Test Harness Architecture & Design

T. Archer, D. Howick, D. Pretola, M. Vats

4/16/2020

Table of Contents

[Table of Contents 2](#_Toc37950777)

[Overview 3](#_Toc37950778)

[Architecture 3](#_Toc37950779)

[Class Diagram 6](#_Toc37950780)

[Architectural Components 7](#_Toc37950781)

[Test Executive 7](#_Toc37950782)

[Requirements Mapping 7](#_Toc37950783)

[Messaging Queues 7](#_Toc37950784)

[Requirements Mapping 8](#_Toc37950785)

[Test Harness 8](#_Toc37950786)

[Requirements Mapping 8](#_Toc37950787)

[Child Tester 9](#_Toc37950788)

[Requirements Mapping 9](#_Toc37950789)

[Test Drivers 9](#_Toc37950790)

[Requirements Mapping 9](#_Toc37950791)

[Logger 9](#_Toc37950792)

[Requirements Mapping 10](#_Toc37950793)

[Testing Requirements 10](#_Toc37950794)

[Requirements Mapping 10](#_Toc37950795)

[Constraints 10](#_Toc37950796)

# Overview

The following figures shows and overview of the architecture and class diagram.

## Architecture

The Test Harness architecture is described below. This is based off the architecture diagram provided by the course instructor, Professor Roueche.

The basic idea is that execution starts at the Test Executor. This can be a command line interface (CLI) or graphical user interface (GUI). This application allows the end-user to select tests from a set of Test DLLs. The Test DLLs, shown in the diagram below the architecture diagram, are simply executables without a main function that contain many functions which server as test functions that will be tested by the Test Harness.

Once the Test DLLs and functions are selected, the test are queued for the Test Harness to dequeue and submit for execution. The Test Harness creates Child Testers with the information about the DLL and test to execute. Each Child Tester executes one Test, but multiple Child Testers can run in parallel; i.e. in separate threads of execution.

Once the Child Testers complete, they each report their results to the Logger which in-turn logs a test results message to the Log Messages queue. This is then dequeued by the Test Executive to display the results.

