# UDI Driver Test Suite Design

by

Gian-Carlo Bava (SCO)
Barry Field (SCO)

**Change History**

| Revision | Date | Description |
|---|---|---|
| 0.1 | 10/06/1999 | Draft. |
| | | |

**Approvers:**

TBD
TBD
TBD

## 1 Introduction

This document contains the design for the UDI driver test suite. This includes a design for UDI NIC and SCSI driver test cases as well as a test framework to support the test cases.

Presented first are the design goals. The design proposal overview follows.

The design proposal overview is a high level view of how the test suite is constructed.

Following the design proposal is the detailed design. First is the UDI NIC drivers test cases design, followed by the UDI SCSI drivers test cases design, followed by the design of the test framework.

Finally, Portability, Extensibility, Packaging, and Documentation design detail is presented.

Each design section begins with a proposal section. The proposal section describes the design decisions that need to be made.

### *1.1 References*

**UDI Network Driver Test Suite Specification**
http://hamilton.pdev.sco.COM/Projects/uditspec.htm

This is the UDI NIC driver test specification document. The UDI NIC driver test cases are described with pass/fail points given for each test case.

The UDI NIC Driver Test Cases Design, section 4.2 below in this document, uses the UDI NIC driver test specification document as input. The UDI NIC Driver Test Cases Design in this document either satisfies the test specification or gives a rationale for a deviation from the test specification.

## 2 Design Goals

- Portable across UNIX platforms.
- Enable developers to write quality UDI drivers with a shortened and efficient test cycle.
- The architecture is extensible. Test suites for more metalanguages can be easily added. New test case operations can be easily added.
- The test is automated. Once configured and started the test cases run to completion unattended.

## 3 Design Proposal Overview

The design proposal consists of constructing three functional blocks, the UDI NIC and SCSI Test Cases, the UDI Test Controller, and the UDI Test Mapper. Additionally, there are two interfaces, the Test Case-Controller Interface and the Controller-Mapper Interface.

The UDI driver test suite includes a set of UDI NIC driver test cases and a set of UDI SCSI driver test cases. The test cases run the tests. They perform operations and determine if the operations pass or fail.

The test cases communicate with the UDI Test Controller to perform test operations. The communication method between a test case and the UDI Test Controller is called the Test Case-Controller Interface. The Test Case-Controller Interface provides a means for a test case to perform functions pertinent to the test case

The UDI Test Controller interprets test case operations and formulates a sequence of metalanguage operations for the UDI driver. The UDI Test Controller communicates the sequence of metalanguage operations to the UDI Test Mapper over the Controller-Mapper Interface. The Controller-Mapper Interface utilizes the GIO metalanguage. To perform a single test case function, the Test Controller may command the Test Mapper to perform any number of metalanguage operations.

The Controller-Mapper Interface provides a means to perform metalanguage specific channel operations to the UDI driver under test. The UDI Test Controller initiates metalanguage operations across this interface without being involved with the metalanguage itself.

The UDI Test Mapper is a substitute for the metalanguage interface provider. For example, the UDI Test Mapper would take the place (for purposes of testing) of the operating system Network Service Requester (NSR). In this case, the Test Mapper uses NIC metalanguage channel operations to put a UDI NIC driver under a controlled test.

The Test Mapper merely formulates metalanguage control blocks and performs metalanguage functions while the Test Controller determines which metalanguage functions are performed.

The UDI Test Controller is an external mapper. It interacts with the operating system environment and the UDI environment. The UDI Test Mapper is an internal mapper. It exists entirely within the UDI environment. As much functionality as possible is contained within the UDI environment. The portability of the UDI environment is leveraged to achieve the portability design goal.

A diagram of the architecture is shown below.

