Review Statistics calculator Component Requirements Specification

Scope

Overview

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 review statistics collection functionality of the upgrade application.

Attention

This specification was the result of an old architecture. A lot may have changed since then in the Online Review components.
Designers are required to verify that the application requirements are met by the updates and propose solutions in case they are not.
We are not looking for a word by word implementation of this specification, but for your design expertise to make this work.

Logic Requirements

Statistics collection

After each review is completed, statistics about the review performance will be collected. This component will provide algorithms to compute statistics from evaluation scorecards and will also provide managers to store these statistics for the given reviewers.
These utilities will be used by the online review application to compute the statistics to store in the user's database at the completion of a project phase.

Statistics implementations

There are several implementations that will be created as part of this component.

- One statistics calculator that will use individual calculator for each statistics coefficient.
- Three individual statistics calculators.

ReviewerStatisticsCalculators

The interface will compute all the review statistics available for the given project.
It will do the following steps:

- Get the project by project id
- Get the project's submissions
- Get the reviews for the project's submission
- For each review comment (getComments()) take the reviewEvaluation field. This field will have one of the values defined in evaluation_type_lu lookup table
- Call the coverage and accuracy calculators
- Get the submission and original timelines; call the timeline reliability calculator like this:
  - For secondary reviewer only the review and appeal response deadline is used
  - For primary reviewers all the rest of the deadlines are used for calculating reliability
- Create ReviewStatistics object from the coefficients (note that for the primary reviewer accuracy and coverage will be a flag -1 and will not be stored in the database, they will only affect payment).
- Call all configured handlers
- Return the statistics. The caller will execute further processing and store them into the database.

Notes: the mapping between the types of evaluation and the coefficients to be used in the algorithms should be configurable at the implementation level.

Individual statistics calculators

The default calculator implementation will depend on 3 individual calculators. In general there will be as many calculators as statistics to be computed. Review statistics are just one kind of statistics but many others can be supported as well.
All 3 calculators will have default implementations that will work as described bellow. The totalEvaluationCoefficient value will be computed as the multiplication of each individual coeffiecient: Accuracy * Coverage * Timeline Reliability
Note that for the primary reviewer the accuracy and coverage flags (-1) will cancel each other which will result in the overall performance to be equal with the timeline reliability. This is accurate since the only thing that the primary reviewer should be sure to meet are timelines.
The default implementation will also have the ability to register a notification handler by the StatisticsHandler interface. This interface and implementations will be created in another component. After the computation of the statistics, the manager will call the handleStatisticsResut for each computed ReviewerStatistics object.

Timeline Reliability Calculator implementation

+calculateReliability(deadline:Date[],commited:Date[]):double
Online Review keeps track of when each deliverable (like a review scorecard is submitted). The project definition also contains a list of all the phases deadlines.
When calculating the timeline reliability coefficient the following aspects will be taken into account:

1. phases have different configurable weights - screening will affect reliability more than other phases since it will directly delay the project
2. the offset induced by the previous phases being late must be considered when evaluating the other phases submissions. This will avoid cases when the screening is very late and it causes the review to be late as well with respect to the original deadline, even if the reviewer was not late at all.
3. the algorithm will have a base penalty which will be multiplied with every base unit of time that the project is delayed with.

For example the coefficient can be decreased with 0.02 for every hour being late.
For a certain competition, if the reviewer meets the timeline in every phase applicable to he/she position, his timeline reliability is set to 1.0, otherwise, the timeline reliability is deducted by 0.05 for every "weighted" hours delayed. If more than 10 "weighted" hours delayed, this number is simply set to be 0.5. Note that this calculator is NOT weighted/compared with other reviewers. Timelines is a synthetic coefficient that is calculated for each review or otherwise each submitted deliverable compared to its submission deadline.