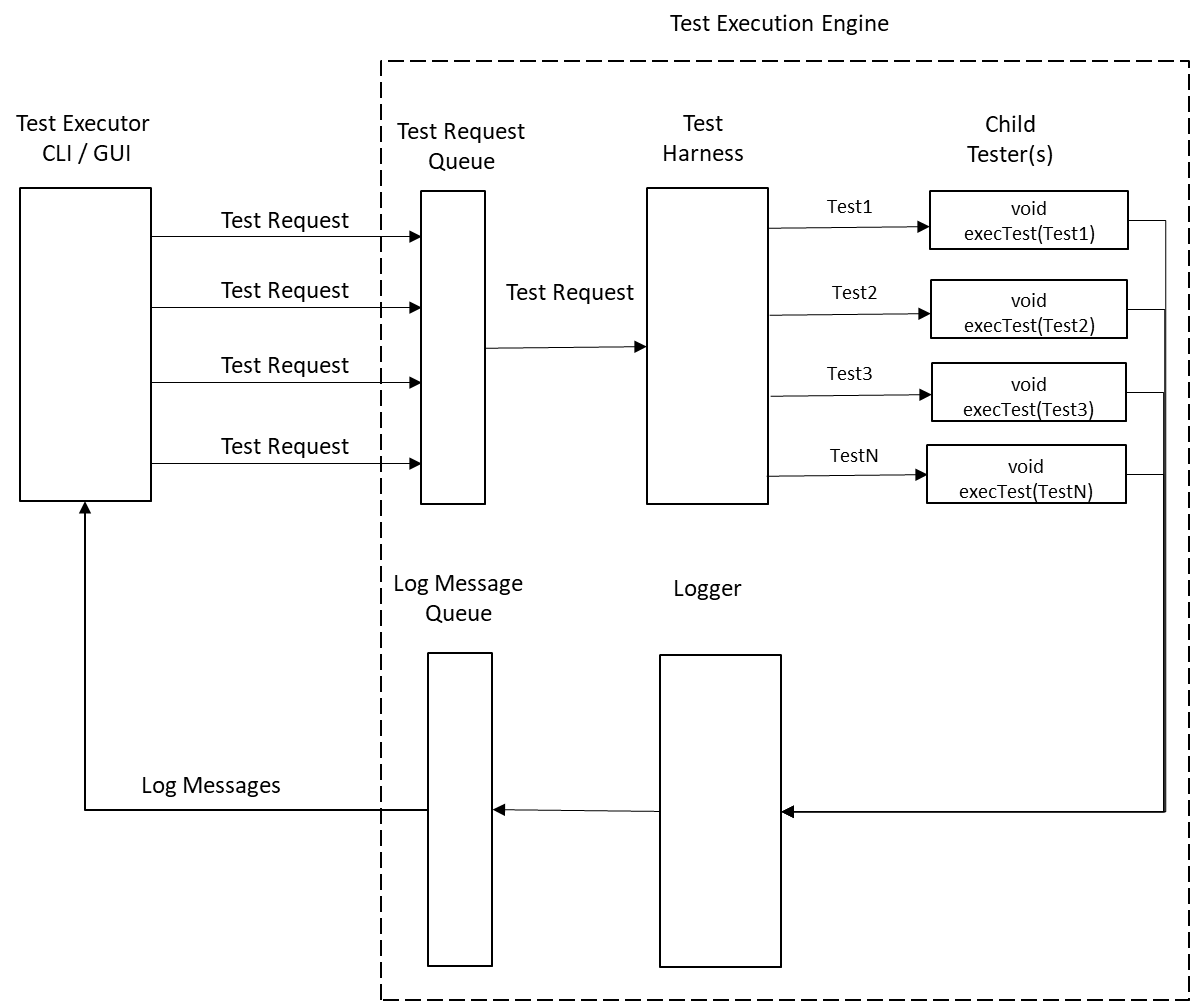


Figure 1: Architecture Diagram

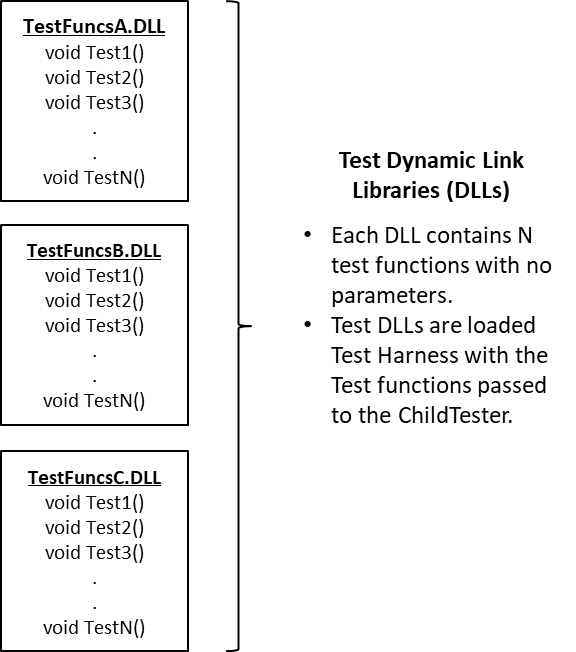
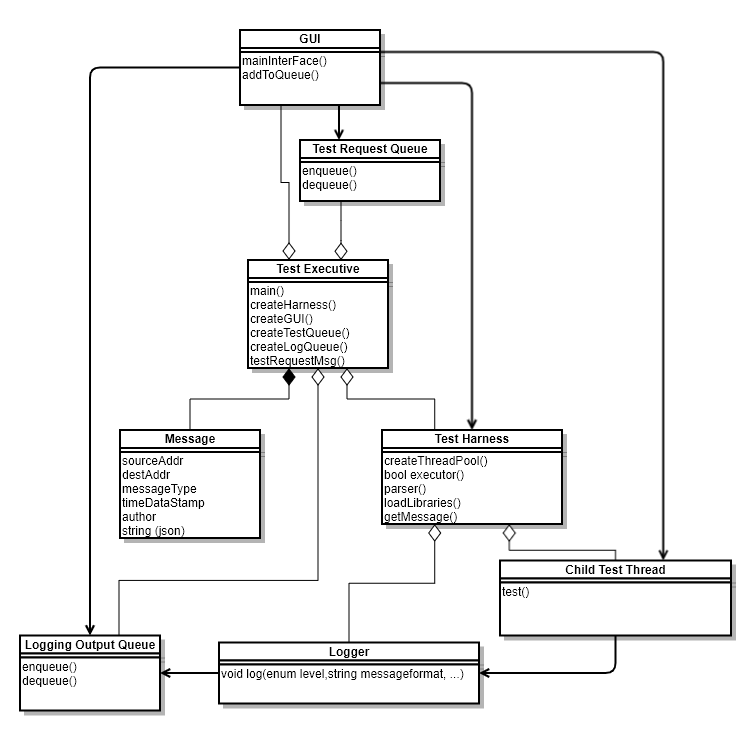