UDI Driver Test Suite Architectural Overview

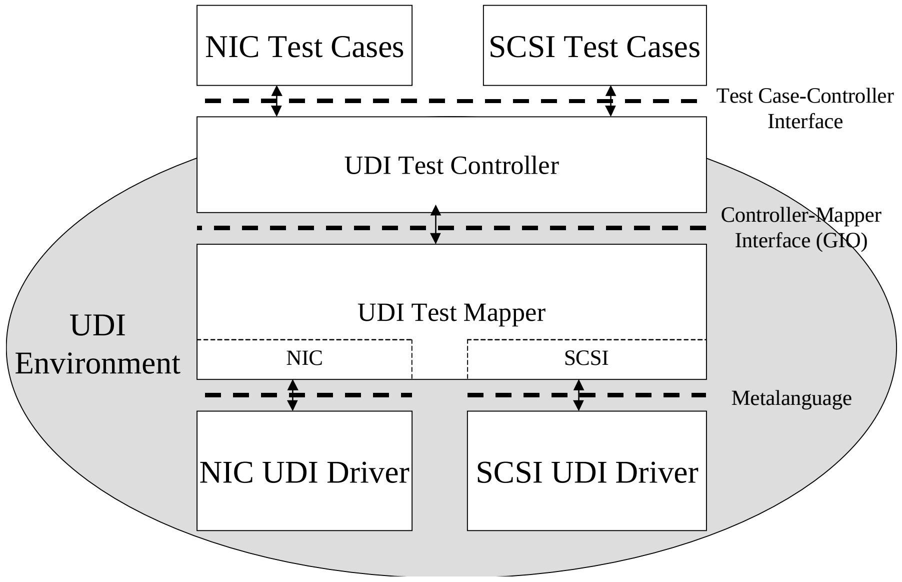

**Figure 3-1 Architectural Overview**

# 4 UDI Driver Test Cases

## *4.1 UDI Driver Test Cases Overview*

The UDI drivers test suite includes a collection of test cases for NIC and SCSI UDI drivers. A test case is a logical set of operations that indicate a pass or fail result. Each operation that produces a pass or fail result is a test purpose.

The test specification document describes the test cases and the test purposes for each test case. For each test purpose, the operation, relevant parameters, and the expected result from the operation is specified. The expected result is the determination point of whether the test purpose passes or fails.

For example, the expected result might be the status value of UDI_OK returned from a control block operation. When the test is run, if UDI_OK is returned as the status value of the control block, then the test purpose passes. If not, the test purpose fails.

The test specification document does not describe how to build the test. It merely describes what to test. The test case design, in this document below, describes how to build the test described by the test specification. The test specification is used as input to the test case design.

The test case design for UDI NIC and SCSI drivers, using the test specification as input, considers the test operations that need to be performed, how to implement them, and how to indicate and report pass and fail results.

The test case design also considers the user interface. The test case user interface design considers how users configure tests, how users execute tests, how results are reported, and what is logged. A test harness can be considered to provide these functions.

The test case design is used as input to the Test Case-Controller Interface design.

### *4.2 UDI NIC Driver Test Cases Design*

TBD

#### 4.2.1 Test Case 1

#### 4.2.2 Test Case 2

#### 4.2.3 Test Case 3

### *4.3 UDI SCSI Driver Test Cases Design*

TBD

#### 4.3.1 Test Case 1

#### 4.3.2 Test Case 2

#### 4.3.3 Test Case 3

4.3.4

## 5 Test Framework

TBD

### *5.1 Test Case-Controller Interface*

#### 5.1.1 Test Case-Controller Interface Overview

The interface mechanism for the Test Case-Controller Interface needs to be designed. The set of requests and responses between the Test Case and Test Controller needs to be designed.

The interface mechanism needs to be portable. The interface needs to be extensible to include new interface requests and responses.

The Test Case-Controller Interface is used as input to the UDI Test Controller design.

