Reliability Calculator 1.0 Component Specification

1. Design

Reliability rating is a factor to measure how reliably a member has shown during recent projects. The reliability bonus information can be found at http://www.topcoder.com/wiki/display/tc/Component+Development+Reliability+Ratings (for development/design). Reliability ratings are calculated by an utility that is run each night. The existing code has some serious logical problems and some architectural issues which make it difficult to maintain. The goal of this project is to re-design the calculator utility with clear rules and flexible design.

The new utility is designed in a way that allows running it in parallel with the existing one to compare the results. One of the implications of this is that the new utility writes the results to a separate table in the DB to avoid the data conflicts.

This component provides ReliabilityCalculationUtility command line utility. This utility uses pluggable ReliabilityCalculator implementation instances per each project category.

To make this component as flexible as possible the provided implementation of ReliabilityCalculator (ReliabilityCalculatorImpl) uses pluggable ReliabilityDataPersistence, UserReliabilityCalculator, ResolutionDateDetector and UserProjectParticipationDataComparator implementation instances.

Also in future it will be possible to extend UserProjectParticipationData and UserProjectReliabilityData entities by adding additional properties to them without making any changes to interfaces.

1.1 Design Patterns

Strategy pattern – ReliabilityCalculationUtility uses pluggable ReliabilityCalculator implementation instances; ReliabilityCalculatorImpl uses pluggable ReliabilityDataPersistence, UserReliabilityCalculator, ResolutionDateDetector and UserProjectParticipationDataComparator implementation instances.

DAO/DTO pattern – ReliabilityDataPersistence is a DAO for UserProjectReliabilityData and UserProjectParticipationData DTOs in this design (though it supports just retrieval operation for participation data and saving operation for reliability data).

Template method pattern – BaseUserReliabilityCalculator#calculate() is a template method that uses abstract calculateReliabilityAfterProjects() method that is implemented by subclasses.

1.2 Industry Standards

JDBC, SQL, JavaBeans

1.3 Required Algorithms

1.3.1 Logging

This component must perform logging in all public methods of ReliabilityCalculationUtility, ReliabilityCalculatorImpl, BaseUserReliabilityCalculator, DatabaseReliabilityDataPersistence and PhaseEndTimeBasedResolutionDateDetector except configure().

All information described below must be logged using log:Log attribute (or log:Log variable for ReliabilityCalculationUtility). If log attribute is null, then logging is not required to be performed.

In all mentioned methods method entrance with input argument, method exit with return value and call duration time must be logged at DEBUG level. It's not required to log method exit when method throws an exception.

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

Additionally the following information should be logged at INFO level in the specified methods:

- ReliabilityCalculationUtility#main(): application start/end timestamps, execution time, updateCurrentReliability flag (whether current reliability should be updated in DB) and projectCategoryIds list (the list of IDs of project categories for which reliability should be

calculated).

- ReliabilityCalculatorImpl#calculate(): project category ID and timestamp at the beginning; timestamp, execution time and userIds.size() at the end; user ID and current reliability for each processed user.

1.3.2 Database schema update

This design requires a new table to be added to the tcs_catalog database. Please see tcs_catalog_schema_update.ddl file provided together with this specification for details.

1.3.3 Calculating the reliability

This component supports two types of reliability calculation formulas. Please see method docs and implementation notes of UniformUserReliabilityCalculator and WeightedUserReliabilityCalculator classes for details.

In future support for other reliability calculation formulas can be added by providing another UserReliabilityCalculator implementations.

The high level description of the algorithm used for calculating and updating the reliability for specific project category is provided below (it's assumed that pluggable interface implementations provided in the current version of the component are used).

- Select all members that have at least one record in project_result with passed_review_ind=1 for this project category.
- For each found user do:
    - Get information about all projects for specific user and project category. This includes submission status, registration date, end dates of some project phases, etc.
    - Detect resolution date for each obtained user/project pair. The provided detector implementation uses Submission/Screening/Appeals Response phase end dates as resolution dates.
    - Skip all projects for which resolution date cannot be detected.
    - Sort projects in which this user participated (currently sorting is performed by resolution date and project ID to resolve ties).
    - Skip all projects at the beginning of the sorted list for which the passed_review_ind=0.
    - Compute reliability after each project using the specific formula.
    - Use reliability after the previous project to set reliability before each project.
    - Detect reliability on registration date for each project.
    - Save reliability data for all projects to project_reliability table.
    - If requested, update the current reliability of the user in user_reliability table.

