# A Perl Based HL7 Test Harness to support Load Generation and Bug Isolation

Robert Sabourin
President and Principal Consultant
AmiBug.com, Inc.
Presented at WOPR Conference September 2004

## The Problem Summer 2003

Customers of ITR, a medical information systems company, have observed many problems using the ITR software. ITR has been unable to reproduce and diagnose these problems in a development environment. ITR required a tool, which allowed them to reproduce and debug field reported problems, in a controlled development environment. AmiBug.Com, Inc. was engaged by ITR to help set up tools enabling developers to reproduce and correct the problem in their labs.

## Business Context

ITR software is used to manage information about medical imaging including all manner of x-rays, scans and various electronic patient records.

ITR customers expect the software to be very reliable. It was an ITR business priority to ensure field reported problems were resolved quickly.

ITR has several dozen clients located throughout the world. It is a small company with six full time developers without an independent testing team.

ITR budget for this testing project was very low and only allowed for about 10 days of consulting over a four-week period. There was no budget for commercial tools.

## Technical Context

ITR software was implemented on a series of Linux servers installed at the clients' facilities. Each server implemented part of the ITR solution and was generally dedicated to one specific ITR function.

One server managed data while another acted as a gateway.

Servers communicated with each other, and with external third party medical information systems, using the HL7 protocol. HL7 is a standard messaging system used in the medical information processing community. HL7 transactions include a series of requests and queries. (ITR supported a subset of HL7.)

HL7 is implemented on top of TCP/IP. Each ITR server listened to a reserved TCP/IP port for relevant messages and queries.

The ITR software was implemented in C++ using an open source SQL database.

ITR software keeps detailed logs about all transactions including a full capture of all HL7 requests and queries. (Including when request was received, when response was issued and all associated data)

Many ITR field reported bugs related to collisions in database record access.

## The solution

AmiBug proposed a test harness design, which used information from log files, to recreate the transactions on controlled servers in the development lab at ITR.

A data driven approach was taken, using action words, in a test manifest, which would be parsed and executed. (A CSV format was used for the test manifest, which allowed developers to create new test descriptions using MS Excel).

Perl was selected as the language of choice for the test harness. The ITR development team and AmiBug staff were experienced Perl programmers.

A Windows 32 development environment was selected due to the availability of systems at AmiBug offices. A goal was to write Perl Scripts, which could be ported between Linux and Windows systems.

Various open source tools were used to support development of this project and simulate host-to-host HL7 communications over TCP/IP.

## Implementation of ITR Test Harness

A series of three tools were developed using Perl scripts to implement the ITR test harness.

1. Test Manifest generation, Create input to the test harness from a series of ITR LOG files.
2. Test Progress Monitor, Track test harness execution.
3. Test Harness Engine, parse and implement the test Manifest. Includes controlling timing between requests and also capturing and confirming against expected results for query responses.

Development took about 8 days. This included gaining a sufficient (Just Enough) understanding of log file structures and HL7 protocol.

## Use of ITR Test Harness

Once the ITR Test Harness was operational, an experiment was set up to recreate a field reported defect from a clinic in New Zealand.

The log files from New Zealand were collected and converted into Test Manifests. (About 10,000 HL7 transactions)

The New Zealand reported defect was repeatable consistently in the lab by running the Manifest through the test harness.

The ITR Test Harness was used to repeat the bug while different diagnostics were used by the developed to help isolate and correct the defect.

Over a two-day period the capabilities of the test hardness were exercised on the New Zealand clinic problem reproduction.

Impact of tool:

- ITR developers able to repeat isolate and correct some important field reported defects
- ITR development staff motivated to use Test Harness as part of Unit Testing
- ITR used the test harness extensively for a period of about three months (shakedown period)

Problems with ITR Test Harness

- No internal champion to continue development of tool in house
- Not as portable as I would have liked due to win32 vs Linux Perl differences especially for spawning processes and low level TCP/IP
- Should have made it a bit more modular

Benefits of ITR Test Harness

- Development Cost was low
- It worked and it still works! (Although it is used less frequently).
- Developers implemented a more realistic load testing environment
- Usable early in development cycle

Other points

- Generic Test Harness Perl Source Code (generic not ITR version) is available on request from AmiBug.com as an instructive example of how to use Perl to make a data driven keyword based test harness.

