Testing Framework 1.0 Requirements Specification

1. Scope

1.1 Overview

NUnitAsp (http://nunitasp.sourceforge.net) simplifies the process of testing web applications by enabling the application to be built, deployed, tested, and shut down entirely within an NAnt build script. This component extends this concept to server-driven applications in general. It provides analogous functionality for database servers, web servers (with a wrapper for NUnitAsp tests), and the extensibility to provide the same support for other types of servers.

1.2 Logic Requirements

1.2.1 NUnit Extension

The test framework must be set up as an NUnit (http://nunit.sourceforge.net) extension. The standard NUnit distribution contains example extensions to show how this is done.

1.2.2 NAnt Integration

Custom tasks must be added to NAnt (http://nant.sourceforge.net) to enable tests defined in the test framework to be driven by a NAnt build script. These tasks must contain all of the configuration information necessary to perform their functions. There must be no external configuration. The standard NAnt distribution contains example custom tasks to show how this is done.

1.2.3 Server Initialization

The test framework must provide a mechanism for starting a server if necessary, and initializing it with the proper data. For a SQL database server, this would mean starting the database server, creating the necessary tables and possibly filling them with test data. For the Internet Information Services (IIS) server, this would mean deploying the web application and starting the IIS server. It must be possible to initialize several servers for one NAnt server application testing task.

1.2.4 Running Test Suites

Once the all of the necessary servers have been properly initialized, the framework will run an NUnit test suite.

1.2.5 Server Clean Up

The test framework must provide a mechanism for cleaning any test data off of a server (if desired), and for shutting down the server (also if desired) after the testing has been completed.

1.2.6 Extensible Server Types

Initially, the testing framework must support testing for web and database applications. It must also provide extensibility, allowing for new server types to be defined within the framework.

1.2.7 Consistency

The interfaces for testing different types of servers must be as similar to each other as possible in order to keep the resulting tool consistent and easy to use.

1.2.8 Pluggable Testing Back-ends

Initially, the framework must support the testing of web applications using NUnitAsp and database applications using an analogous custom component (which is a separate component, but must have its interfaces defined by this component). Use of these testing back-ends must be pluggable, to allow for new types of databases and web servers to be used. These pluggable back-ends must be appropriately wrapped, such that NUnitAsp (or any other back-end) could change its interface without necessitating a change in the testing framework. Any newly defined server type must also support pluggable testing back-ends.

In summary, this version will provide a concrete back-end implementation for IIS using NUnitAsp.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

A web application uses the IIS ASP.NET server and SQL Server 2000 database. The server application test framework provides a simple way to specify how to configure and initialize both of these servers. Testing the application is then simply a matter of calling the framework, which initializes the servers, runs a test suite, and cleans up after itself.

1.5 Future Component Direction

Database testing will be added as a separate component. New server types (file server, ftp server, etc) will be added, and existing test back ends will be replaced (NUnitAsp may be upgraded or replaced by some other IIS testing framework).

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Must extend NUnit 2.2.5 and must use NUnitAsp 1.5.1 and NAnt 0.84.

2.1.3 Environment Requirements

- Development language: C# 1.1
- Compile target: C# 1.1
- IIS 6.0
- NAnt 0.84
- NUnit 2.2.5
- NUnitAsp 1.5.1

2.1.4 Package Structure

TopCoder.Test.Framework

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

Database connection should be configurable.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

- .NET Framework 1.1
- NAnt 0.84
- NUnit 2.2.5
- NUnitAsp 1.5.1

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.

3.2.3 Third Party Component, Library, or Product Dependencies:

None.

3.2.4 QA Environment:

- Windows 2000
- Windows Server 2003
- IIS 6.0

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.