Review Assignment 1.0 Component Specification

1. Design

TopCoder is improving its review process by collecting the reviewers' performance stats and modifying the review signup process so that the best performing reviewers have higher chances to get a review position. The first part of the system upgrade that added the reviewer rating into the Online Review application has been already implemented, see the details here. The next step is to modify the review signup process: instead of the "first come, first served" approach when members assign themselves to review positions they will apply for review positions and the system will select the best applicants for reviewing that project.

This component makes use of the API defined by the Project Management, Phase Management, Resource Management and Review Application Management components to regularly monitor pending review applications for active projects and assign applicants to review positions. The projects will be examined periodically.

1.1 Design Patterns

1.1.1 Strategy

Strategy pattern is used a lot in this component. E.g. ReviewAssignmentManager provides a strategy context for ReviewAssignmentNotificationManager, ReviewAssignmentProjectManager, ReviewAssignmentAlgorithm, etc strategy interfaces and component provides EmailBasedReviewAssignmentNotificationManager, MaxSumOfRatingReviewAssignmentAlgorithm and DefaultReviewAssignmentProjectManager strategy implementations for the mentioned strategy interfaces.

1.1.2 DAO

Component uses several external DAO-s. E.g. ReviewApplicationManager, ReviewAuctionManager, ProjectManager, ResourceManager, PhaseManager, etc.

1.1.3 DTO

Component uses several external DTO-s. E.g. Resource, ReviewApplication, ReviewAuction, etc.

1.1.4 Template method

BruteForceBasedReviewAssignmentAlgorithm.assign() is a template method. It’s constructed from several steps (prepare() , measureQuality() and finalize()), which can/must be overridden by subclass. E.g. some of its steps – measureQuality() and prepare() methods are overridden in MaxSumOfRatingReviewAssignmentAlgorithm subclass.

1.2 Industry Standards

EJB, HTML, JDBC, SQL, XML

1.3 Required Algorithms

1.3.1 Logging

This component must perform logging in all business methods.

All information described below must be logged using log:Log attribute. If log attribute is null, then logging is not required to be performed.

In all business methods method entrance with input argument, method exit with return value and call duration time must be logged at DEBUG level.

All thrown exceptions and errors must be logged at ERROR level.

Non-critical issues must be logged at WARN level. Only 1 such case is indentified – see DefaultReviewAssignmentProjectManager.addReviewersToProject() method, case when phase is not found.

Here is the additional information which must be logged at INFO level:

1. Timestamp for each tracking start (in ReviewAssignmentJobRunner.run()).
2. Auction ID, project ID, total number of open review positions and total number of pending review applications for the processed open auctions with pending review applications (in ReviewAssignmentManager.execute()).
3. What applicants are assigned to what roles (in ReviewAssignmentManager.execute() after calling ReviewAssignmentAlgorithm.assign()).
4. Whether the phases have been extended (what phases, the new end dates etc) (in DefaultReviewAssignmentProjectManager.addReviewersToProject()).
5. The email address and email type (i.e. approved/rejected/PM email) for every sent email (in EmailBasedReviewAssignmentNotificationManager. notifyXXX() methods).

1.3.2 Algorithm to find the review auctions to be processed.

See ReviewAssignmentManager.execute() method implementation notes.

High-level workflow is shown below (but please note that the UML methods docs is the primary source of information):

1) Get all auction categories.

2) For every auction category:

2.1) Get the list of all open auctions for that category.

2.2) For every open auction:

2.2.1) Check if the assignment date has been already reached. If the assignment date is still in the future, skip this auction.

2.2.2) Get the list of all pending review applications for the auction. I.e. the review applications in the "Pending" status.

2.2.3) If there are no pending review applications for the auction, skip this auction.

2.2.4) Call a pluggable strategy to assign review applicants to the review application roles (see CS 1.3.3). The input to that will be the review auction and the list of pending review applications. The result is the map from review applications to review application roles.

2.2.5) For every review application that was assigned a review application role:

2.2.5.1) Update the review application status to "Approved".

2.2.5.2) Add the member to the Online Review project (see CS 1.3.4).

2.2.5.3) Send a notification email (see section 1.3.6, notifyApprovedApplicants() method).

