Software Documentation : Java Custom Late Deliverables Tracker 1.2

This page last changed on Mar 10, 2011 by volodymyrk.

1. Scope

1.1 Overview

Late Deliverables Tracker makes use of the API defined by the Project Management, Phase Management and Deliverable Management components to watch the active projects and make records when the deliverables are late in the Online Review (e.g. review scorecard, final fixes etc.). The projects will be examined periodically.

The version 1.1 enhances the component to track by how much time each deliverable is late. It also accounts for the timeline changes due to the prematurely ended phases.

The version 1.2 enhances the component to regularly email the project managers about unresponded late deliverables which are waiting for the resolution. It also fixes the inconsistencies in how the real deadlines vs compensated deadlines are handled.
The designer of the version upgrade should also update the design with all the changes done to the code of the previous version.

1.2 Logic Requirements

1.2.1 "Track Late Deliverables" property

Late Deliverables Tracker will only examine the projects that satisfy the following conditions:

1. The project is active (i.e. has "Active" status)
2. There is at least one late phase.
3. The "Track Late Deliverables" property has value "true" in the project extended properties.

1.2.2 Time Interval

Late Deliverables Tracker will examine the projects at a configurable interval. The default value will be 5 minutes. It should run at every interval. Designer should use the TopCoder Job Scheduling component to achieve this purpose.

1.2.3 Saving records

A deliverable is considered late when it is not yet completed and the respective phase is late against the "compensated deadline".
The "compensated deadline" is the deadline that accounts for the dependency phases that ended prematurely. To explain how it works let's consider a typical example with the Appeals and Appeals Response phases. The default duration for the Appeals phase is 24 hours but the phase often completes early when all the submitters indicate that they finished their appeals. In that case the Appeals Response starts early as well and that effectively moves the deadline of the Appeals Response back in the timeline. Since the Appeals Response is usually short (e.g. 12 hours) the reviewer may easily miss the new deadline due to being offline when the Appeals phase ended.

The "compensated deadline" is computed as the maximum of the current deadline (project_phase.scheduled_end_time column) and the deadline that would have been if all the dependency phases had ended as scheduled. The latter is computed as the "compensated start time" plus the phase duration (project_phase.duration column), where the "compensated start time" is the phase start time that would have been if all the dependency phases had ended as scheduled. The "compensated start time" can be computed as the maximum of the phase's fixed start time (project_phase.fixed_start_time column) and the end times of the dependency phases that would have been if all of them had ended as scheduled.

So, the algorithm to compute the "compensated deadline" for the phase A is basically the following:

1. Find all the dependency phases for the phase A such that A's start depends on the end of these phases.
2. Iterate through the found dependency phases and compute the scheduled end time plus time lag for each as project_phase.actual_start_time + project_phase.duration + phase_dependency.lag_time
3. Find the maximum of the values computed on step 2 and the A's fixed start time (project_phase.fixed_start_time column).
4. Add A's duration to the value computed on step 3.
5. Find the maximum of the value computed on step 4 and the A's scheduled end time (project_phase.scheduled_end_time column).

The "compensated deadline" is only used for the short phases whose duration is less than a configurable threshold (24 hours by default). So, if the Appeals Response phase in the example above was 24 hours long instead of 12 hours long the lateness would be checked against the "real" scheduled end time of the phase.

When a late deliverable is found a record is added to the tcs_catalog.late_deliverable table. The following information needs to be saved:

1. Project phase ID
2. Resource ID
3. Deliverable ID
4. Deadline. This should be the "real" scheduled end time of the phase, not the "compensated" one.
5. Compensated deadline. Null if there's no compensated deadline.
6. Timestamp when the record was added. The version 1.0 used System.currentTimeMillis() for getting the current time. This should be changed so that the DB's current time is used instead.
7. "Forgive" flag, should be 0 by default.
8. Timestamp when the record was last notified (emailed) about the lateness. The version 1.0 used System.currentTimeMillis() for getting the current time. This should be changed so that the DB's current time is used instead.
9. Delay, which is the number of seconds the deliverable is late for. This should be based on the compensated deadline if present.
10. Explanation of the late resource. Null by default.
11. Explanation date. Null by default.
12. Response to the resource's explanation. Null by default.
13. Response author. Null by default.
14. Response date. Null by default.

The list of deliverable IDs to be tracked will be configurable.

The component should not create records if the particular lateness has already been logged. It is also possible that there is more than one record for the same project phase ID, resource ID and deliverable ID. This can happen, for example, when the deadline was extended after the resource had been late but then the resource is late again for the extended deadline. To meet the two requirements above the following logic should be applied when a late deliverable is detected:

1. If there is not yet a record for the project phase ID, resource ID and deliverable ID add one.
2. If there's already at least one record for the project phase ID, resource ID and deliverable ID find the one with the largest deadline value. If this deadline value is greater or equal than the current deadline then don't add the record. Otherwise add a new record with the new deadline. Please note that only the real deadlines are compared here, not the compensated ones.

If the record already exists the component should update the "delay" field of the existing record. The new value should be computed as the current time (the DB time not the JVM time) minus the record's deadline (the compensated one if present). The value should be in seconds. Please note that only the record with the current deadline is to be updated, so if there are multiple records for the project phase ID, resource ID and deliverable ID no more than one of them is to be updated.

1.2.4 Warning emails

When a record is added for the late deliverable a warning email should be sent to the corresponding member. The content of the email will be created based on a configurable email template with the following fields:

1. Late Deliverable ID

2. Project Name.
3. Project Version
4. Project ID
5. Phase Name
6. Deliverable Name
7. Deadline
8. Delay in "[X day[s]][Y hour[s]][Z minute[s]]" format.
9. Compensated deadline.
10. Explanation deadline, which is the record's creation date plus a configurable number of hours.

