

SOFTWARE TESTING AND IEC 61508 – PROJECT CASE STUDY

Published September 2008 in 'System Safety', the newsletter of the Safety Critical Systems Club, based at the University of Newcastle upon Tyne, UK..

Authors: Ian Gilchrist, (IPL Software Projects Manager, ian.gilchrist@ipl.com) with Wayne Flint (IPL Software Project Manager).

The text was approved by the UK Highways Agency.

SUMMARY

This paper describes the testing activities carried out by IPL in the implementation of a system to the safety standards mandated by IEC 61508. The bare facts of the project are that the system has been evaluated as SIL 1, and was coded in C++ with about 115 KLoC produced, tested and delivered to the customer. The project started in late 2002, and there have been three phased deliveries, in January 2004, April 2005, and March 2007. References given in [] refer to IEC 61508-3 (first edition).

HIGHWAY SYSTEMS AND THE NASS PROJECT

The client, the Highways Agency (HA), has responsibility for managing, maintaining and improving the motorway and trunk road network in England. To help with the task of avoiding congestion several computer systems are already in place. The current system used to control roadside equipment and monitor road conditions is called NMCS2 (National Motorway Communication System 2). An ATM (Active Traffic Management) pilot project is currently being run by the HA on one section (in the West Midlands) of the UK motorway network. The aim of ATM is to make best use of existing road space to increase capacity and ease congestion by controlling traffic according to actual and predicted road conditions.

In 2002 the HA went out to tender for the development of a new subsystem called the Network ATM Supervisory Subsystem (NASS), to form an additional element within the existing NMCS2 and future ATM systems. NASS takes real-time actual traffic flow data, combines this with historical traffic flow data, and then predicts future flows. If congestion is predicted, NASS will evaluate a number of predefined traffic control plans to avoid or minimise the predicted congestion, selecting the optimal plan. NASS will then issue requests for the settings of roadside signals and message signs to implement predefined traffic control plans.

The NASS contract was awarded to IPL in late 2002. The first milestone was a "proof of concept" (PoC) system, which came in at about 20 KLoC and completed Factory Acceptance Tests (FAT) in December 2003. The second phase was for the production of a demonstrator system to be supplied to Traffic Engineering consultants working for the HA. The purpose of this system was to allow those consultants to refine the rules and algorithms internal to NASS for its safe and efficient functioning. This was delivered in April 2005, and was approximately three times the code size (60KLoC) of the initial PoC system.

The current/third phase of NASS was delivered to the HA's West Midlands Regional Control Centre (RCC), located at Quinton, in March 2007. This is being gradually integrated with all the other systems running at the RCC with the aim hopefully of being able to demonstrate its worth by Summer 2008. At this point NASS could be used to directly request sign and signal settings thereby influencing drivers using the West Midlands motorway network with a view to reducing/mitigating congestion. Due to the target environment for NASS having evolved during its development it is likely that closer integration with existing user interfaces will be necessary prior to making this step. Once this is achieved new variants of NASS could be produced and installed at other RCCs (of which there are 7) across the UK's motorway network.

See Table 1 for a summary of the phases of the NASS project to date.

Phase	Name	Purpose	Delivered	Approx code size
A	PoC	Prove NASS concept	Jan 2004	20KLoC
B	Demonstrator	Testbed for refinement of NASS rules	Apr 2005	60 KLoC
C	NASS V1	For live use at West Midlands RCC	Mar 2007	115 KLoC

Table 1. NASS project development phases

SYSTEM SAFETY AND IEC 61508

The use of IEC 61508 was decided on by the HA as a result of consideration of the hazards involved. At the start of the NASS project, IPL engineers assessed that the safety level appropriate for the project was SIL 1. This relatively low grading reflects the fact that NASS does not directly control any hazardous equipment, but is involved in issuing requests for traffic sign and signal settings – which can have safety consequences when acted upon.