2.2.6) For every review application that was not assigned a review application role, update the review application status to "Rejected".

2.2.7) For every distinct applicant that was not assigned to any review application role send a notification email (see CS 1.3.6, notifyRejectedApplicants() method).

2.2.8) Send a notification email to all PMs/copilots (see CS 1.3.6, notifyManagers() method).

1.3.3 Algorithm to match applications to the review application roles.

Algorithm is based on brute force, so it simply tries each possible assignment and then finds the best one (the one with the maximum sum of the ratings of the assigned reviewers).

Brute force is implemented in BruteForceBasedReviewAssignmentAlgorithm.assign() method and calculating of ratings sum is implemented in MaxSumOfRatingReviewAssignmentAlgorithm. measureQuality() method.

Ratings are extracted from database. Rating of each reviewer is calculated as average (mean) of his review feedbacks for all the projects of same category as the review auction project. If there is less than a configurable amount (4 by default) such projects found for a reviewer, configurable default rating (1.0 by default) is assumed for this reviewer.

See implementation notes in UML docs of these methods for more details.

1.3.4 *Algorithm to add a member to an Online Review project.*

See DefaultReviewAssignmentProjectManager.addReviewersToProject() method implementation notes.

When members are added to projects they are added as "resources". A resource has a role (e.g. "Primary Screener", "Accuracy Reviewer" etc.) and a number of additional properties. Resource Management component is used to add project resources. A member can have multiple resources in the same project.

A review application role can consists of multiple resource roles (e.g. "Primary Reviewer" application role consists of four resource roles: "Primary Screener", "Reviewer", "Aggregator", "Final Reviewer"). The list of resource roles for each application role is in the ReviewApplicationRole#resourceRoles list.

High-level workflow is shown below (but please note that the UML methods docs is the primary source of information):

1) For each assignment (pair of ReviewApplication and ReviewApplicationRole):

1.1) For each ReviewApplicationResourceRole in the ReviewApplicationRole#resourceRoles list:

1.2) Get the ResourceRole instance by its ID value.

1.3) Create a new Resource instance.

1.4) Set Resource#project to the ReviewAuction#projectId value.

1.5) Set Resource#resourceRole to the ResourceRole found earlier.

1.6) If the ResourceRole#phaseType is not null, find the Phase instance whose Phase#phaseType.id value equals to ResourceRole#phaseType and set Resource#phase to the Phase#id value and perform extension if needed (see CS 1.3.5). If no phase found, then skip current resource role.

1.7) Set the "External Reference ID" resource property to ReviewApplication#userId.

1.8) Set the "Handle" resource property to the member's handle. Member's handle can be found by calling UserRetrieval#retrieveUser() method.

1.9) Set the "Registration Date" resource property to the current date and time.

1.10) Set the "Payment" resource property to "0".

1.11) Set the "Payment Status" resource property to "No".

1.12) Persist the new Resource instance.

1.13) Get the forum category ID associated with the project.

1.14) If the forum category ID property is not null and not empty, call the Forums EJB to grant the member forum access.

2) If any phase has been extended, then recalculate start and end dates of project phases if extension was made and persist these changes.

1.3.5 *Algorithm to extend phases*

See DefaultReviewAssignmentProjectManager.addReviewersToProject() method implementation notes (steps “make extension if needed” and “recalculate start and end dates of project phases if extension was made”).

High-level workflow of extending a single phase is shown below (but please note that the UML methods docs is the primary source of information):

1) If phase is open and there is a phase extension rule configured for this phase type, then:

1.1) Calculate the new value for the Phase#length field (newLength variable in the UML docs). The new value should be the duration (in milliseconds) from Phase#actualStartDate to X hours past the current time, where X is the extension value configured for this phase type.

1.2) Check the current Phase#length value. If it's smaller than the calculated value, then update the Phase#length field (populate it with newLength value calculated on previous step).

1.3.6 *Algorithm for preparing and sending the notification emails.*

See EmailBasedReviewAssignmentNotificationManager.notifyXXX() methods implementation notes.

All 3 methods have similar workflow, shown below (but please note that the UML methods docs is the primary source of information):

1) Create a params:Map<String,Object>, which will contain the parameters to be passed to document generator component for constructing the email messages.

