValue represents the result calculated based on the reviewer's past performance statistics and current workload. Once the best coefficientValue is found, we can retrieve the corresponding reviewApplication.

This class is immutable and thread-safe.

WorkloadFactorCalculator [interface]

This interface is used by DefaultReviewApplicationProcessor to calculate workload factor based on the concurrent number of review. This component provides the DefaultWorkloadFactorCalculator as an implementation of this interface.

Implementation of this interface should be thread-safe.

DefaultWorkloadFactorCalculator

This class implements the WorkloadFactorCalculator interface. It uses the formula "workloadFactorBase - reviewNumberMultiplier * concurrentReviewNumber" to calculate workload factor.

This class is immutable and thread-safe as long as the configure() method will be called just once right after instantiation.

1.5 Component Exception Definitions

ReviewSelectionException

This exception is thrown if any error occurs while selecting reviewer. The ReviewSelection and RatingBasedSelectionAlgorithm will all throw this exception.
This class is not thread-safe because the base class is not thread-safe.

ReviewSelectionConfigurationException

This exception is thrown if any error occurs while initializing instance of ReviewSelection. The ReviewSelection and RatingBasedSelectionAlgorithm will all throw this exception.
This class is not thread-safe because the base class is not thread-safe.

ReviewApplicationProcessorException

This exception is thrown if any error occurs while processing reviewer selection result. The ReviewApplicationProcessor and DefaultReviewApplicationProcessor will all throw this exception.
This class is not thread-safe because the base class is not thread-safe.

ReviewApplicationProcessorConfigurationException

This exception is thrown if any error occurs while initializing instance of ReviewApplicationProcessor. The ReviewApplicationProcessor and DefaultReviewApplicationProcessor will all throw this exception.
This class is not thread-safe because the base class is not thread-safe.

ReviewBoardApplicationListenerException

This exception is thrown if any error occurs while handling review registration event. The ReviewBoardApplicationListener and DefaultReviewBoardApplicationListener will all throw this exception.
This class is not thread-safe because the base class is not thread-safe.

ReviewBoardApplicationListenerConfigurationException

This exception is thrown if any error occurs while initializing instance of ReviewBoardApplicationListener. The ReviewBoardApplicationListener and DefaultReviewBoardApplicationListener will all throw this exception.
This class is not thread-safe because the base class is not thread-safe.

WorkloadFactorCalculatorConfigurationException

This exception is thrown if any error occurs while initializing instance of WorkloadFactorCalculator. The WorkloadFactorCalculator and DefaultWorkloadFactorCalculator will all throw this exception.
This class is not thread-safe because the base class is not thread-safe.

1.6 Thread Safety

RatingBasedSelectionAlgorithm, DefaultReviewApplicationProcessor, DefaultReviewBoardApplicationListener and DefaultWorkloadFactorCalculator are immutable and thread-safe as long as the configure() method will be called just once right after instantiation. ReviewSelectionResult and ReviewerCoefficient are immutable and thread-safe. ReviewSelectionException, ReviewSelectionConfigurationException, ReviewApplicationProcessorException, ReviewApplicationProcessorConfigurationException, ReviewBoardApplicationListenerException, ReviewBoardApplicationListenerConfigurationException and WorkloadFactorCalculatorConfigurationException are not thread-safe, and they must be used in a thread-safe manner.

2. Environment Requirements

2.1 Environment

Java 1.5
Solaris 7
RedHat Linux 7.1
Windows 2000
Windows 2003
Oracle
JBoss 4.2.2

2.2 TopCoder Software Components

“Base Exception 2.0” - provides the BaseCriticalException class which is the base class for all the custom exception classes in this component.

“Project Management 1.2” - provides the ReviewApplicationsManager and ReviewApplication classes used to retrieve review application, also provides the ProjectManager, Project, ProjectCategory and ProjectType classes which are used to retrieve project type id.

“Resource Management 1.3” - provides the ReviewerStatisticsManager and ReviewerStatistics classes used to retrieve and update reviewer statistics.

“Phase Management 1.0” - provides the PhaseManager class used to retrieve phase information and end registration phase.

“Project Phases 2.1” - provides the Project, Phase, PhaseType which are all DTO classes used by this component.

“Logging wrapper 1.2” - used to write log.

