

1. Scope

1.1 Overview

Reliability rating is a factor to measure how reliably a member has shown during recent projects. You can find the reliability bonus information 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 will be designed in a way that allows to run it in parallel with the existing one to compare the results. One of the implications of this is that the new utility will write the results to a separate table in the DB to avoid the data conflicts. The reason for that is that there will be a test period when both utilities are run together each night. During the test period the data generated by the existing utility will still be used to calculate the bonuses and update the members' reliability details page. We will switch to the new data only after the new utility proves to be correct and stable. You should keep this in mind when designing the component to make it as flexible as possible in order to be seamlessly integrated.

1.2 Logic Requirements

1.2.1 The problems of existing code



Read the existing code!

The existing code can be found in the attachments.
It is essential that you study it thoroughly so that you well understand how the existing system works. There are many things in the existing code that simply could not be mentioned in this spec yet the designers are expected to provide a similar functionality in the component. If there are any questions about the existing code please post in the forum!

1. The reliability bonus is calculated based on the reliability 'before' the project. However, 'before' is a vague concept. 'The projects before' are defined as those which have completed or have finished the review by that time. There is no definite point of time for which the reliability value is computed. Currently the value for every project is re-calculated nightly. Because of the overlapping of projects, the reliability value may change on the next day (e.g., some projects which are not included in previous calculation have finished review). So by the existing code, the reliability is calculated in an inconsistent way.
2. The incomplete projects are also included in the last 15 projects when the reliability value is calculated, if they have finished review. However, the projects included in the calculation are ordered by the completion date, so when the incomplete projects complete, the order changes so that the reliability value changes too. This makes the bonus (which has already been paid) incorrect.
3. The reliability bonus is paid against the reliability value which is calculated after the project has completed. Because the overlapping of different projects for the same user, the reliability value at the payment time is undetermined, so is the bonus. However, members should be able to fully control their reliability and know the amount of bonus they could get beforehand.
Also since reliability is a measure of how reliable a member is, and the probability of whether he will submit and pass the review for the current project, it is better to make the bonus and its time of determination consistent.
4. The minimum passing score used to determine if the submission passed review is hard coded. Having the hard coded score is redundant since project_result.passed_review_ind column can be used to tell if the submission passed review or not.



1.2.2 The rules for reliability calculation

1.2.2.1 When the reliability is calculated?

The new system defines the reliability rating of a member in a track as a piecewise constant function of time. The function's values change only at certain moments of time, which are called the 'resolution dates'. Resolution dates are the moments when the information required to calculate the reliability comes in. This includes the following scenarios:

1. If a member registered but did not submit, the resolution date is the end of the submission phase.
2. If a member submitted but did not pass screening, the resolution date is the end of the screening phase.
3. If a member passed screening, the resolution date is the end of the appeals response phase (regardless of whether the member passed review or not).

The reliability is calculated nightly in each track for all the members who had passing submissions before (i.e. with at least one record in project_result table with passed_review_ind=1 for the corresponding track).

1.2.2.2 How the reliability is calculated?

Here are rules for how the reliability is calculated:

1. Reliability rating at some point of time T is calculated as the ratio of X to Y, where Y is a configurable history length N (currently it is 15) and X is the number of reliable projects the user signed up for among the last Y projects with resolution dates prior the time T. If there are less than N projects prior the time T then Y equals to the total number of projects the user signed up for prior the time T. All projects before the first reliable project are not counted. This is the rule that must be implemented in this version.
2. A submission is reliable if and only if the submission passes review.
3. If the user is unreliable for a project but he does not have any previous reliability record in the track, this project does not count either. The idea is that the reliability rating only start since the first project the user passed review for.
4. The reliability bonus for a project is determined by the reliability the member had at the time he registered for that project.

For each track there is a date that all projects posted before that date don't count towards the reliability. This date should be configurable on a per-track basis.

With the rules above it becomes easier for the members to track how their reliability ratings are computed and removes the ambiguities present in the existing system.

1.2.3 Persistence

We are going to use a new table 'tcs_catalog.project_reliability' to store the reliability records. The table contains the following columns:

1. Project ID
2. User ID
3. Resolution Date
4. Reliability before the resolution date
5. Reliability after the resolution date
6. Reliability at the moment the user registered for this project
7. Whether the user is reliable for this project

The new system also uses the tables existing in the current system, but the reliability results will be written to 'project_reliability' instead of the 'project_result' table.

The 'user_reliability' table should also be updated as it is done in the existing code. However, this should be optional because this table is used by the rest of the system and thus the data should be left intact during the test period.

1.2.4 Interfaces

The component should be designed flexible on at least the following aspects.

1. The algorithm to calculate the reliability. Currently it is 'the ratio of the number of reliable projects to a configured history length' but we should be able to switch to another scheme easily. Different tracks may have different schemes.
2. For the current scheme, the history length should be configurable. Also, it may be different for different tracks.

1.2.5 Thread-Safety


The component is not required to be thread-safe.

1.3 Required Algorithms

Designers need to describe the following algorithms:

1. Algorithm to calculate the reliability.

A reference algorithm is listed below. You are free to improve it or use your own algorithm, but make sure the rules in 1.2.2 are obeyed.

 For each track:

1. Select all members that have at least one record in project_result with passed_review_ind=1 (here only projects of the current track are considered)
2. For all users in the retrieved list
 - a. Get all records from the project_result for the specific user and the specific track
 - b. For each record
 - i. Find out the following values: the status of the submission (i.e. active, failed review, failed screening) or N/A if there was no submission at all, end dates of submission, screening and appeal response.
 - ii. Based on these values determine if we can compute the resolution date (date the information came in). For example, if the submission failed screening we take the screening end date, if there was no submission we take the submission end date etc.
 - iii. If there was no submission and the submission phase is still open, we don't know the information yet so we just skip this record.
 - iv. If there was submission but the screening phase is still open, skip the record too.
 - v. If there was submission, and it passed screening, but the project is still in review, appeal, or appeal response phase, skip the record too.
 - c. For all non-skipped records sort them by the resolution date.
 - i. Skip all the records at the beginning of the sorted list for which the passed_review_ind=0.
 - ii. Compute the reliability for each as the ratio of the number of records with passed_review_ind=1 in the previous 15 records (including the current record) to the total number of the previous 15 records. If there are less than 15 records it should be as many as there are of them.
 - d. For all non-skipped records, again,
 - i. Compute the reliability at the registration date for each record

Then the current reliability should be the last record in the sequence. Special care should be taken for the projects with no Screening phase.

1.4 Example of the Software Usage

The component will be used to nightly compute the reliability for all the members involved in a certain project of the tracks.



1.5 Future Component Direction

Any enhancement needs to be approved either in forum or in email with managers to eliminate over-complicating the component with useless functions.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None, only API and command line interface is required.

2.1.2 External Interfaces

A command line interface should be provided. It will be called in a shell script to run nightly to calculate the reliability for all the tracks. Please see ReliabilityRating.java for reference.

2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5, Java 1.6

2.1.4 Package Structure

com.topcoder.reliability

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The algorithm used to calculate the reliability.
- The concrete configurations for the algorithm used.
- List of tracks to compute reliability for.
- Whether to update the user_reliability table.
- The date on a per track basis since which projects are counted towards the reliability.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

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

3.2.3 Third Party Component, Library, or Product Dependencies:

Any third party library needs to be approved.

3.2.4 QA Environment:

- Java 1.5
- RedHat Linux 4
- Windows 2000



- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification
- DDL for all new and updated tables and/or relationships

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of TC UML Tool.