2) Populate this map with values, common for all email messages, which need to be sent.

3) For each specific email message:

3.1) Populate message specific parameters to this map.

3.2) Send email message via EmailSendingUtility (which simply constructs the message via document generation component and sends that via email engine component).

1.4 Component Class Overview

ReviewAssignmentUtility

This is the main class of the standalone command line application that performs periodical review assignment. It uses ReviewAssignmentJobRunner, and schedules its repetitive execution with use of Job Scheduling and Job Processor components. This utility reads a configuration from a file using Configuration Persistence and Configuration API components. ReviewAssignmentUtility performs the logging of errors and debug information using Logging Wrapper.

Thread Safety:

This class is immutable and thread safe. But it's not safe to execute multiple instances of ReviewAssignmentUtility command line application (configured to use the same persistence) at a time.

ReviewAssignmentJobRunner

This class is an implementation of ScheduledJobRunner that aggregates an instance of ReviewAssignmentManager and can be used for scheduling the review assignment with use of Job Scheduling and Job Processor components. This job runner doesn't allow two jobs to be executed at the same time, thus if the previous job is not yet finished, a new one is not started. This is done in order to avoid concurrency issues.

Thread Safety:

This class is mutable, but it uses additional synchronization when accessing any mutable attribute (except reviewAssignmentManager and log attributes that are assumed to be immutable after initialization). It's assumed that configure() method will be called just once right after instantiation or static setConfig() method will be called instead, before calling any business methods. This class uses a not thread safe ReviewAssignmentManager instance, but it guarantees that it will be accessed from one thread only at a time (but not allowing to run two simultaneous jobs).

ReviewAssignmentManager

This class provides a programmatic API for review assignment. It provides just a single execute() method that uses pluggable ReviewAssignmentAlgorithm, ReviewAssignmentProjectManager and ReviewAssignmentNotificationManager instances to perform vairous steps of the assignment procedure.

This class performs the logging of errors and debug information using Logging Wrapper component.

Thread Safety:

This class is immutable, but not thread safe since it uses ReviewAssignmentAlgorithm, ReviewAssignmentProjectManager and ReviewAssignmentNotificationManager instances that are not guaranteed to be thread safe.

Configurable

This interface should be extended by interfaces and implemented by classes that can be configured from Configuration API object. It's assumed that configure() method defined in this interface will be called just once for each instance.

Thread Safety:

Implementations of this interface are not required to be thread safe.

ReviewAssignmentNotificationManager

This manager is responsible for sending notifications related to review assignment.

This interface extends Configurable interface to support configuration via Configuration API component.

Thread Safety:

Implementations of this interface are not required to be thread safe.

EmailBasedReviewAssignmentNotificationManager

Email based implementation of ReviewAssignmentNotificationManager, which sends notifications related to review assignment.

It uses EmailSendingUtility to send emails. It uses UserRetrieval and ResourceManager to retrieve necessary information.

This class performs the logging of errors and debug information using Logging Wrapper component.

Thread Safety:

This class is mutable and not thread safe since it uses ResourceManager instance that is not thread safe. It's assumed that configure() method will be called just once right after instantiation, before calling any business methods.

EmailSendingUtility

This is a helper utility class that is used by EmailBasedReviewAssignmentNotificationManager for sending warning email messages. This class supports constructing email message subjects and bodies from templates. It uses Document Generator component for this. This class uses Email Engine component to perform the actual sending of email messages.

Thread Safety:

This class is immutable and thread safe. It uses thread safe EmailEngine class.

ReviewAssignmentAlgorithm

This interface represents a review Assignment algorithm. It's responsible for generating an assignment (ideally an optimal assignment) of review applications to review roles.

This interface extends Configurable interface to support configuration via Configuration API component.

Thread Safety:

Implementations of this interface are not required to be thread safe.

BruteForceBasedReviewAssignmentAlgorithm

This is a base class for brute force based implementations of a review assignment algorithm.

It generates all possible assignments via brute force, lets subclass to measure the "quality" of assignment via protected measureQuality() method, and picks the best assignment.