Figure 2: Test DLLs

## Class Diagram

The following classes will be implemented: Graphical User Interface (GUI), Test Request Queue, Logging Output Queue, Test Executive, Test Harness, Child Test Thread, Logger and Message. Those classes are described in the following sections.



Components

* Test Executive
  + Command-Line Interface
  + Graphical User Interface
* Messaging Queues
  + Test Request Queue
  + Logging Output Queue
* Test Harness
  + Child Tester
  + Test Object
* Logger
* Test Drivers

# Architectural Components

This section describes architectural components.

## Test Executive

The test executive will be the main driver of the application and will launch the user interface of the Test Harness project. In earlier phases, this will be in the form of a Command-Line Interface application. In later phases, this will be a Graphical User Interface application. While the interaction of the application will differ, the functionality which allows choosing tests, messaging them to the Test Harness and displaying results should remain the same.

### Requirements Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirement | Section | # |
| TE001 | The test executive shall include a graphical user interface (GUI) allowing the operator to define tests to be executed by the Test Harness. | Test Executive Requirements | 1 |
| TE002 | The test executive shall support building a Test Request by selecting library files from a repository using the GUI. | Test Executive Requirements | 2 |
| TE003 | The test executive shall define methods for sending and processing messages. | Test Executive Requirements | 3 |
| TE004 | The test executive shall provide facilities to transfer a Test Request to the Test Harness. (See Messaging Requirements for message details.) | Test Executive Requirements | 4 |
| TE005 | The test executive shall support viewing the test status on the GUI. | Test Executive Requirements | 5 |
| TE006 | Instantiate a test queue | Project 3 Requirements | 7 |
| TE007 | Instantiate a logging queue | Project 3 Requirements | 7 |
| TE008 | Instantiate the Test Harness | Test Harness Requirements | 1 |
| TE009 | Instantiate capability for command line interface | Test Executive Requirements | 1 |

## Messaging Queues

Message queues will be used as the inter-process communication mechanism between application processes. Two queues will be used:

1. Test Request Queue
2. Logging Queue

The Test Request Queue will receive test requests messages from the Test Executor. These messages will be dequeued by the Test Harness in order to communicate which tests to run.

The Logging Queue will receive log messages from the Test Harness. The Test Executor will dequeue these messages and display the logging messages.

### Requirements Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirement | Section | # |
| MQ001 | The test system shall implement a message-passing communication channel using sockets. This can be based on the Channel prototype found in the class Repository. | Messaging Requirements | 1 |
| MQ002 | The messaging system shall provide:  - Source and Destination addresses  - Message type  - Author  - Time-Date  -String body:  Expected to hold an XML string supplying information needed to execute a specific request. | Messaging Requirements | 2 |
| MQ003 | Each communication end point (FN #4) shall provide both a sender and a receiver (FN #5). | Messaging Requirements | 3 |

## Test Harness

This component is a application which is responsible for the execution of the tests. It primarily dequeues tests, loads the DLLs, calls the Child Tester to execute test (i.e. functions) within the DLL and enqueues results of the test.