1.4 Component Class Overview

BaseUserProjectData [abstract]

This is a base class for UserProjectReliabilityData and UserProjectParticipationData. It holds fields that are common for both entities: project ID, user ID and resolution date. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

BaseUserReliabilityCalculator [abstract]

This class is an abstract implementation of UserReliabilityCalculator that performs tasks that are common for all reliability calculators: initializes UserProjectReliabilityData

instances, sets reliability before the projects and reliability on the registration dates using the reliability after the projects calculated in calculateReliabilityAfterProjects() method to be implemented by subclasses.

Configurable [interface]
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.

DatabaseReliabilityDataPersistence
This class is an implementation of ReliabilityDataPersistence that uses DB Connection Factory and JDBC to access reliability specific data in the database persistence. It caches a Connection instance between open() and close() calls.

PhaseEndTimeBasedResolutionDateDetector
This class is an implementation of ResolutionDateDetector that uses phase end times to detect the resolution dates. If a user registered but did not submit, the resolution date is the end of the submission phase. If a user submitted but did not pass screening, the resolution date is the end of the screening phase. If a user passed screening, the resolution date is the end of the appeals response phase (regardless of whether the member passed review or not).

ProjectCategoryParams
This is an inner class of ReliabilityCalculatorImpl that is a container for reliability calculation parameters for specific project category. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

ReliabilityCalculationUtility
This is the main class of the standalone command line application that calculates and updates users' reliability ratings for specific project categories. Actually it just uses multiple pluggable ReliabilityCalculator instances to perform this task. This utility reads a configuration from a file using Configuration Persistence and Configuration API components. ReliabilityCalculationUtility performs the logging of errors and debug information using Logging Wrapper.

ReliabilityCalculator [interface]
This interface represents a reliability calculator that provides a method for calculating all users' reliability ratings for the specified project category and optionally updating current reliability for all users. Where to take the information for calculating the reliability from and where to save the calculated reliability ratings is up to implementations. This interface extends Configurable interface to support configuration via Configuration API component.

ReliabilityCalculatorImpl
This class is an implementation of ReliabilityCalculator that uses pluggable ReliabilityDataPersistence instance to access user participation data in persistence and write calculated user reliability data to persistence, and pluggable per project category ResolutionDateDetector and UserReliabilityCalculator instances to detect resolution dates and calculate actual reliability ratings for each user respectively. In the context of this implementation "resolution dates" are the moments when the information required to calculate the reliability comes in.

ReliabilityDataPersistence [interface]
This interface represents a reliability data persistence. Before and after accessing the persistence open() and close() methods of this interface must be called respectively. This interface defines methods for retrieving IDs of users that have reliability, retrieving participation data, saving reliability history data and updating current reliability of specific user. This interface extends Configurable interface to support configuration via Configuration API component.

ResolutionDateDetector [interface]
This interface represents a resolution date detector that provides a method for detecting

a resolution date for the specified user project participation data instance. This interface extends Configurable interface to support configuration via Configuration API component.

UniformUserReliabilityCalculator
This class is an implementation of UserReliabilityCalculator that uses information about N (or less if not enough exist yet) recent projects to calculate the reliability. The reliability calculation formula is A/B; here B is the number of recent projects taken into account (i.e. N or less), and A - the number of reliable projects among ones taken into account. I.e. impact of the recent projects on the reliability rating is distributed uniformly in this calculator.

UserProjectParticipationData
This class is a container for information about user's participation in one specific project. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

UserProjectParticipationDataComparator [interface]
This interface simply extends Comparator<UserProjectParticipationData> and doesn't define any new methods. It is used by ReliabilityCalculatorImpl for sorting UserProjectParticipationData instances.

UserProjectParticipationDataResolutionDateBasedComparator
This class is an implementation of UserProjectParticipationDataComparator that can be used for sorting UserProjectParticipationData instances by their resolution dates (ascendingly). It two UserProjectParticipationData instances have the same resolution date, they are compared using project ID properties to always provide consistent sorting results.

UserProjectReliabilityData
This class is a container for information about user's reliability corresponding to one project in which this user has participated. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

UserReliabilityCalculator [interface]
This interface represents a user reliability calculator that provides a method for calculating reliability ratings for specific user based on his full project participation data. This interface extends Configurable interface to support configuration via Configuration API component.