This class performs the logging of errors and debug information using Logging Wrapper component.

See assign() method docs for details about assignment algorithm.

Thread Safety:

This class is mutable and not thread-safe.

MaxSumOfRatingReviewAssignmentAlgorithm

This class is an implementation of a review assignment algorithm. It's responsible for generating an assignment (ideally an optimal assignment) of review applications to review roles.

This class extends BruteForceBasedReviewAssignmentAlgorithm, overrides prepare() method to calculate users' ratings beforehand and measureQuality() method to calculate the sum of ratings of assigned reviewers. Thus, this class allows to find a maximum assignment with maximum sum of ratings of the assigned reviewers.

This class performs the logging of errors and debug information using Logging Wrapper component.

Thread Safety:

This class is mutable and not thread-safe.

ReviewAssignmentProjectManager

This manager is responsible for managing project resources and phases according to review assignment.

This interface extends Configurable interface to support configuration via Configuration API component.

Thread Safety:

Implementations of this interface are not required to be thread safe.

DefaultReviewAssignmentProjectManager

Default implementation of ReviewAssignmentProjectManager, which manages project resources and phases according to review assignment.

It uses UserRetrieval, ResourceManager and PhaseManager to manage necessary information.

This class performs the logging of errors and debug information using Logging Wrapper component.

Thread Safety:

This class is mutable and not thread safe since it uses ResourceManager instance that is not thread safe. It's assumed that configure() method will be called just once right after instantiation, before calling any business methods.

1.5 Component Exception Definitions

ReviewAssignmentConfigurationException

This runtime exception indicates error with component configuration.

Thread Safety:

This class is not thread safe because its base class is not thread safe.

ReviewAssignmentException

This exception is a base class for all other custom checked exceptions defined in this component. It indicates a general error in the component.

Thread Safety:

This class is not thread safe because its base class is not thread safe.

ReviewAssignmentAlgorithmException

This exception indicates error in review assignment algorithm.

Thread Safety:

This class is not thread safe because its base class is not thread safe.

ReviewAssignmentNotificationException

This exception indicates error with preparing or sending a notification.

Thread Safety:

This class is not thread safe because its base class is not thread safe.

ReviewAssignmentProjectManagementException

This exception indicates error with project management.

Thread Safety:

This class is not thread safe because its base class is not thread safe.

1.6 Thread Safety

Anyways, the component is not explicitly required to be thread-safe since it will be used as a standalone Java.

The component is not thread safe, because it contains non thread-safe classes (due to mutability and utilizing non thread-safe external classes). See class level documentation for details per each class.

2. Environment Requirements

2.1 Environment

Development language: Java 1.5

Compile target: Java 1.5, Java 1.6

QA Environment: Java 1.5, RedHat Linux 4, Windows 2000, Windows 2003

2.2 TopCoder Software Components

Base Exception 2.0 – is used by custom exceptions defined in this component.

Configuration API 1.1.0 – is used for initializing classes from this component.

Configuration Persistence 1.0.2 – is used for reading configuration from file.

Logging Wrapper 2.0.0 – is used for logging errors and debug information.

Email Engine 3.2.0 – is used for sending email messages.

Document Generator 3.1.1 – is used for generating text of email messages from templates.

Command Line Utility 1.0.0 – is used for parsing command line arguments.

DB Connection Factory 1.1.1 – is used for creating database connections.

Object Factory 2.2.0 – is used for creating pluggable object instances.

Object Factory Configuration API Plugin 1.1 – allows to use Configuration API for creating Object Factory.

Search Builder 1.4.1 – defines Filter entity used in this component.

Job Scheduling 3.2.0 – is used for loading job schedule from ConfigurationObject, defines ScheduledJobRunner interface implemented in this component.

Job Processor 3.0.1 – is used for executing review assignment job periodically.

User Project Data Store 1.1.2 – defines UserRetrieval interface and ExternalUser entity used in this component.

Project Management 1.2.2 – defines ProjectManager and Project entity used in this component.

Phase Management 1.1.0 – defines PhaseManager interface used in this component.

Project Phases 2.0.2 – defines phase specific entities used in this component.

Resource Management 1.3.0 – defines ResourceManager interface and Resource entity used in this component.