The emails should be sent in the HTML format.

The template should support the "if" construction for the compensated deadline not being equal to the real deadline. This will be used to optionally add a sentence to the email saying that the deadline was compensated due to the dependency phases having ended prematurely. This sentence will only appear if the deadline was indeed compensated, i.e. if the compensated deadline is not equal to the real deadline. Naturally, if the phase has the duration large enough for the compensated deadline not to be used (see section 1.2.3), the compensated deadline should be equal to the real deadline.

The template should also support the "if" construction for whether the late member can still submit the explanation. The late member can submit the explanation if it has not been submitted yet (i.e. the explanation field is null) and if the explanation deadline has not been reached yet.

The email template for each deliverable ID should be configurable (but it should be possible to set a common email template for all deliverable IDs). It should also be possible to turn off email notifications for specific deliverable IDs. Please note that if a deliverable ID is not being tracked (see 1.2.3) the warning email for it won't be sent anyway.

1.2.5 PM emails

The component will regularly monitor all explained and unresponded late deliverable records and send notification emails to the project managers with a reminder to resolve the records. The Late Deliverables Management component will be used to search for the outstanding records.

The component will search for the outstanding records and send emails each X hours starting from the midnight, where X is configurable. For example, if X is equal to 6 hours the email will be sent at 00:00, 06:00, 12:00 and 18:00 each day.

The component will perform as following:

1. Construct search filters for the "explained" status being true and the "responded" status being false and compose the filters with the AND filter.
2. Use the filter from the step 1 with the Late Deliverables Management component to retrieve all outstanding records.
3. For each record retrieve the list of resources for the associated project. The list of the resource roles to be retrieved will be configurable (and will typically be Manager and Copilot). Use Resource Management and Resource Management Persistence components for that.
4. For each unique user from the list of resources collect the list of outstanding late deliverable records. Please note that resources and users are not the same entities, a user can have multiple resources.
5. For each unique user send a email notification with the list of the user's outstanding records.

The content of the email will be created based on a configurable email template with the following fields for each late deliverable record:

1. Late Deliverable ID
2. Project Name.
3. Project Version
4. Project ID
5. Phase Name
6. Deliverable Name
7. Deadline
8. Delay in "[X day[s]][Y hour[s]][Z minute[s]]" format.
9. Compensated deadline.

Loop construct from the Document Generator component should be used to handle the list of late deliverable records.

The emails should be sent in the HTML format.

1.2.6 Command Line

The component should provide a command line interface.

The PM emails should be sent by the same command line utility that tracks late deliverables. No need to create a separate command line utility for that.

1.2.7 Thread-Safety

The component is not required to be thread-safe (if otherwise is not required by the Job Scheduling component).

Special care should be taken though to make sure component works if the time to process all projects is larger than the scheduled period (i.e. when the jobs can overlap in time).

1.2.8 Performance

Special care should be taken to make sure the component is fast:

1. Instead of retrieving all deliverables and then filtering by their IDs it is better to filter by the deliverable IDs in the first place (see DeliverableFilterBuilder#createDeliverableIdFilter method)
2. Instead of retrieving deliverables for all phases and then choosing only those that are for late phases it is better to filter by the late phase IDs in the first place (see DeliverableFilterBuilder#createPhaseIdFilter method)
3. Instead of retrieving all deliverables and then filtering by their status (i.e. completed vs not completed) it is better to retrieve not completed deliverables in the first place (see *complete* parameter of the DeliverableManager's methods).
4. Instead of retrieving late deliverables for each project it is better to retrieve all late deliverables for all projects in a single call. This can be done by pre-collecting the list of the late phase IDs for all projects which are to be processed and then making a **single** call to DeliverableManagement for all projects instead of one call per project.

Also, make sure that the "compensated deadline" from section 1.2.3 is computed only when needed, i.e. when the phase duration is less than the configurable threshold.

1.3 Required Algorithms

Designers need to describe the following algorithms:

1. Algorithm to retrieve the list of projects to be inspected.
2. Algorithm to get all late deliverables for a project.
3. Algorithm to decide if the record needs to be added.
4. Algorithm for preparing PM emails.

1.4 Example of the Software Usage

The component will be used to regularly monitor all active contests in the Online Review and track who is late. The collected information will later be used to apply penalties for the members who are constantly late.

1.6 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 interface and command line interface will be provided.

2.1.2 External Interfaces

None.

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.management.deliverable.latetracker

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Time interval the jobs are scheduled at.
- Warning email templates for each deliverable type ID. The templates should be read form text files.
- Deliverable IDs to track.
- Deliverable IDs to send warning emails for.
- The maximum duration of the phases for which the "compensated deadline" is to be used.
- Time interval between the late record creation date and the explanation deadline (in hours).
- Time interval the PM emails to be sent at.
- The list of resource roles to send the PM emails to.
- PM email template.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

- Base Exception 2.0
- Configuration API 1.0
- Configuration Persistence 1.0.2
- Email Engine 3.2.0
- Document Generator 3.1.1
- Project Management 1.0.1
- Project Management Persistence 1.1.2
- Phase Management 1.0.4
- Phase Management Persistence 1.0.2
- Deliverable Management 1.1.1
- Deliverable Management Persistence 1.1.2
- Online Review Deliverables 1.0.2
- Job Scheduling 3.2.0
- Search Builder 1.3.1
- Command Line Utility 1.0.0

- DB Connection Factory 1.1.0
- Object Factory 2.0.1
- Late Deliverable Management 1.0.4
- Resource Management 1.1.1
- Resource Management Persistence 1.2.2

**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

3.4.2 Help / User Documentation

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