Requirements Specification

1. Scope

1.1 Overview

In order to handle exceptions and errors in a unified manner, a generic exception class is needed. Error processing logic is simplified since an application can throw one Throwable object wrapping several error types. The Base Exception component follows the chained exception paradigm.

1.2 Logic Requirements

1.2.1 Support chain exception paradigm

- Supports chained exception paradigm (currently only available in Java 1.4)
- Chained exception stack trace is included in the stack trace print outs.
- Chained exception messages are included when retrieving the message of an exception.
- The chained exception is settable and retrievable.

1.2.2 Provide base Error class for custom errors to inherit from

1.2.3 Provide base Exception class for custom exceptions to inherit from

1.2.4 Provide base Throwable class for both Error and Exception classes (if necessary)

1.3 Example of the Software Usage

TopCoder components throw custom exceptions inherited from the same base class to simplify error handling.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

- None.

2.1.2 External Interfaces

- None.

2.1.3 Environment Requirements

- Development language: Java 1.4
- Compile targets: Java 1.3, Java 1.4

2.1.4 Package Structure

- com.topcoder.util.errorhandling

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

3.3 Design Constraints

3.3.1 Development Standards:

- It is required that all code be clearly commented.
- All code must adhere to javadoc standards, and, at minimum, include the @author, @param, @return, @throws and @version tags.
  - When populating the author tag, use your TopCoder member handle. In addition, please do not put in your email addresses.
  - Copyright tag: Copyright © 2003, TopCoder, Inc. All rights reserved
- For standardization purposes, code must use a 4-space (not tab) indentation.
- Do not use the ConfigurationManager inside of EJB's. The usage of the java.io.* package to read/write configuration files can potentially conflict with a restrictive security scheme inside the EJB container.
- All code must adhere to the Code Conventions outlined in the following: http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

3.3.2 Database Standards:

- Table and column names should be as follows: table_name, not tableName.
- Every table must have a Primary Key.
- Always use Long (or database equivalent) for the Primary Key.
  For maximum portability, database objects (table names, indexes, etc) should be kept to 18 characters or less. Informix and DB2 have an 18-character limit.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification
- Test Cases

3.4.2 Help / User Documentation

- Javadocs must provide sufficient information regarding component design and usage.