Basic Structure,

The test bed allows series of HL7 commands to be sent to a server.

The input file format is taken from the ITR HL7GateWay Server input Log file format. One input file is required for each HL7 command.

Test Harness Directory Structure

D:\users\amibug\cvs_tree\Testing.old\Tools\Hl7GatewayServer\

- Root directory of test structure
- Used to launch test runs from this directory
- All scripts are relative to this directory

D:\users\amibug\cvs_tree\Testing.old\Tools\Hl7GatewayServer\Real_Client_DATA

- Example log data files taken from real client systems
- Used to create realistic test cases to replay actual transactions from a client system

D:\users\amibug\cvs_tree\Testing.old\Tools\Hl7GatewayServer\Test_Data

- Launching pad for test runs
- All log files, master files and test manifest files are placed in this directory before the tests are launched.

D:\users\amibug\cvs_tree\Testing.old\Tools\Hl7GatewayServer\Test_Results

- A new subdirectory of Test_Results is created to store the results of each test run
- Results include
    - Copy of TestHarness.pl used at time test is run
    - Copy of manifest.csv used at time test is run
    - Copy of all test input files
    - Copy of all test master files
    - Copy of  test run log file
    - Copy of one log file for each test case
    - The subdirectory path name is derived from the time to the second when the test run is launched combined with the test index (which is 0 when only one test is run)

D:\users\amibug\cvs_tree\Testing.old\Tools\Hl7GatewayServer\Test_Sandbox

- Working area for manipulating log and data files to create test runs
- Existing areas include
    - Addqry

        - Perl scripts used to create test manifest, test cases and test masters for any number of sequential add / query sequences. Example of how to automatically generate a series of test cases based on a working set of example test files.
    - Hlfiles
        - Perl script HL7FILES.pl that generates a test manifest from a directory of test files. HL7FILES.pl includes different examples to parse file names and generate test manifest files.
        - Provides a list of files and hl7 message type from each file one per line. This is useful to determine which HL7 commands are used in a collection of log files.
    - Preplogs
        - Perl script used to strip 0x0A from log files.
        - Perl script to create batch file to strip 0x0A from series of files.

    - Msocs
        - Msocs is a windows test program that allows simple messages to be interchanged between a client and server.
        - Useful for debugging HL7 format data packages and to ensure host client connectivity whenever in doubt
        - Tool is in public domain and is recommended by the HL7 standard web site

D:\users\amibug\cvs_tree\Testing.old\Tools\Hl7GatewayServer\Test_Sandbox\msocs

D:\users\amibug\cvs_tree\Testing.old\Tools\Hl7GatewayServer\Doc

- Documentation

Files in Root Directory

AmiTestLauncher.pl

- Launches in parallel the indicate number of instances of the test defined in Test Data
- Manages execution of test harness
- Manages process creation and monitors status
- Maintains a log of test runs with one entry for launch and one entry for termination of each test run
- Uses Win32 API

AmiTestHarness.pl

- Executes test commands from Manifest.csv
- Data driven
- Action word driven

- Uses Socket interface to communicate to HL7 server
- Confirms response to command matches master if a master is provided
- Each line of Manifest.csv is a separate command to the test harness

testrunlog.csv

- Log file maintained by AmiTestLauncher.pl

Notes ideas for future work

- Review variable declarations in perl source, move to rigid declaration or compiled model
- Generate master files for sample MRAD Real_Client_Data
- Create Add Merge generator based on model of ADD Query generator.
- Create HTML test result view which links data in testrunlog.csv to actual repositories and also test data directory
- Allow multiple Test Data directories through AmiTestLauncher.pl command structure

References

- Perl for dummies
- Perl in a nutshell
- Perl cookbook
- ActiveState web site

Perl

- Active State version 5.8 for windows.