Coverage Calculator implementation

+calculateCoverage(reviews:Review[]):double[]
This implementation will maintain a mapping between the review comment evaluations and a coefficient. One example of such mapping (and the default one) is:

- Serious - 10 points
- Medium - 3 points
- Minor - 1 point
- Comment - 0 points
- Combined - 0 points
- Incorrect - 10 (negative) points

The algorithm will retrieve all review comments from the each review object passed in the arguments array. Typically there will be 2 of these, one for each secondary reviewer.
The review evaluation type will be retrieved. If it's any positive value it will be summed up - these will represent the raw scores.
The ratio for each reviewer will be computed as value/sum(all values).
The ratios are returned in a double[]
For example:

Submission	#Serious	#Medium	#Minor	#Comment	#Combined	#Incorrect
S1	2	3	6	21	4	0
S2	1	7	5	23	3	1

Then for submission S1,
A's raw coverage score is 1*10+2*3+8*1 = 24
B's raw coverage score is 2*10+3*3+6*1 = 35
So A's coverage for submission S1 is 24/(24+35)=0.407 and
B's coverage for submission S1 is 35/(24+35)=0.593.

Accuracy Calculator implementation

+calculateAccuracy(reviews:Review[]):double[]
This implementation will maintain a mapping between the review comment evaluations and a coefficient. One example of such mapping (and the default one) is:

- Serious - 10 points
- Medium - 3 points
- Minor - 1 point
- Comment - 0 points
- Combined - 0 points
- Incorrect - 10 (negative) points

The algorithm will retrieve all review comments from the each review object passed in the arguments array. Typically there will be 2 of these, one for each secondary reviewer.
The review evaluation type will be retrieved and if it is of type incorrect, a counter that starts from 0 is increased for the given review. This will be called raw accuracy count for a submission.
After the raw count is computed for all submission, the relative accuracy can be computed:
*Sum(weight*accurate_issues) / Sum(weight*all_issues)*
Weight is a configurable parameter. In this case there is only one type of incorrect. So the weights will cancel one another. However, in the future, more types of evaluation types may be added and that is why the algorithm must take into account weight. Possible types of evaluations for incorrect comments could be:

- A little Incorrect - 2 (negative) points
- Very Incorrect - 3 (negative) points
- OMG incorrect - 10 (negative) points

Corner case
In the case when the denominator is 0, that is when the sum of all issues found is 0, either because no issues are found or all issues are comments, then the accuracy is considered to be 1, the maximum. This is because some submissions may not have any mistakes.
In this case the accuracy is considered maximum.

Eligibility points calculation

After individual metrics are computed, the eligibility points distribution can be calculated.
If the reviewer completed a secondary review, his/her eligibility points are updated with a number of points proportional with his performance for the review. The performance is evaluated using the formulas presented in this specification.
For this implementation, the eligibilityPointsPool the following normalization formula is applied.
Let coeff1 be the evaluation coefficient of the first reviewer, for which the eligibility points allocation is calculated
Let coeff2 be the evaluation coefficient of the second reviewer.
The formula is:
[coeff1 / (coeff1 + coeff2)] * eligibilityPointsPool
This is a normalization step, making sure that no more than eligibilityPointsPool points are awarded in total.

Exception

Exceptions thrown from the manager should extend the exception from the interface diagram and should use BaseException defined classes.

Logging

The exceptions should be logged with ERROR level in the action classes. And user request information should be logged with DEBUG level in the action classes.

Required Algorithms

None

Example of the Software Usage

This component will implement the review statistics collection functionality of the upgrade application.

Future Component Direction

None.

Enhancement policy

To avoid bloated enhancements that will slow down development without adding much value to the application, an enhancement policy will be used for this component.
Any major enhancement should be approved by the architect before being implemented. Not getting an enhancement approved is a risk of not only not getting the enhancement considered for review scoring but potentially being asked to remove it in final fix, should the given designer win the competition.
Enhancement approvals can be obtained by contacting the architect directly using the "[Send a message]" button from the member's profile page. Please note that using Contact Managers button from Online review will not also contact the architect.
Any enhancement already mentioned in future direction is considered approved.