WeightedUserReliabilityCalculator
This class is an implementation of UserReliabilityCalculator that uses information about N (or less if not enough exist yet) recent projects to calculate the reliability. The reliability calculation formula is SUM(reliable_project_weights)/SUM(all_project_weights); here weights are configurable and are specified based on the chronological order of the projects: i.e. weight for the most recent project, weight for the last but one project, etc. The expected usage of this class is to use higher weight for newer projects and lower weights for older projects.

1.5 Component Exception Definitions

ProjectCategoryNotSupportedException
This exception is thrown by implementations of ReliabilityCalculator when project category with the given ID is not supported by this reliability calculator.

ReliabilityCalculationException
This exception is thrown by implementations of ReliabilityCalculator when some other error occurs while calculating or updating the reliability. Also this exception is used as a base class for other specific custom exceptions.

ReliabilityCalculatorConfigurationException
This exception is thrown by implementations of Configurable when some error occurs

while initializing an instance using the given configuration.

ReliabilityDataPersistenceException
This exception is thrown by ReliabilityCalculatorImpl and implementations of ReliabilityDataPersistence when some error occurs while accessing the persistence.

ResolutionDateDetectionException
This exception is thrown by implementations of ResolutionDateDetector when some error occurs while detecting a resolution date.

UserReliabilityCalculationException
This exception is thrown by implementations of UserReliabilityCalculator when some error occurs while calculating the reliability rating.

1.6 Thread Safety

This component is not thread safe.

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

Implementations of ReliabilityDataPersistence, UserReliabilityCalculator, ResolutionDateDetector, UserProjectParticipationDataComparator, ReliabilityCalculator and Configurable are not required to be thread safe.

ReliabilityCalculatorImpl is mutable and not thread safe. It uses ReliabilityDataPersistence, UserReliabilityCalculator and ResolutionDateDetector instances that are not required to be thread safe. It's assumed that configure() method will be called just once right after instantiation, before calling any business methods.

DatabaseReliabilityDataPersistence is mutable and not thread safe. It's assumed that configure() method will be called just once right after instantiation, before calling any business methods. saveUserReliabilityData() and updateCurrentUserReliability() methods that can modify data in persistence use transactions.

UserProjectParticipationDataResolutionDateBasedComparator is immutable and thread safe when UserProjectParticipationData instances passed to it are used by the caller in thread safe manner.

UniformUserReliabilityCalculator, BaseUserReliabilityCalculator, WeightedUserReliabilityCalculator and PhaseEndTimeBasedResolutionDateDetector are mutable and not thread safe. But it's assumed that configure() method will be called just once right after instantiation, before calling any business methods (in this case this class is used in thread safe manner).

ProjectCategoryParams, BaseUserProjectData, UserProjectReliabilityData and UserProjectParticipationData are mutable and not thread safe entities.

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 1.0 – is used for initializing classes from this component.

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

Logging Wrapper 2.0 – is used for logging errors and debug information.

Command Line Utility 1.0 – is used for parsing command line arguments.

DB Connection Factory 1.1 – is used for creating database connections.

Object Factory 2.1.2 – is used for creating pluggable object instances.

Object Factory Configuration API Plugin 1.0 – allows to use Configuration API for creating Object Factory.

NOTE: 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

None

3. Installation and Configuration

3.1 Package Name

com.topcoder.reliability
com.topcoder.reliability.impl
com.topcoder.reliability.impl.calculators
com.topcoder.reliability.impl.comparators
com.topcoder.reliability.impl.detectors
com.topcoder.reliability.impl.persistence

3.2 Configuration Parameters

3.2.1 Configuration of ReliabilityCalculationUtility

The following table describes the structure of ConfigurationObject used in the main() method of ReliabilityCalculationUtility 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.
<objectFactoryConfig>	This section contains configuration of Object Factory used by this class for creating pluggable object instances.	ConfigurationObject. Required.
<reliabilityCalculatorXXX>	Here XXX is any substring, e.g. “1”, “2”, “3”, etc. The configuration for one of ReliabilityCalculator instances to be used by this class.	ConfigurationObject. Multiple. At least one is required.
<reliabilityCalculatorXXX>.key	The Object Factory key that is used for creating an instance of ReliabilityCalculator.	String. Not empty. Required.
<reliabilityCalculatorXXX>.config	The configuration for ReliabilityCalculator instance.	ConfigurationObject. Required.

