Member Guide

Software Versions

These are the current software versions used in the design, development and testing of TopCoder Software applications and components.

- ANT version: 1.6.2
- NANT version: 0.84
- Poseidon version: 3.1
- JUnit version: 3.8.1 or greater
- NUnit version: 2.1

Documentation

Design Documents

- Poseidon diagrams must clearly define intended component usage in the 'Documentation' tab of the class diagrams.
- All 'Documentation' tabs for methods must detail both valid and invalid argument inputs. This information will be used to facilitate Unit tests, Accuracy tests and Failure tests.
- All component documents should begin with the component name.
  (i.e. Configuration_Manager_Requirements_Specification.doc)
- Document names should contain the document type.
  (i.e. Configuration_Manager_Component_Specification.doc)
  Existing document types:
  - Component_Specification
  - Class_Diagram
  - Use_Case_Diagram
  - Sequence_Diagram
- For multiple documents of the same type, use the naming convention described above with a differentiator for each file.
  (i.e. Configuration_Manager_Sequence_Diagram_1 OR Configuration_Manager_Sequence_Diagram-Add_Namespace)

Implementation Source Code

- Implementations of the design must clearly translate the design documentation into the corresponding language format (i.e. java -> javadoc, .NET -> XML).
- Source code documentation must be sufficient to enable component usage without additional documentation.

Coding Standards

Java

- Line length should not exceed 120 characters.
- All code must adhere to javadoc standards, and, at minimum, include the @author, @param, @return, @throws and @version tags.
  - When populating the author tag, please use TCSDEVELOPER for your initial submission, for final fixes use your TopCoder Handle. Do not include any personal information such as email address.
  - Copyright tag: Copyright © 2005, TopCoder, Inc. All rights reserved
- For standardization purposes, code must use 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.
- Except where overridden above, all code must adhere to the Code Conventions outlined in the following: http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

.NET

- Line length should not exceed 120 characters.
- All code must adhere to the C# documentation compiler tags (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/csref/html/vclrftagsfordocumentationcomments.asp), and, at minimum, include the following tags; <param>, <summary>, <returns>, and the <value> tag where necessary.
    - When populating the author tag, please use your TopCoder Handle. Do not include any personal information such as email address.
    - Copyright tag: Copyright © 2005, TopCoder, Inc. All rights reserved
- For standardization purposes, code must use 4-space (not tab) indentation.
- Except where overridden above, all code must adhere to the Code Conventions outlined in the following: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsent7/html/vxconcodingtechniques.asp

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.

Component Project Deliverables

Design

Initial Submission Date

Architects must submit a completed design by the initial submission date; any designs submitted after will not be reviewed. The following items are required as part of the component design:

Software Architecture Documents

- Class Diagrams*
  - Class definitions must include detail for the following:
    - Package
    - Scope
    - Inheritance
    - Interfaces
    - Constructors
    - Modifiers
    - Use the 'Documentation' tab in Poseidon to provide a description of class usage. This documentation may include a simple example.**
  - Variables definition must include:
    - Scope
    - Type
    - Modifiers
    - Use the 'Documentation' tab in Poseidon to provide a description of the variable and its initial value. Any public variables must be defined. It is at the discretion of the designer to detail private variables. However, some designs may require the definition of private variables if they are critical to the component implementation and requirements.**
  - Method definition must include:
    - Scope
    - Return Type
    - Argument type and order
    - Exceptions
    - Modifiers
    - Use the 'Documentation' tab in Poseidon to provide:
      - Method description
      - Detail arguments, exceptions and return types
      - Detail error and invalid argument handling.**
      - List the range of valid and invalid inputs for testing.
  - Associations between classes are well defined.
- Use-Case Diagrams*
  - Use Case diagrams must contain one or more Use Cases to outline the functional usage of the component as outlined in the Requirements Specification.
  - NOTE: A Use Case is represented as a "bubble" in the Use Case diagram. Similar functionality, such as administrative methods, can be aggregated into a single Use Case.
- Sequence Diagrams*
  - Sequence Diagrams must demonstrate the implementation of each Use Case.
  - One or more Sequence Diagrams must exist for each Use Case.
  - Provide descriptive names for each sequence diagram. Names should also correlate the Sequence Diagram to the corresponding Use Case Diagram.
    For example: "Sequence_Diagram-Create_User", "Sequence_Diagram-Delete User".
  - Sequence Diagram names should also appear in a text box in the top left corner of

the sequence diagram.