### Requirements Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirement | Section | # |
| TH001 | The test system shall provide a Test Harness class that defines an executor method, which accepts any callable object that requires no arguments (FN #2) and returns a Boolean predicate to indicate success or failure. | Test Harness Requirements | 1 |
| TH002 | The Test Harness shall accept a Test Request XML string defining one or more test elements by naming dynamic link libraries. | Test Harness Requirements | 5 |
| TH003 | The Test Harness shall create, at startup, a number of child Tester processes or thread pool implemented as a queue. The child processes should post Ready messages to the Test Harness on startup and upon completion of a test. | Test Harness Requirements | 6 |
| TH004 | The Test Harness shall define methods for processing Test Request messages and sending Test Status messages. | Test Harness Requirements | 7 |
| TH005 | On receiving a Test Request, the Test Harness shall parse the request into individual tests and assign each test to a single child Tester or thread object. | Test Harness Requirements | 8 |
| TH006 | The Test Harness shall, on completion of processing of a Test Request, send a named, time-date stamped, test status message to the Client requesting the test. | Test Harness Requirements | 10 |
| TH007 | The Test Harness shall support any number of callable objects for test execution and shall provide a mechanism for execution sequence. | Project1 Requirements | 4 |

## Child Tester

This component executes a tests given a test function. After execution it provides results based on whether the function passed or failed.

### Requirements Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirement | Section | # |
| CT001 | The Test Harness executor method shall invoke a passed callable object in the scope of a try block. | Test Harness Requirements | 3 |
| CT002 | If exceptions are thrown during execution of the test, the catch block shall log test failure and the contents of the exception message. | Test Harness Requirements | 4 |
| CT003 | Testers shall load the corresponding library, create an instance of the Test Driver, and log pass-fail status of each test, along with a time-date stamp. | Test Harness Requirements | 9 |
| CT004 | Shall report to the Test Harness completion and ready to receive another test (queue) | Testing Requirements | 2 |

## Test Drivers

The test drivers are dynamic link libraries which have callable functions that will be used by the Test Harness for testing.

### Requirements Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirement | Section | # |
| TD001 | Test Drivers shall implement a common interface that declares a test method taking no arguments and returning a Boolean value indicating test pass or failure. | Testing Requirements | 1 |
| TD002 | Test Drivers will be run concurrently by child processes / threads | Project Requirements | 1 |

## Logger

The Logger component is responsible for logging test result output to the Logging Queue. It also supports various logging levels for less or more verbose output. These levels consist of INFO, DEBUG and ERROR. The INFO level describes programmer specific information and for pass fail reporting. The DEBUG level describes more specific programmer provided information to aid in debugging the software, this level will be used to provide application specific messages for the test results. The ERROR level will describe the most detail debugging output for the examination of software test failures.

### Requirements Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirement | Section | # |
| LG001 | The Test Harness shall provide a multi-level logging mechanism, intended to show:  - Just pass-fail status  - Application specific messages for pass and fail, along with the result  - Detailed debugging output that includes level-two logging and a time-date stamp | Test Harness Requirements | 2 |

# Testing Requirements

These are requirements the project must satisfy to prove that the test harness can operate as required.

## Requirements Mapping

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirement | Section | # |
| TR001 | The test system shall provide a test case where several Test Requests are sent in quick succession to demonstrate that the process pool executes Test Requests concurrently and that each will, on completion, post a Ready status message to the Test Harness and await the next Test Request message. | Testing Requirements | 2 |
| TR002 | The test system shall provide a sequence of tests demonstrating all the features of your Test Harness. | Testing Requirements | 3 |

# Constraints

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Requirement | Section | # |
| CN001 | Shall be written using the standard C++ programming language and the standard C++ Libraries. | Project Requirements | 1 |
| CN002 | Shall be developed using latest Visual Studio Community Edition. | Project Requirements | 2 |
| CN003 | Shall implement a message passing communication using Sockets, based on the Channel prototype found in the Repository | Project 3 Requirements | 2 |
| CN004 | Test driver shall be in the form of a dynamic link library with known interface to the Test Harness | Project Requirements | 1 |