Review Application Management 1.0.0 – defines review application and auction related entities and managers, used in this component.

Forums EJB (not a component) – defines EJB used to grant forum access to approved applicants.

NOTE: 1. the dependencies of above components should be configured.

2. The default location for TopCoder Software component jars is../lib/tcs/COMPONENT_NAME/COMPONENT_VERSION relative to the component installation. Setting the tcs_libdir property in topcoder_global.properties will overwrite this default location.

2.3 Third Party Components

Apache Log 1.2.16 – provides PropertyConfigurator.

3. Installation and Configuration

3.1 Package Name

com.topcoder.management.review.assignment

com.topcoder.management.review.assignment.algorithm

com.topcoder.management.review.assignment.notification

com.topcoder.management.review.assignment.project

com.topcoder.management.review.assignment.utility

3.2 Configuration Parameters

3.2.1 ReviewAssignmentUtility

3.2.1.1 File based configuration

The following table describes the structure of ConfigurationObject used in the main() method of ReviewApplicationUtility class (angle brackets are used for identifying child configuration objects). This ConfigurationObject is read from a configuration file using Configuration Persistence component.

Parameter	Description	Values
loggerName	The name of Logging Wrapper logger to be used for logging errors and debug information. When not provided, logging is not performed.	String. Not empty. Optional.
log4jConfigFile	log4j ConfigFile used to configure PropertyConfigurator	String. Not empty. Optional.
schedulerConfig	The configuration used for creating ConfigurationObjectScheduler instance.	ConfigurationObject. Required.

jobName	The name of the job from scheduler configuration that corresponds to ReviewAssignmentJobRunner. Note that the type of this job must be “JOB_TYPE_JAVA_CLASS”, and run command – equal to the full class name of ReviewAssignmentJobRunner.	String. Not empty. Required.
jobConfig	The configuration used for initializing ReviewAssignmentJobRunner.	ConfigurationObject. Required.

3.2.1.2 Command line parameters

The following table describes the command line switches and arguments that are supported by ReviewAssignmentUtility standalone application.

Switch and arguments	Description
-c <file_name>	Optional. Provides the name of the configuration file for this command line application. This file is read with use of Configuration Persistence component. Default is “com/topcoder/management/review/assignment/utility/ ReviewAssignmentUtility.properties”.
-ns <namespace>	Optional. The namespace in the specified configuration file that contains configuration for this command line application. Default is “com.topcoder.management.review.assignment.utility. ReviewAssignmentUtility”.
-trackingInterval <interval_in_sec>	Optional. The interval in seconds between checks of projects for review assignment. If not specified, the value from the scheduler configuration is used.
-guardFile <file_path>	Required. The path to guard file which should be used to signal to Review Assignment that it has to stop.
-background [true|false]	Required. The flag indicating whether the review assignment is going to run in background thread or not.
-help -? -h	When one of the specified switches is provided, the application prints out the usage string to the standard output and terminates immediately.

3.2.2 *ReviewAssignmentJobRunner*

Configured via configure(config:ConfigurationObject) method.

Parameter	Description	Value
loggerName	The name of Logging Wrapper logger to be used for logging errors and debug information. When not provided, logging is not performed.	String. Not empty. Optional.

3.2.3 *ReviewAssignmentManager*

Configured via constructor, which is passed with a ConfigurationObject instance.

Parameter	Description	Value
loggerName	The name of Logging Wrapper logger to be used for logging errors and debug information. When not provided, logging is not performed.	String. Not empty. Optional.
objectFactoryConfig	This section contains configuration of	ConfigurationObject.