- Component Specification Document
  - Outlines additional details required to develop a solution. This document will follow TopCoder Software's Component Specification Template.

Note: The thoroughness of the required diagrams and documents weighs heavily in the review process.

* All diagrams must be created using Poseidon Community Edition. A free download of Poseidon can be found at: http://www.gentleware.com/products.

Design Submission

- Design submissions must be uploaded to the Online Review application (http://software.topcoder.com/review) by the specified deadline as specified on the project-posting page on www.topcoder.com.
- Java Submissions
  - Execute 'ant design_submission' to package the required elements of your solution.
  - The internal structure of the jar should be as follows:
    - /build.xml
    - /docs
    - /src/java/main/com/topcoder….
    - /src/java/tests/com/topcoder….
    - /log
    - /test_files
- C# Submissions
  - Execute 'nant design_submission' to package the required elements of your solution.
  - The internal structure of the jar should be as follows:
    - /docs
    - /src/csharp/main/com/topcoder….
    - /src/csharp/tests/com/topcoder….
    - /log
    - /test_files

Final Submission Date

The Final Submission Date serves to provide time for the winning architect to apply design recommendations made by the design review board. This phase will not always be required and is at the discretion of TopCoder Software and the design review board. Specific deliverables will be defined when the winning design is announced.

Failing to meet these deliverables by the set deliverable date will result in disqualification. At this time, the designer with the next highest score will be deemed as the winner.

Development Support

Winning Designers cannot submit development solutions for their component design during the development phase.

The winning designer must provide design support during the development project. Interaction with developers must occur within the TopCoder Software forums. This is necessary to ensure that all developers receive the same information from the designer. In addition, it maintains a log of all discussions in case of future discrepancies.

At the end of the development phase, the designer must submit any updated design documents to the Product Manager. Only changes that are accepted by the architect review board or that are clarifications

of the existing design may be included. Changes that modify the design or enhance the functionality without approval from the architect board cannot be incorporated into the design.

Post Development Review

After the component has completed the development process, the designer and developer will be notified that the component is available for download on the software.topcoder.com website. It is the responsibility of both the designer and developer to ensure that the distribution package contains the correct design documents and source code, and that all documents are synchronized. In addition, designers should validate that the documents available for download are the most recent versions of the design.

This phase is an opportunity for members to validate the quality of their component. Once complete, the handles of all members who worked on this component will be displayed on the site.

Development

Initial Submission Date

Developers must submit a solution for the posted component design by the initial submission date; any solutions submitted after will not be reviewed. The following items are required as part of the component solution:

Completed Solution

- A working solution that meets the required functionality, uses the required technologies and adheres to the component design.
  - All source code (component code AND test case code) must adhere to the TopCoder Software Coding Standards.
  - The solution must implement the design that was distributed for this component. Any deviations from items that are clearly defined in the design will result in loss of points during the review phase. To request a modification to a design, please post your comments in the corresponding component Developer Forum. If the change is applied, the updated documents will be posted to the forum.

Test Cases

- Test Case Framework
  - Java components: J-Unit
  - .NET components: N-Unit
- Unit test cases written in the corresponding framework must exist to thoroughly test the implementation. This includes, but is not limited:
  - Checking methods for handling of valid and invalid arguments.
  - Testing constructors for accessibility.
  - Testing expected output.
- Unit tests cases are to be placed in the package test directory.
- Java Components: /src/java/tests/PACKAGENAME/UnitTests.java must be modified to call the individual tests written by the developer.
- DO NOT modify or add tests to any other test directories. These directories will not be packaged with your submission.
- Each test case should use setup() and teardown() methods to properly configure the environment for testing. Any files or folders created by test cases must be removed after the tests have executed.
- Additional files or folders required for testing must be placed in the /test_files directory.
  - All source code must follow the TopCoder Software Coding Standards. At minimum, a brief overview of each test must be included. This should include what the test is intended to validate.
- Test cases must compile using the build script (Java: 'ant compile_tests', .NET: 'nant compile_tests') distributed by the Product Manager.