“Configuration API 1.0” - used to provide configuration for this component.

Note: some of the component versions are updated based on the “Online Review Module component diagram” in the architecture.

“RBoardApplicationBean” - this is a class from online review application, it provides methods to retrieve concurrent review number and assign reviewer to certain application. The source code for this class is attached (RBoardApplicationBean.java).

2.3 Third Party Components

None.

3. Installation and Configuration

3.1 Package Name

com.cronos.onlinereview.review.selection
com.cronos.onlinereview.review.selection.impl
com.cronos.onlinereview.review.selection.impl.workloadfactorcalculator

3.2 Configuration Parameters

The following table describes the structure passed to the configure() method.

3.2.1 RatingBasedSelectionAlgorithm

Parameter	Description	Values
reviewerStatisticsManager	Represents the reviewer statistics manager used to retrieve reviewer statistics.	***ReviewerStatisticsManager*** Required. Cannot be null.
projectManager	Represents the project manager used to retrieve project object.	***ProjectManager.*** Required. Cannot be null.
phaseManager	Represents the phase manager used to retrieve project phase.	***PhaseManager*** Required. Cannot be null.
reviewApplicationProcessor	Represents the review application processor used to process reviewer selection result.	***ReviewApplicationProcessor*** Required. Cannot be null.
rboardApplicationBean	Represents the review board application bean used to retrieve concurrent review number and assign reviewer.	***RBoardApplicationBean*** Required. Cannot be null.
workloadFactorCalculator	Represents the workload factor calculator used to calculate workload factor.	***WorkloadFactorCalculator*** Required. Cannot be null.
loggerName	Represents the name of a logger.	***String*** Optional. Cannot be empty.
timelineReliabilitySubtractor	Represents the timeline reliability subtractor used to calculate reviewer coefficient.	***Double*** Optional. Default to 0.5. Should be positive.
rboardDataSourceName	Represents the review board data source name used to invoke methods in rboardApplicationBean.	***String*** Required. Cannot be null or empty.
primaryReviewerResponsibilityId	Represents the primary reviewer responsibility id used to invoke the createNewRBoardApplication() method in rboardApplicationBean.	***Integer*** Optional. Default to 5. Should be positive.
reviewerResponsibilityId	Represents the reviewer responsibility id used to invoke the createNewRBoardApplication() method in rboardApplicationBean.	***Integer*** Optional. Default to 4. Should be positive.
reviewPhaseName	Represents the review phase name used to find the review phase object.	***String.*** Optional. Default to “review”. Cannot be null or empty.
secondaryReviewersNumber	Represents the secondary reviewers number used to select secondary reviewers.	***Integer*** Optional. Default to 2. Should be positive.

3.2.2 DefaultReviewApplicationProcessor

Parameter	Description	Values
reviewerStatisticsManager	Represents the reviewer statistics manager used to retrieve and update reviewer statistics.	***ReviewerStatisticsManager*** Required. Can not be null.
projectManager	Represents the project manager used to retrieve project object.	***ProjectManager*** Required. Can not be null.
loggerName	Represents the name of a logger.	***String*** Optional. Cannot be empty.
primaryEligibilityPoints	Represents the primary eligibility points used to restore the primary eligibility points in reviewer statistics.	***Double*** Optional. Default to 2.0 Should be positive.

3.2.3 DefaultReviewBoardApplicationListener

Parameter	Description	Values
phaseManager	Represents the phase manager used to retrieve project phase.	***PhaseManager*** Required. Can not be null.
reviewApplicationsManager	Represents the review applications manager used to retrieve review applications.	***ReviewApplicationsManager*** Required. Can not be null.
loggerName	Represents the name of a logger.	***String*** Optional. Cannot be empty.
reviewersNumber	Represents the reviewers number used to determine if the registration phase can be ended.	***Integer*** Optional. Default to 3. Should be positive.
registrationPhaseTypeName	Represents the registration phase name used to retrieve registration phase.	***String*** Optional. Default to “review registration”. Cannot be null or empty.
registrationPhaseOperator	Represents the registration phase operator used as a parameter to end registration phase.	***String*** Required. Cannot be null or empty.