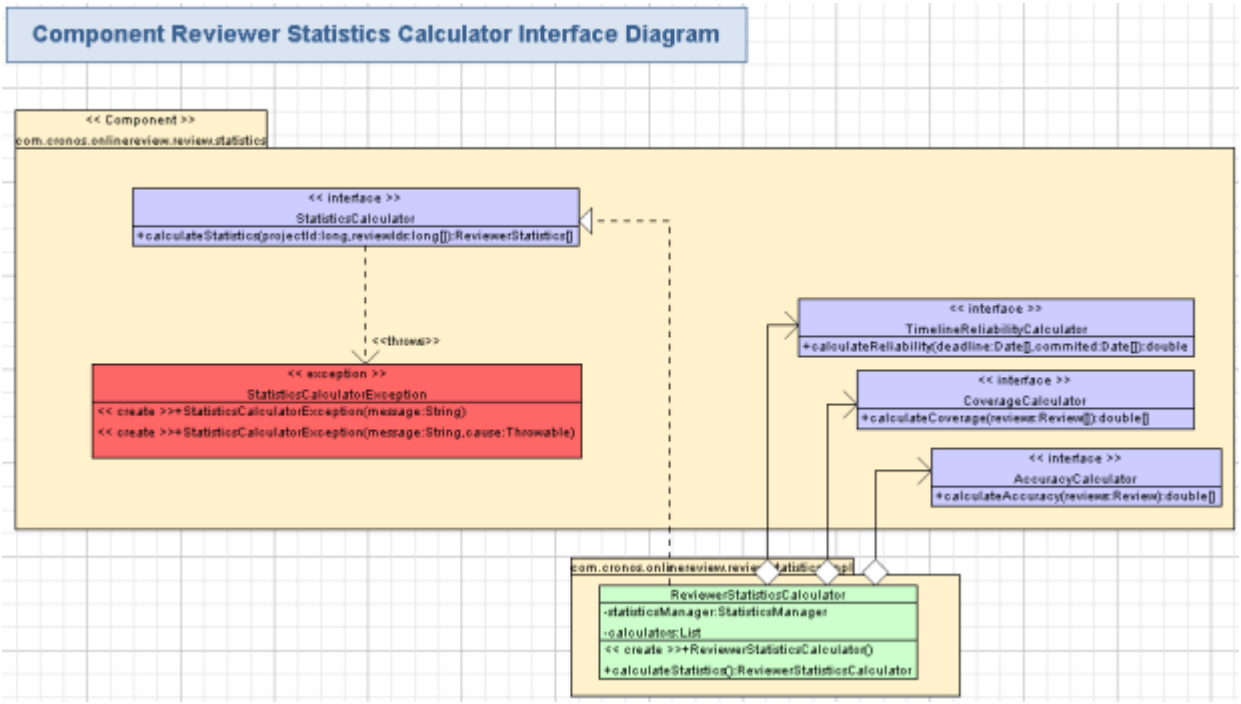
Interface Requirements

Graphical User Interface Requirements

None.

External Interfaces

Designs must adhere to the interface diagram definition found in the architecture TCUML file provided. Changes to the interfaces should be approved in the forum.



Environment Requirements

- Development language: Java1.5
- Compile target: Java1.5

Package Structure

com.cronos.onlinereview.review.statistics
com.cronos.onlinereview.review.statistics.impl

Software Requirements

Administration Requirements

What elements of the application need to be configurable?

- ProjectManager
- ReviewManager
- Any other persistence manager subject to design decisions

Technical Constraints

Are there particular frameworks or standards that are required?

None

TopCoder Software Component Dependencies:

- Base Exception 2.0
- Project management 1.2
- Configuration API 1.0

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

Third Party Component, Library, or Product Dependencies:

None

QA Environment:

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

Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

Required Documentation

Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

Help / User Documentation

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