<reliabilityCalculatorXXX>.projectCategoryIds	The list of project category IDs for which this ReliabilityCalculator instance should be used. Each element in the list must be a string representation of a valid positive long integer. Note that ReliabilityCalculator instance must be configured to support all these project categories.	String[]. Not empty. Required.

3.2.2 Configuration of ReliabilityCalculatorImpl

The following table describes the structure of ConfigurationObject passed to the constructor of ReliabilityCalculatorImpl class (angle brackets are used for identifying child configuration objects).

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.
<objectFactoryConfig>	This section contains configuration of Object Factory used by this class for creating pluggable object instances.	ConfigurationObject. Required.
reliabilityDataPersistenceKey	The Object Factory key that is used for creating an instance of ReliabilityDataPersistence to be used by this class.	String. Not empty. Required.
reliabilityDataPersistence Config	The configuration of ReliabilityDataPersistence to be used by this class.	ConfigurationObject. Required.
participationDataComparator Key	The Object Factory key that is used for creating an instance of UserProjectParticipationDataComparator to be used by this class for sorting user participation data instances.	String. Not empty. Required.
<projectCategoryConfigXXX>	Here XXX is any substring, e.g. “1”, “2”, “3”, etc. This section contains configuration specific to some project category with the specified ID.	ConfigurationObject. Multiple. At least one is required.
<projectCategoryConfigXXX>.projectCategoryIds	The list of project category IDs to be associated with this configuration section. Each element in the list must be a string representation of a valid positive long integer.	String[]. Not empty. Required.
<projectCategoryConfigXXX>.reliabilityStartDate	The date/time when reliability for this project category started to be counted in format “yyyy-MM-dd HH:mm”.	String. Not empty. Required.
<projectCategoryConfigXXX>.userReliabilityCalculatorKey	The Object Factory key that is used for creating an instance of UserReliabilityCalculator to be used for this project category.	String. Not empty. Required.

<projectCategoryConfigXXX>.userReliabilityCalculatorConfig	The configuration of UserReliabilityCalculator instance to be used for this project category.	ConfigurationObject. Required.
<projectCategoryConfigXXX>.resolutionDateDetectorKey	The Object Factory key that is used for creating an instance of ResolutionDateDetector to be used for this project category.	String. Not empty. Required.
<projectCategoryConfigXXX>.resolutionDateDetectorConfig	The configuration of ResolutionDateDetector instance to be used for this project category.	ConfigurationObject. Required.

3.2.3 Configuration of DatabaseReliabilityDataPersistence

The following table describes the structure of ConfigurationObject passed to the constructor of DatabaseReliabilityDataPersistence class (angle brackets are used for identifying child configuration objects).

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.
dbConnectionFactoryConfig	The configuration to be used for creating DBConnectionFactoryImpl instance.	ConfigurationObject. Required.
connectionName	The connection name to be passed to the connection factory when establishing a database connection. If not specified, a default connection is used.	String. Not empty. Optional.

3.2.4 Configuration of PhaseEndTimeBasedResolutionDateDetector

The following table describes the structure of ConfigurationObject passed to the constructor of PhaseEndTimeBasedResolutionDateDetector class (angle brackets are used for identifying child configuration objects).

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.

3.2.5 Configuration of BaseUserReliabilityCalculator subclasses

The following table describes the structure of ConfigurationObject passed to the constructor of BaseUserReliabilityCalculator class (angle brackets are used for identifying child configuration objects).

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.

3.2.6 Configuration of UniformUserReliabilityCalculator

The following table describes the structure of ConfigurationObject passed to the constructor of UniformUserReliabilityCalculator class (angle brackets are used for identifying child configuration objects).

Parameter	Description	Values
historyLength	The history length that represents the number of recent resolved projects that affect the user's reliability.	String representation of a valid positive integer. Required.

Please see section 3.2.5 for additional configuration parameters used by the base class.

3.2.7 Configuration of WeightedUserReliabilityCalculator

The following table describes the structure of ConfigurationObject passed to the constructor of WeightedUserReliabilityCalculator class (angle brackets are used for identifying child configuration objects).

Parameter	Description	Values
weights	The comma separated list of weights for the recent resolved projects that affect the user's reliability. The last element in the list corresponds to the most recently resolved project. Each element in the comma separated list must be a valid positive double.	String. Not empty. Required.

Please see section 3.2.5 for additional configuration parameters used by the base class.