	Object Factory used by this class for creating pluggable object instances.	Required.
reviewApplicationManagerKey	The Object Factory key that is used for creating an instance of ReviewApplicationManager to be used by this class.	String. Not empty. Required.
reviewAuctionManagerKey	The Object Factory key that is used for creating an instance of ReviewAuctionManager to be used by this class.	String. Not empty. Required.
projectManagerKey	The Object Factory key that is used for creating an instance of ProjectManager to be used by this class.	String. Not empty. Required.
reviewAssignmentAlgorithmKey	The Object Factory key that is used for creating an instance of ReviewAssignmentAlgorithm to be used by this class.	String. Not empty. Required.
reviewAssignmentProjectManagerKey	The Object Factory key that is used for creating an instance of ReviewAssignmentProjectManager to be used by this class.	String. Not empty. Required.
reviewAssignmentNotificationManagerKey	The Object Factory key that is used for creating an instance of reviewAssignmentNotificationManager to be used by this class.	String. Not empty. Required.
reviewAssignmentAlgorithmConfig	The configuration of ReviewAssignmentAlgorithm to be used by this class.	ConfigurationObject. Required.
reviewAssignmentProjectManagerConfig	The configuration of ReviewAssignmentProjectManager to be used by this class.	ConfigurationObject. Required.
reviewAssignmentNotificationManagerConfig	The configuration of ReviewAssignmentNotificationManager to be used by this class.	ConfigurationObject. Required.
pendingReviewApplicationStatusId	ID of pending ReviewApplicationStatus.	long. Required.
approvedReviewApplicationStatusId	ID of approved ReviewApplicationStatus.	long. Required.
rejectedReviewApplicationStatusId	ID of rejected ReviewApplicationStatus.	long. Required.

3.2.4 *EmailBasedReviewAssignmentNotificationManager*

Configured via configure(config:ConfigurationObject) method.

Parameter	Description	Value
loggerName	The name of Logging Wrapper logger to be used for logging errors and debug information. When not provided, logging is not performed.	String. Not empty. Optional.
objectFactoryConfig	This section contains configuration of Object Factory used by this class for creating pluggable object instances.	ConfigurationObject. Required.
resourceManagerKey	The Object Factory key that is used for creating an instance of ResourceManager to be used by this class.	String. Not empty. Required.

userRetrievalKey	The Object Factory key that is used for creating an instance of UserRetrieval to be used by this class.	String. Not empty. Required.
approvedApplicantEmailSubjectTemplateText	The email subject template text to be used for constructing messages to be sent to the approved applicants.	String. Not empty. Required.
approvedApplicantEmailBodyTemplatePath	The email body template path (resource path or file path) to be used for constructing messages to be approved applicants.	String. Not empty. Required.
rejectedApplicantEmailSubjectTemplateText	The email subject template text to be used for constructing messages to be sent to the rejected applicants.	String. Not empty. Required.
rejectedApplicantEmailBodyTemplatePath	The email body template path (resource path or file path) to be used for constructing messages to be rejected applicants.	String. Not empty. Required.
managerEmailSubjectTemplateText	The email subject template text to be used for constructing messages to be sent to the managers.	String. Not empty. Required.
managerApplicantEmailBodyTemplatePath	The email body template path (resource path or file path) to be used for constructing messages to be sent to the managers.	String. Not empty. Required.
managerResourceRoleIds	The set of IDs of the manager resource roles that represent managers to be notified about review assignment.	long[]. Not empty. Required.
emailSender	The email sender to be used for constructing messages to be sent.	String. Not empty. Required.

3.2.5 *BruteForceBasedReviewAssignmentAlgorithm*

Configured via configure(config:ConfigurationObject) method.

Parameter	Description	Value
loggerName	The name of Logging Wrapper logger to be used for logging errors and debug information. When not provided, logging is not performed.	String. Not empty. Optional.

3.2.6 *MaxSumOfRatingReviewAssignmentAlgorithm*

Configured via configure(config:ConfigurationObject) method.

Parameter	Description	Value
connectionName	The name of connection in database connection factory. When not provided, default connection will be retrieved.	String. Not empty. Optional.
dbConnectionFactoryConfig	The configuration to be used for creating DBConnectionFactoryImpl instance.	ConfigurationObject. Required.
minimumFeedbacks	Represents the minimal amount of feedbacks, which reviewer should have, in order to have non-default rating.	int. Optional. Positive. Default value is 4.
defaultRating	Represents the default review rating.	double. Optional. Non-negative.

		Default value is 1.0.

3.2.7 *DefaultReviewAssignmentProjectManager*

Configured via configure(config:ConfigurationObject) method.

