Phase Management Persistence 1.1 Requirements Specification

1. Scope

1.1 Overview

Project Phases defines the logic structure of the phase dependencies in a project. Phase Management builds an execution layer and defines a DAO interface. This component provides DAO Implementations for the Phase Management component.

The version 1.0 DAO implementation manages its own transaction. Every DAO call commits any changes made and closes the database connection.

Version 1.1 of this component must provide a DAO implementation capable of running inside a transaction that is managed externally, possibly by a J2EE container. This new version must not break backward compatibility and both new and old implementations must be interchangeable in terms of the database schema.

1.2 Logic Requirements

1.2.1 Phase Operations

The following operations should be supported.

1.2.1.1 Create Phases

Phases can be created. The identifiers of any new phases will be provided using the ID Generator. Existing phases should continue to use the original identifiers. Other phases for the project not specified in the input should be deleted.
The operator needs to be provided for auditing purpose.

1.2.1.2 Get Project Phases

Project phases can be retrieved by specified project ID or a list of project ID's.

1.2.1.3 All Phase Types

There should be a way to retrieve all phase types in the system.

1.2.1.4 All Phase Statuses

There should be a way to retrieve all phase statuses in the system.

1.2.1.5 Update Phases

Phases can be updated.
The operator needs to be provided for auditing purpose.

1.2.2 Informix Persistence

An Informix plug-in implementation will be developed. The SQL scripts will be provided.

1.2.3 Unmanaged Transaction Persistence

For Version 1.1, another Informix plug-in must be added. This one must not manage its own transaction.

1.2.3.1 It must get the connection in every method and never store it in an instance variable.

1.2.3.2 It must never call close(), commit() or rollback() on the connection.

1.2.3.3 It must throw a RuntimeException (or subclass) in order to rollback the transaction.

1.2.3.4 It must comply with EJB development restrictions

(http://java.sun.com/blueprints/qanda/ejb_tier/restrictions.html).

1.2.4 Backward compatibility

In terms of external use, version 1.1 must not break backward compatibility. The existing DAO implementation must behave exactly the same.

1.3 Required Algorithms

No specific algorithms are required.

1.4 Example of the Software Usage

A project management application can use the component to provide the persistence of the project phases.
J2EE applications will be able to use the Project Phases and the Phase Management components from an EJB Session Bean. Persistence must be configured to use the new persistence implementation. The J2EE application server will manage the JTA transaction.

1.5 Future Component Direction

None.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Design must adhere to the PhasePersistence interface definition from the Phase Management Component. Designer can choose to add more methods to the classes, but must keep the ones defined as a minimum.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4

2.1.4 Package Structure

com.topcoder.management.phase.db

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Database connection

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

JDBC

3.2.2 TopCoder Software Component Dependencies:

- Project Phases
- Phase Management
- Configuration Manager
- DB Connection Factory
- ID Generator

**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:*

None.

3.2.4 *QA Environment:*

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003
- Informix 10.0

3.3 Design Constraints

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

3.3.1 *Database Connections*

Database connections must not be cached within the component. Connections should be created for each operation and closed afterwards.

3.3.2 *Component Scalability*

The component needs to be scalable. Running multiple instances in the same JVM or in multiple JVM's concurrently should not cause any problem.
Version 1.1 implementation of added plug-in should not use thread synchronization primitives to synchronize access with other enterprise bean instances (see EJB restrictions).

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 Poseidon.