3.2.8 ReliabilityCalculationUtility command line parameters

The following table describes the command line switches and arguments that are supported by ReliabilityCalculationUtility 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/reliability/ ReliabilityCalculationUtility.properties”.
-ns <namespace>	Optional. The namespace in the specified configuration file that contains configuration for this command line application. Default is “com.topcoder.reliability.ReliabilityCalculationUtility”.
-pc <project_category_ids>	Optional. The comma separated IDs of project categories for which reliability calculation must be performed. If not specified, reliability is calculated for all project categories mentioned in the configuration (see <reliabilityCalculatorXXX>.projectCategoryIds parameter in the section 3.2.1).
-u [yes|no]	Optional. The value indicating whether current reliability ratings must be updated for all users. The only correct arguments are “yes” and “no” (case insensitive). Default is “no”.
-help -? -h	When one of the specified switches is provided, the application prints out the usage string to the standard output and terminates immediately.

3.3 Dependencies Configuration

Please see docs of dependency components to configure them properly.

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

Please see the demo.

4.3 Demo

4.3.1 API usage

```
// Create an instance of ReliabilityCalculatorImpl
ReliabilityCalculatorImpl reliabilityCalculator = new ReliabilityCalculatorImpl();

// Configure reliability calculator
ConfigurationObject config = ...
reliabilityCalculator.configure(config);

// Calculate reliability for "Design" project category (with ID=1) and
// update current reliability ratings for all users
reliabilityCalculator.calculate(1, true);
```

4.3.2 Usage of command line utility

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

```
java com.topcoder.reliability.ReliabilityCalculationUtility -help
```

If configuration for the utility is stored in the default namespace of the default configuration file, then the application can be executed without additional arguments:

```
java com.topcoder.reliability.ReliabilityCalculationUtility
```

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

```
java com.topcoder.reliability.ReliabilityCalculationUtility -c custom_config.properties
```

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

```
java com.topcoder.reliability.ReliabilityCalculationUtility -c cusom_config.properties -ns custom_namespace
```

The following command line allows to calculate reliability for project categories with ID=1,2 and to update the current reliability ratings of users for these project categories:

```
java com.topcoder.reliability.ReliabilityCalculationUtility -pc 1,2 -u yes
```

