Object Factory 2.0 Requirements Specification

1. Scope

1.1 Overview

The Object Factory component provides a generic infrastructure for dynamic object creation at run-time. It provides a standard interface to create objects based on configuration settings or some other specifications. Using an object factory facilitates designing a broader solution by allowing the specific details of the instantiated class to be designed at a later time.

1.2 Logic Requirements

1.2.1 Dynamic Object Creation

The component will provide the API to dynamically create objects for any defined interface at run-time based on some specification.

1.2.2 Dynamic Object Specification

At minimum, the design should include specifying the object using Configuration Manager. The design should not enforce the use of Configuration Manager when another form of object definition is required.

1.2.2.1 Pluggable Specification

The mechanism to specify the object will be pluggable.

1.2.2.2 Specification Parameters

In addition to the object type, the specification for creating the object should be flexible enough to include the following parameters:

- The JAR that the object type is defined in
- The list of constructor arguments that are needed to create the object
- Any other useful options that are appropriate

1.2.3 Constructor Arguments

The existing 1.0 design requires a simple no-argument constructor or the using component to pass constructor arguments to utilize an argument constructor. This makes it inconvenient to the using component because it now needs to know how the object is created.

For example, when the using component calls the createObject() method with no constructor parameters, it will be possible to define the arguments in the configuration file. The arguments will be loaded from the configuration file and passed to the object's constructor.

1.2.4 Multiple Instances

Sometimes a using component may need to create multiple different instances of the same object type. The design will allow the using component to optionally specify the "instance identifier", in addition to the object type. The factory will match both the object type and instance identifier provided in the specification, in order to determine the exact constructor arguments to instantiate the object.

 ©TopCoder Software, Inc. 2006

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

Many Java components need to create a pluggable interface implementation from configuration settings. Different designers have come up with different ways to specify these settings in the past. Using Object Factory will standardize this common task, thus improving consistency and quality of the components in this area.

1.5 Future Component Direction

Networked implementations will be able to take advantage of network technologies to enable a remote object factory.

2. Interface Requirements

2.1.1 Graphical User Interface Requirement

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java 1.4
- Compile target: Java 1.4, Java 1.5

2.1.4 Package Structure

com.topcoder.util.objectfactory

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:

- Configuration Manager

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

©TopCoder Software, Inc. 2006

3.2.3 Third Party Component, Library, or Product Dependencies:

Informix Database.

3.2.4 QA Environment:

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

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 Poseidon.

©TopCoder Software, Inc. 2006