Phase Management Requirements Specification

1. Scope

1.1 Overview

Project Phases defines the logic structure of the phase dependencies in a project. This component builds a persistence and execution layer. Phases can be started, ended or cancelled. The logic to check the feasibility of the status change as well as to move the status will be pluggable. Applications can provide the plug-ins on a per phase type/operation basis if extra logic needs to be integrated.

1.2 Logic Requirements

1.2.1 Phase Operations

The following operations should be supported.

1.2.1.1 Update Project Phases

Project phases can be updated. The identifiers of any new phases will be provided with ID Generator. Existing phase should continue to use the original identifiers. Other phases for the project not specified in the input should be deleted. Project phases should be validated according to the defined rules.

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 Start Phase

A phase can be started. The operation will set the phase to the Open status and also persist the actual start timestamp. There should also be a method to query whether a phase can be started.

The operator needs to be provided for auditing purpose.

1.2.1.6 End Phase

A phase can be ended. The operation will set the phase to the Closed status and also persist the actual end timestamp. There should also be a method to query whether a phase can be ended.

The operator needs to be provided for auditing purpose.

1.2.1.7 Cancel Phase

A phase can be canceled. The operation will set the phase to the Closed status and also persist the actual end timestamp. There should also be a method to query whether a phase can be canceled.

The operator needs to be provided for auditing purpose.

1.2.2 Phase Handler

Optional pluggable phase handling mechanism can be configured per phase type/operation. The handler will provide the decision of whether the start, end or cancel operations can be performed

as well as extra logic when the phase is starting, ending or canceling. Notice that the status and timestamp persistence is still handled by the component.

The component should come with an implementation that handles the phase start query based on the dependencies.

1.2.3 Persistence

Persistence needs to be pluggable. For this release an Informix plug-in will be developed. The SQL scripts will be provided.

1.2.3.1 Persistence Implementation

The persistence implementation needs to be designed in this component, but will be separated into a second development project. Please put all persistence implementation related information into a separate sub-package and clearly mark the responsibilities of the two development projects.

1.3 Required Algorithms

The algorithm to determine if a phase can be started should be provided by the designer.

1.4 Example of the Software Usage

A project management application can use the component to provide the persistence of the project phases. User will be able to execute the phases from the web interface. Plug-ins need to be developed if extra logic is necessary.

1.5 Future Component Direction

Phase automation mechanism can be provided so that phases are moved once the criteria are met.

2. Interface Requirements

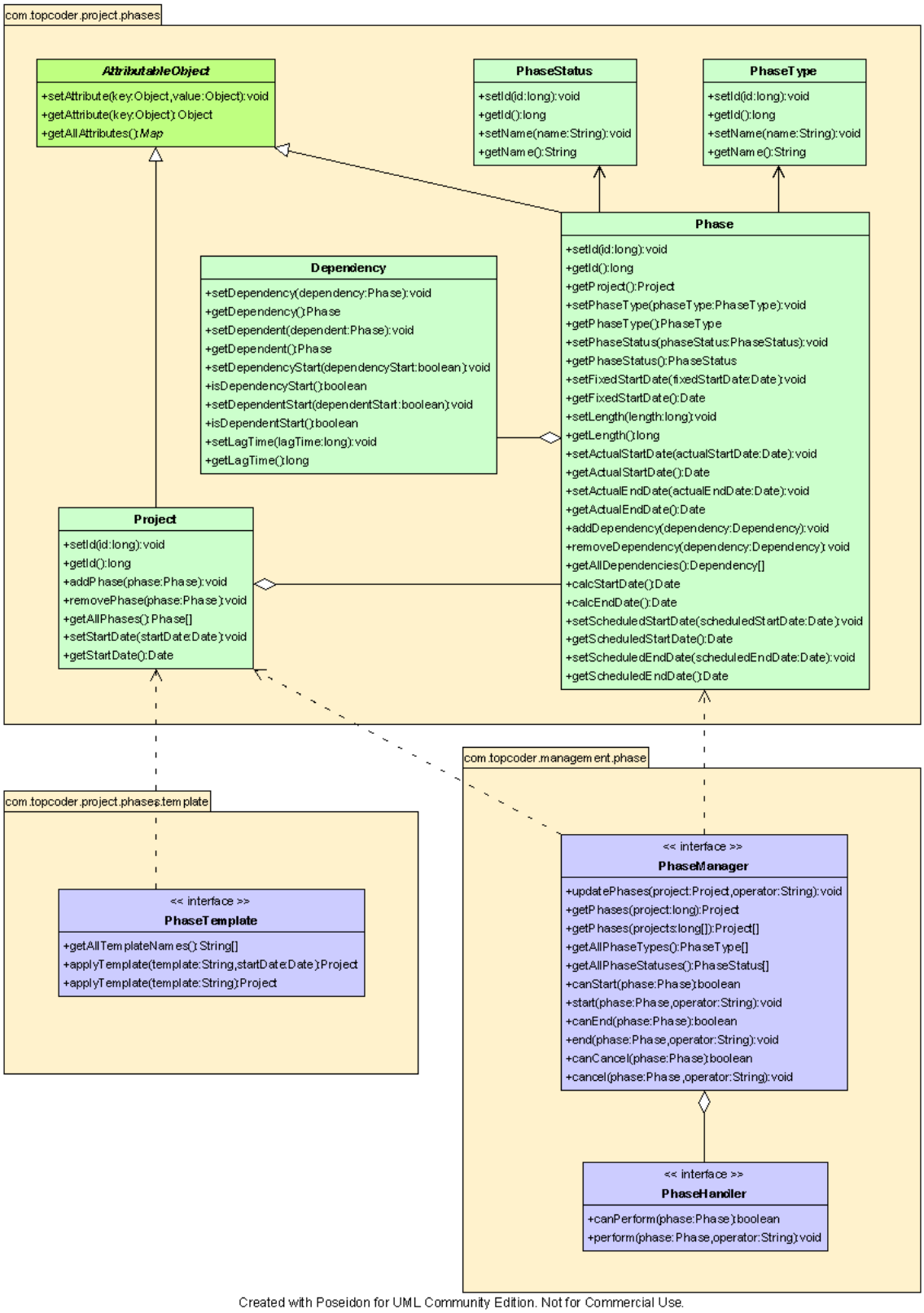
2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Design must adhere to the interface diagram definition. Designer can choose to add more methods to the classes/interfaces, but must keep the ones defined on the diagram as a minimum. Source files can be found in the distribution.

Phase Management Interface Diagram



2.1.3 Environment Requirements

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

2.1.4 Package Structure

com.topcoder.management.phase

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Database connection
- Phase handlers

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
- 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.

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.