4.3.3 Sample ReliabilityCalculationUtility configuration file

The file provided in this section is not a file that is specified as the command line argument. Instead the command line should contain the name of the properties file passed to ConfigurationFileManager. And this properties file should contain the link to the actual ReliabilityCalculationUtility configuration file, sample of which is provided in this section.

```
<?xml version="1.0"?>
<CMConfig>
  <Config name="com.topcoder.reliability.ReliabilityCalculationUtility">
    <Property name="loggerName">
      <Value>myLogger</Value>
    </Property>
    <Property name="objectFactoryConfig">
      <!-- Put Object Factory configuration here -->
    </Property>
```

```
<Property name="reliabilityCalculator1">
  <Property name="projectCategoryIds">
    <Value>1</Value>
    <Value>2</Value>
    <Value>6</Value>
    <Value>7</Value>
    <Value>13</Value>
    <Value>14</Value>
    <Value>19</Value>
    <Value>23</Value>
    <Value>24</Value>
    <Value>26</Value>
  </Property>
  <Property name="key">
    <Value>ReliabilityCalculatorImpl</Value>
  </Property>
  <Property name="config">
    <Property name="loggerName">
      <Value>myLogger</Value>
    </Property>
    <Property name="objectFactoryConfig">
      <!-- Put Object Factory configuration here -->
    </Property>
    <Property name="reliabilityDataPersistenceKey">
      <Value>DatabaseReliabilityDataPersistence</Value>
    </Property>
    <Property name="reliabilityDataPersistenceConfig">
      <Property name="loggerName">
        <Value>myLogger</Value>
      </Property>
      <Property name="dbConnectionFactoryConfig">
        <!-- Put DB Connection Factory configuration here -->
      </Property>
      <Property name="connectionName">
        <Value>myConnection</Value>
      </Property>
    </Property>
    <Property name="participationDataComparatorKey">
      <Value>UserProjectParticipationDataResolutionDateBasedComparator</Value>
    </Property>
    <Property name="projectCategoryConfig1">
      <Property name="projectCategoryIds">
        <Value>1</Value>
        <Value>2</Value>
      </Property>
      <Property name="reliabilityStartDate">
        <Value>2005-10-05 09:00</Value>
      </Property>
      <Property name="userReliabilityCalculatorKey">
        <Value>UniformUserReliabilityCalculator</Value>
      </Property>
      <Property name="userReliabilityCalculatorConfig">
        <Property name="loggerName">
          <Value>myLogger</Value>
        </Property>
        <Property name="historyLength">
          <Value>15</Value>
        </Property>
      </Property>
      <Property name="resolutionDateDetectorKey">
        <Value>PhaseEndTimeBasedResolutionDateDetector</Value>
      </Property>
      <Property name="resolutionDateDetectorConfig">
        <Property name="loggerName">
          <Value>myLogger</Value>
        </Property>
      </Property>
    </Property>
    <Property name="projectCategoryConfig2">
      <Property name="projectCategoryIds">
        <Value>6</Value>
        <Value>7</Value>
```

```
            <Value>13</Value>
            <Value>14</Value>
            <Value>19</Value>
            <Value>23</Value>
            <Value>24</Value>
            <Value>26</Value>
          </Property>
          <Property name="reliabilityStartDate">
            <Value>2009-03-23 00:00</Value>
          </Property>
          <Property name="userReliabilityCalculatorKey">
            <Value>WeightedUserReliabilityCalculator</Value>
          </Property>
          <Property name="userReliabilityCalculatorConfig">
            <Property name="loggerName">
              <Value>myLogger</Value>
            </Property>
            <Property name="weights">
              <Value>0.82</Value>
              <Value>0.84</Value>
              <Value>0.86</Value>
              <Value>0.88</Value>
              <Value>0.9</Value>
              <Value>0.92</Value>
              <Value>0.94</Value>
              <Value>0.96</Value>
              <Value>0.98</Value>
              <Value>1.00</Value>
            </Property>
          </Property>
          <Property name="resolutionDateDetectorKey">
            <Value>PhaseEndTimeBasedResolutionDateDetector</Value>
          </Property>
          <Property name="resolutionDateDetectorConfig">
            <Property name="loggerName">
              <Value>myLogger</Value>
            </Property>
          </Property>
        </Property>
      </Property>
    </Property>
  </Config>
</CMConfig>
```

4.3.4 *Sample reliability calculation*

In this section it's assumed that configuration provided in the section 4.3.3 is used.

Assume that the table below describes all the projects from some category in which some user participated.

Project ID	Parameter	Value
1001	Participation status	Registered, but did not submit
	User registration time	2010-01-06 09:12
	Submission phase end	2010-01-10 09:00
1002	Participation status	Submission passed review
	User registration time	2010-01-08 09:25
	Appeals response phase end	2010-01-15 16:24
1003	Participation status	Submission failed screening
	User registration time	2010-01-14 10:30
	Screening phase end	2010-01-18 12:31
1004	Participation status	Submission passed review
	User registration time	2010-01-16 09:47
	Appeals response phase end	2010-01-24 21:48
1005	Participation status	Registered, but did not submit
	User registration time	2010-01-18 11:29
	Submission phase end	2010-01-22 09:00

Then projects should be sorted by their resolution dates in user's reliability history in the following manner:

Order	Project ID	Resolution Date	Reliable
1	1001	2010-01-10 09:00	No
2	1002	2010-01-15 16:24	Yes
3	1003	2010-01-18 12:31	No
4	1005	2010-01-22 09:00	No
5	1004	2010-01-24 21:48	Yes

Note that project with ID=1001 must be ignored when calculating the reliability (like all not reliable projects before the first reliable one).

Assuming that project category is "Design" (ID=1), the calculated reliability ratings must be the following:

Project ID	Reliability before	Reliability after	Reliability on registration
1002	NULL	100%	NULL
1003	100%	50%	NULL
1005	50%	33.33%	100%
1004	33.33%	50%	100%

Assuming that the user ID is equal to 123456, project_reliability table must be updated with the following records:

project_id	user_id	resolution_date	reliability_ before_ resolution	reliability_ after_ resolution	reliability_on _ registration	reliable _ ind
1002	123456	2010-01-15 16:24	NULL	1	NULL	1
1003	123456	2010-01-18 12:31	1	0.5	NULL	0
1005	123456	2010-01-22 09:00	0.5	0.3333333	1	0
1004	123456	2010-01-24 21:48	0.3333333	0.5	1	1

5. Future Enhancements

None