3.2.4 DefaultWorkloadFactorCalculator

Parameter	Description	Values
workloadFactorBase	Represents the workload factor base used to calculate workload factor.	***Double*** Optional. Default to 1.0. Should between 0(inclusive), 1(inclusive).
reviewNumberMultiplier	Represents the review number multiplier used to calculate workload factor.	***Double*** Optional. Default to 0.05. Should between 0(inclusive), 1(inclusive).

3.3 Dependencies Configuration

Logging wrapper component needs to be properly configured.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.

- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

Please see the demo.

4.3 Demo

4.3.1 API usage

```
// create a RatingBasedSelectionAlgorithm class and set the calculator and processor to it

RatingBasedSelectionAlgorithm reviewerSelection = new RatingBasedSelectionAlgorithm();

// configure the instance

reviewerSelection.configure(config);

.. // initialize other properties of reviewerSelection

// create a list of review applications

ReviewApplication[] applications = ...

// assume the application data is as the following table

// Note that this table does not show all the data related to the application, some data is
omitted for simplicity
```

Reviewer id	Eligibility points	Accept primary	Coefficient Value
1	10	True	50
2	10	True	40
3	0	False	30
4	0	False	20

```
// select reviewer based on their performance statistics and current workload

ReviewSelelctionResult result = reviewerSelection.selectReviewers(applications);

// assume the above data is used, and default values are used for calculator, processor and
reviewerSelection, based on the coefficient values, the id for primary reviewer should be 1,
the id for secondary reviewers should be 2 and 3. The reviewer whose id is 4 should not be
selected.
```

Reviewer id	Result
1	Primary Reviewer
2	Secondary Reviewer
3	Secondary Reviewer
4	Not Selected

```
// the selectReviewers() method will also restore eligibility points for unselected primary
reviewer, so the reviewer whose id is 2 should have its eligibility points changed from 10 to
12
```

Reviewer id	Eligibility points
2	12

```
// create a DefaultReviewApplicationProcessor

DefaultReviewApplicationProcessor processor = new DefaultReviewApplicationProcessor();

// configure the instance

processor.configure(config);

// create a list of unassigned review applications

ReviewApplication[] applications = ...

// assume the application data is as the following table

// Note that this table does not show all the data related to the application, some data is
omitted for simplicity
```

Reviewer id	Eligibility points	Accept primary
5	10	True
6	8	True
7	0	False

```
// process unassigned reviewers

processor.updateUnassignedReviewersStatistics(applications);

// assume the above data is used, and default values are used for processor, the reviewers
whose id are 5 and 6 will have their eligibility points restored to 12 and 10 respectively.
As to the reviewer whose id is 7, the eligibility points will remain 0 because he/she didn't
apply for primary reviewer.
```

Reviewer id	Eligibility points
5	12
6	10

```
// create a list of assigned review applications

ReviewApplication[] applications = ...

// process unassigned reviewers

processor.updateReviewAssignmentStatistics(applications);

// nothing will happen since the current implementation of this method is blank


// create a DefaultReviewBoardApplicationListener

DefaultReviewBoardApplicationListener listener = new DefaultReviewBoardApplicationListener();

// configure the instance

listener.configure(config);

// assume review positions are not filled at the completion of the review registration phase
and the application data is as the following table

// Note that this table does not show all the data related to the application, some data is
omitted for simplicity
```

Reviewer id	Accept primary
8	False
9	False

```
// a new reviewer just register for this project

ReviewApplication application1 = new ReviewApplication();

... // initialized the properties of application1

// assume the new review application has the following data:
```

Reviewer id	Accept primary
10	False

```
// the listener will be invoked

listener.applicationReceived(application1);

// although there are 3 applications now, since none of these reviewer would like to be
primary reviewer, the registration phase should not be ended

// after that a new reviewer register for this project and he/she would accept the
responsibility of primary reviewer

ReviewApplication application2 = new ReviewApplication();

... // initialized the properties of application2

// assume the new review application has the following data:
```

Reviewer id	Accept primary
11	True

```
// the listener will be invoked again

listener.applicationReceived(application2);

// since we have 4 reviewers and there is one primary reviewer, the registration phase should
be ended by the listener
```

5. Future Enhancements

New implementation of ReviewSelection interface can be added to select reviewer using other algorithm.

New implementation of WorkloadFactorCalculator interface can be added to use other formula to calculate workload factor.