Parameter	Description	Value
loggerName	The name of Logging Wrapper logger to be used for logging errors and debug information. When not provided, logging is not performed.	String. Not empty. Optional.
objectFactoryConfig	This section contains configuration of Object Factory used by this class for creating pluggable object instances.	ConfigurationObject. Required.
userRetrievalKey	The Object Factory key that is used for creating an instance of UserRetrieval to be used by this class.	String. Not empty. Required.
phaseManagerKey	The Object Factory key that is used for creating an instance of PhaseManager to be used by this class.	String. Not empty. Required.
resourceManagerKey	The Object Factory key that is used for creating an instance of ResourceManager to be used by this class.	String. Not empty. Required.
registrationDateFormatString	Format string for formatting registration date.	String. Not empty. Optional. Default value is "MM.dd.yyyy hh:mm".
forumsBeanUrl	ForumsBean URL.	String. Not empty. Required.
forumsBeanFactory	ForumsBean context factory name.	String. Not empty. Required.
forumsBeanName	ForumsBean bean name.	String. Not empty. Required.
phaseTypeExtensionRules	Configuration object, which contains mapping of phase type IDs to extensions lengths (in seconds). All keys and values should be castable to "long" type, values must be non-negative.	ConfigurationObject. Required.
operator	Operator, used to update phases via PhaseManager.	String. Not empty. Required.

3.3 Dependencies Configuration

Please see docs of dependency components to configure them properly.

3.3.1 *Notification email templates*

3.3.1.1 Approved applicant email

Used by EmailBasedReviewAssignmentNotificationManager.notifyApprovedApplicants() method.

Subject:

Your review application has been approved.

Body:

```
<p>
%USER_HANDLE%, your review application has been approved. Details are provided below.
Project ID\: %PROJECT_ID%
Project Name\: %PROJECT_NAME%
Project Version\: %PROJECT_VERSION%
Review application role\: %REVIEW_APPLICATION_ROLE_NAME%
</p>
```

3.3.1.2 Rejected applicant email

Used by EmailBasedReviewAssignmentNotificationManager.notifyRejectedApplicants() method.

Subject:

Your review application has been rejected.

Body:

```
<p>
%USER_HANDLE%, your review application has been rejected. Details are provided below.
Project ID\: %PROJECT_ID%
Project Name\: %PROJECT_NAME%
Project Version\: %PROJECT_VERSION%
</p>
```

3.3.1.3 Manger/copilot email

Used by EmailBasedReviewAssignmentNotificationManager.notifyManagers() method.

Subject:

Reviewers have been assigned to your project.

Body:

```
<p>
Reviewers have been assigned to your project. Details are provided below.
Project ID\: %PROJECT_ID%
Project Name\: %PROJECT_NAME%
Project Version\: %PROJECT_VERSION%
Review assignment\:
%loop:ASSIGNMENTS%
Reviewer %USER_HANDLE% assigned to role "%REVIEW_APPLICATION_ROLE_NAME%".
%endloop%

%if: OPEN_POSITIONS>'0'%
Please note that there's no enough reviewers for the contest!
%endif%
</p>
```

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- 1). create database tcs_catalog with buffered log
- 2). populate tables with the script provided by PM and then run init.sql
- 3). modify the db connection properties in xml configuration file.
- 4). start devnullsmtp.jar
- 5). 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 Command line usage

This command line can be used to print out the usage string:

```
java com.topcoder.management.review.assignment.utility.ReviewAssignmentUtility -help
```

If configuration for the utility is stored in the default namespace of the default configuration file, then the application can be executed in background with the following arguments:

```
java com.topcoder.management.review.assignment.utility.ReviewAssignmentUtility –guardFile
guard.tmp –background true
```

To use the custom configuration file the user can provide "-c" switch:

```
java com.topcoder.management.review.assignment.utility.ReviewAssignmentUtility –c
custom_config.properties –guardFile guard.tmp –background true
```

The user can specify custom import files utility configuration file name and namespace:

```
java com.topcoder.management.review.assignment.utility.ReviewAssignmentUtility –c
custom_config.properties –ns custom_namespace –guardFile guard.tmp –background true
```