Component Specification

- Complete the Component Specification packaged with the development distribution.
- Alter the "Environment Requirements", "Installation and Configuration", "Usage Notes" and "Build Modifications" to comply with the component implementation. These instructions should provide sufficient information for the component to be installed and used in a "clean" environment.
- For components that require additional development, detail how that development needs to occur. An example is defining the proper way to implement interfaces and configure pluggable classes.

Development Submission

- Development submissions must be uploaded to the Online Review application (http://software.topcoder.com/review) by the specified deadline as specified on the project-posting page on www.topcoder.com.
- Execute the 'test' target ('ant test' or 'nant test') to execute test cases and generate the log file in /log. This directory will be packaged with your submission and must contain the log file.
- Execute the 'dev_submission' target ('ant dev_submission' or 'nant dev_submission') to package the required elements of your solution.
- The internal structure of the jar should be as follows:
  - /build.xml
  - /docs
  - /src/java/main/com/topcoder….
  - /src/java/tests/com/topcoder….
  - /log
  - /test_files
- Notes:
  - It is not necessary to submit any compiled code. Once the solution is received, the "test" target will be executed against the solution. If a solution does not compile, and the developer did not include any special instructions the solution will fail the screening phase.

Final Submission Date

The Final Submission Date serves to provide time for the winning developer to modify their implementation to pass review board test cases and to incorporate suggestions made by the review board. This phase will not always be required and is at the discretion of TopCoder Software and the Development Review Board.

Failure to meet these deliverables OR failure to modify the component to pass the required test cases by the Final Submission Date will result in disqualification. At this time, the developer with the next highest score that exceeds the minimum allowable score will be deemed winner.

Post Development Review

After the component has completed the development process, the designer and developer will be notified that the component is available for download on the software.topcoder.com website. It is the responsibility of both the designer and developer to ensure that the distribution package contains the correct design documents and source code, and that all documents are synchronized. In addition, designers should validate that the documents available for download are the most recent versions of the design.
This phase is an opportunity for members to validate the quality of their component. Once complete, the handles of all members who worked on this component will be displayed on the site.

TopCoder Test Framework

Test Types

Unit

Unit tests validate that the component works properly at the method level. Unit tests evaluate every public method in the component to ensure that valid and invalid inputs are handled properly AND as defined in the documentation.

In addition, Unit tests MUST include an implementation of the “demo” as provided in the Component Specification. This will validate that the component operates as intended by the designer.

Accuracy

Accuracy tests validate that given correct input, the component operates as expected. *

Failure

Failure tests ensure incorrect component usage is handled properly. *

Stress

Stress tests serve two purposes:

1. Execute high volume, multi-threaded usage against the component.
2. Obtain a set of performance metrics for the component implementation. (i.e. 500 users, 50 requests/user, average response time: 50ms)

*Failure and Accuracy tests do NOT need to test every argument permutation for each public method in the component. These tests should use the component in end-to-end usage and at minimum must implement the scenarios defined in the sequence diagrams.

Notes

Configuration

Testing components may require connections to databases, telnet servers, ftp servers, etc. Any configuration parameters must be configurable. The best way to achieve this is to set configuration parameters in the build file using the `jvmarg` parameter and retrieving that value through the `System.getProperty` method.

Build.xml:

```
<jvmarg value="-Djditestport=${jdi_port}"/>
```

Test Case Source Code:

```
private String getPort() {
    String port = System.getProperty("jditestport");
    if (port == null) {
        port = "8000";
    }
    return port;
}
```