Project Phases 2.0 Requirements Specification

1. Scope

1.1 Overview

The Project Phases component will allow an application to define a project, its phases and all its dependencies. Each phase will be defined by a start date, length of time and any other custom attributes necessary. This will enable the application to automatically adjust the timeline of dependent phases.

This enhancement of the component will provide a more flexible dependency specification. Besides the phase structure will incorporate the execution status and actual execution timestamps.

1.2 Logic Requirements

1.2.1 Compatibility

Compatibility with version 1.0 is not required.

1.2.2 Project

In addition to the attributes from the previous version, each project should be associated with a numeric identifier.

1.2.3 Phase Type

A phase type consists of a numeric identifier and a name.

1.2.4 Phase Status

A phase status consists of a numeric identifier and a name. The component has three built in phases – Scheduled, Open and Closed.

1.2.5 Phase

In addition to the attributes from the previous version, the following additions/modifications should be made:

- Add a numeric identifier
- Add a phase type
- Add a phase status
- Phase length should be accurate to milliseconds
- Fixed start time would be interpreted as “start no early than”. Fixed start time is optional.
- Add scheduled start timestamp, scheduled end timestamp, actual start timestamp and actual end timestamp.

1.2.6 Phase Dependency

Phase dependency should be able to specify whether the dependent phase will start/end after the dependency phase starts/ends with a lag time. The lag time should be accurate to milliseconds.

There should be a way to validate the phase dependencies. If a cyclic dependency is detected, an error should be indicated.

1.2.7 Phase Calculation

If the actual start time or end time is present, the dependent phases should refer to this timestamp instead of the scheduled start time or end time.

1.3 Required Algorithms

The algorithm to detect cyclic dependency and the algorithm to calculate project phases must be provided by the designer.

1.4 Example of the Software Usage

A project management application could use this component to see how adjusting various phases of a project affect other phases and the overall timeline of a project.

1.5 Future Component Direction

A pluggable persistence layer will be added in the future to support the ability of loading and storing a project's phase data. A template mechanism can generate phases for similar project structure.

2. Interface Requirements

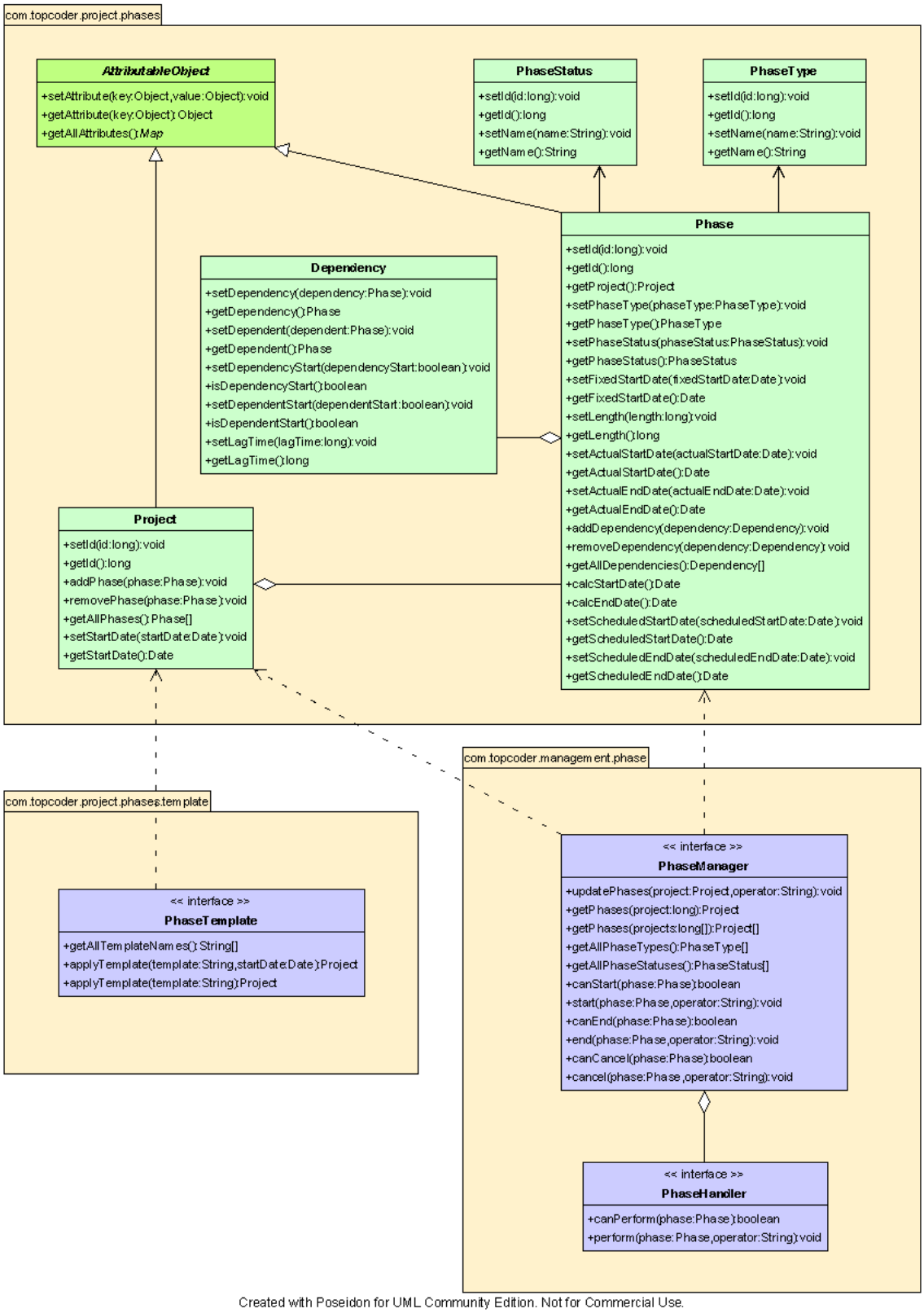
2.1.1 Graphical User Interface Requirements

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.project.phases

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

None.

**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

3.3 Design Constraints

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

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.