4.3.2 API usage

```
       // How to assign the reviewer.
            ReviewAuction reviewAuction = new ReviewAuction();
            Map<ReviewApplication, ReviewApplicationRole> assignment = new
HashMap<ReviewApplication, ReviewApplicationRole>();
            ReviewApplicationRole reviewApplicationRole = new
ReviewApplicationRole(1, "Reviewer", 2, null);
            List<ReviewApplication> reviewApplications = new
ArrayList<ReviewApplication>();

            reviewApplications.clear();
            ReviewApplication app1 = new ReviewApplication();
            app1.setUserId(1);
            app1.setApplicationRoleId(1);
            ReviewApplication app2 = new ReviewApplication();
            app2.setUserId(2);
            app2.setApplicationRoleId(1);
            ReviewApplication app3 = new ReviewApplication();
            app3.setUserId(3);
            app3.setApplicationRoleId(1);
            reviewApplications.add(app1);
            reviewApplications.add(app2);
            reviewApplications.add(app3);

            assignment.clear();
            assignment.put(app1, reviewApplicationRole);
            assignment.put(app2, reviewApplicationRole);
            assignment.put(app3, reviewApplicationRole);

            ReviewAuctionCategory auctionCategory = new ReviewAuctionCategory(1,
"Contest Review");
            List<ReviewApplicationRole> applicationRoles = new
ArrayList<ReviewApplicationRole>();

            ReviewApplicationRole rar = new ReviewApplicationRole(1, "Reviewer",
2,
                new ArrayList<ReviewApplicationResourceRole>());
            applicationRoles.add(rar);

            ReviewAuctionType auctionType = new ReviewAuctionType(1,
"developement contest", auctionCategory,
                applicationRoles);

            reviewAuction.setProjectId(1);
            reviewAuction.setAuctionType(auctionType);
            List<Long> openPositions = new ArrayList<Long>();
            openPositions.add(new Long(1));
            openPositions.add(new Long(2));

            reviewAuction.setOpenPositions(openPositions);

            BruteForceBasedReviewAssignmentAlgorithm algo = new
MaxSumOfRatingReviewAssignmentAlgorithm();
            ConfigurationObject algoConfig = getConfigurationObject(
                "test_files/config/ReviewAssignmentAlgorithm.xml",
                MaxSumOfRatingReviewAssignmentAlgorithm.class.getName());
            algo.configure(algoConfig);

            algo.assign(reviewAuction, reviewApplications);

            // How to execute assignment manager.
            ConfigurationObject ramConfig = getConfigurationObject(
                "test_files/config/ReviewAssignmentManager.xml",
ReviewAssignmentManager.class.getName());

           ReviewAssignmentManager ram = new ReviewAssignmentManager(ramConfig);
            ram.execute();

            // How to notify member.
            ReviewAssignmentNotificationManager ranm = new
EmailBasedReviewAssignmentNotificationManager();

            ConfigurationObject ranmConfig = getConfigurationObject(

"test_files/config/EmailBasedReviewAssignmentNotificationManager.xml",
                EmailBasedReviewAssignmentNotificationManager.class.getName());
            ranm.configure(ranmConfig);

            reviewAuction = new ReviewAuction();
            openPositions = new ArrayList<Long>();
            openPositions.add(new Long(2));
            reviewAuction.setOpenPositions(openPositions);
            ReviewApplication reviewApplication = new ReviewApplication();
            reviewApplication.setUserId(100008);

           assignment = new HashMap<ReviewApplication, ReviewApplicationRole>();
            assignment.put(reviewApplication, reviewApplicationRole);

            List<Long> unassignedUserIds = new ArrayList<Long>();
            unassignedUserIds.add(new Long(12345));
            ProjectStatus projectStatus = new ProjectStatus(1, "developement");
            Project project = new Project(1, ProjectCategory.BANNERS_ICONS,
projectStatus);
            project.setProperty("Project Name", "Review Assignment");
            project.setProperty("Project Version", "1.0.0");

            // Show how to use notify
            ranm.notifyApprovedApplicants(reviewAuction, project, assignment);

            ranm.notifyRejectedApplicants(reviewAuction, project,
unassignedUserIds);

            ranm.notifyManagers(reviewAuction, project, assignment);
```

5. Future Enhancements

Add more granular review assignment algorithm.