### 5.1.2 Test Case-Controller Interface Design

TBD

## *5.2 UDI Test Controller*

### 5.2.1 UDI Test Controller Overview

The UDI Test Controller design considers how to interact with the Test Case-Controller Interface and the Controller-Mapper Interface. The design considers how to convert Test Case-Controller Interface requests into a sequence of Controller-Mapper Interface requests. The design considers how to interpret responses from the Controller-Mapper Interface and pass results to the Test Case-Controller Interface.

The design considers how the Test Controller handles multiple metalanguages and corresponding test cases. Is there a unique Test Controller for each metalanguage/test case set or does a single test controller handle all metalanguages? How is the Test Mapper designed to facilitate extensions to the Test Case-Controller Interface and the Controller-Mapper Interface?

The UDI Test Controller design is used as input to the Controller-Mapper Interface design.

### 5.2.2 UDI Test Controller Design

TBD

## *5.3 Controller-Mapper Interface*

### 5.3.1 Controller-Mapper Interface Overview

The Controller-Interface utilizes the GIO metalanguage. The set of requests and responses between the Test Controller and Test Mapper needs to be designed.

The interface design considers extensibility. Additions to test case functionality may cause extensions to the Controller-Mapper Interface.

The Controller-Mapper Interface design is used as input to the UDI Test Mapper design.

### 5.3.2 Controller-Mapper Interface Design

TBD

## *5.4 UDI Test Mapper*

### 5.4.1 UDI Test Mapper Overview

The UDI Test Mapper design considers how to interact with the Mapper-Controller interface, how to convert from Controller-Mapper Interface requests to metalanguage requests, and how to masquerade as a UDI metalanguage provider under the control of the Test Controller.

The design considers how the Test Mapper accommodates multiple metalanguages.  Is there a single mapper containing all metalanguages such as SCSI and NIC or is there a separate mapper for each?  How is the Test Mapper extended to include other metalanguages?

#### 5.4.2 UDI Test Mapper Design

TBD

### *5.5 Configuration*

#### 5.5.1 Configuration Overview

Configuration considers what must be in place before any of the test software begins to run. Configuration sets up the Test Mapper, Test Controller, and the test cases with the proper environment and configuration parameters to run the test.  The Test Mapper needs to be configured as a substitute UDI metalanguage provider for the driver under test.  The Test Mapper and Test Controller need to be configured for a GIO metalanguage interface.  Test cases may have configurable parameters.

#### 5.5.2 Configuration Design

TBD

### *5.6 Initialization*

#### 5.6.1 Initialization Overview

Initialization considers the software that is run to prepare the test framework (Controller, Mapper) for execution of test cases.  It also considers preparation by a test case to provide a proper running environment for the test case.  Typically, configuration parameters are read, machine state is checked or setup, and default or initial state values are set up.

#### 5.6.2 Initialization Design

TBD

# 6 Portability

### *6.1 Portability Overview*

The elements of the test suite outside of the UDI environment such as the test cases, the Test Case-Controller interface, and the top half of the Test Controller need to be constructed in such a way that they can be built to run on any UNIX platform.

Elements inside the UDI environment such as the bottom half of the Test Controller, the Test Mapper, and the UDI driver under test are portable by definition.

### *6.2 Portability Design*
TBD

## 7 Extensibility

### *7.1 Extensibility Overview*
The design considers two vectors of extensibility.

The test framework can be extended for test of other metalanguages. A set of test cases may be developed for a metalanguage other than NIC or SCSI.

The test framework can be extended within existing metalanguages. NIC or SCSI test cases may require new test operations. This kind of extensibility could have impact on the Test Case-Controller Interface, the Test Controller, the Controller-Mapper Interface, and the Test Mapper.

### *7.2 Extensibility Design*
TBD

## 8 Packaging

### *8.1 Packaging Overview*
Packaging considers source and binary distribution, versioning, installation, and removal of the test suite. The distribution format, install, and remove technology must be portable to all UNIX platforms. The packaging design must be extensible for easy addition or removal of components.

### *8.2 Packaging Design*
TBD

## 9 Documentation

TBD