NASS SOFTWARE DESIGN

IPL started work in early 2003. Having agreed the system requirements in detail for the PoC phase, software design followed a method based on use of UML. The design hierarchy led to the identification of sub-systems, which in turn comprise software components, which can be either executables or libraries (DLLs). Further OO design decomposition leads to the identification of classes, for which module specifications were created ready for coding and testing by programmers. The module specs detailed class public and private methods with code flow shown in pseudo-code, and class test plans.

The initial (PoC) phase design had 19 software components (6 executables and 13 libraries) to make up the active NASS elements, comprising a total of 94 classes. The current phase delivery has grown to 8 executables and 15 libraries, comprising a total of about 280 classes. The NASS system runs on Windows, and code production has been done using MSVC++ V6.

TESTING STRATEGY

The IEC 61508 standard calls [7.3.2] for validation planning, and furthermore [7.7.2.6] that 'testing shall be the main validation method for software'. Accordingly the NASS project team together with the HA drew up a test strategy which included a formalised (i.e. under independent QA monitoring) approach to testing each and every entity at each every identifiable stage of the project. Each entity has its own test plan, which details as appropriate the configurations, inputs and expected outputs which, when run successfully, will give the required confidence in the correct working of the entity under test. At the higher levels the test plans were contained in a separate (version-managed) document; at the lower levels test plans were included in the design specification. Table 2 summarises the relationship between the principal design documents and the corresponding test/specifications.

Design Document	Informs test plan for...
Existing NMCS Documentation	System Interaction Test
NASS System Requirements Spec (SRS)	Factory and Site Acceptance Tests
Architectural Design Spec (ADS)	System Integration Test
Sub-systems Design Spec (SSDS)	System Integration Test
Component Specification	Component Test (in the design spec)
C++ Class Specification	Module/Unit Test (in the design spec)

Table 2. Hierarchy of principal requirements and design documents

Throughout IEC 61508 [e.g. 7.9.2.4] there are demands to the effect that test results should be generated to show that tests have been run and 'satisfactorily completed'. This put quite a necessity on the team to ensure that not only were they using tools that would make the

various testing activities as easy and repeatable as possible, but also that the tests should, as far as possible, be self-documenting.

CODE AND MODULE TESTING

Following classical 'V-model' lifecycle principles and IEC 61508 [7.4.7 Software Module Testing] the first task for the IPL software engineers after programming the classes was to test them. For this purpose the IPL tool Cantata++ was used. This was partly because the engineers were familiar with it, and also because it gave all the functionality needed to test to the IEC 61508 SIL 1 standard. The basic requirement is to test every class in isolation and to demonstrate code coverage to the levels of 100% entry-point (every function/method called at least once) and statements. In fact, IPL took the reasonable decision to additionally test to 100% condition coverage. This is more work than the basic project SIL demanded but was felt by the developers to give a useful additional level of confidence.

Since every class had a number of external interfaces, not all of which could be stubbed, the Cantata++ 'wrapping' facility was vital in allowing such isolation testing to be completed as planned. Stubbing involves the replacement of an external function with a programmable simulation having the same interface. Wrapping allows for the 'real' external code to be included in the test build, but with the option to intervene at the interface in order to, for example, check values being passed out or alter values being returned to the code under test.

INTEGRATION TESTING

Cantata++ was also used for the next level up of testing, namely component testing. This formed the first level of integration testing, and was aimed at verifying the correct working of each NASS executable or DLL against specifications defined in the appropriate level of design. The testing involved calling public interfaces of the component under test, and stubbing or wrapping calls to external functions. Since NASS has its own database the component tests included 'real' database code so that testing included the option to initialise the database and check that updates to the database were as expected.

The project created and has maintained a fairly elaborate regression test facility which has allowed for nightly builds and re-runs of each class and component test in the entire system. This has served very well to enhance confidence in the change impact analysis system by ensuring that changes in one module are completely and properly compensated for in other affected modules and tests.

After testing the components the project test strategy called for sub-system testing. Since 100% coverage had already been achieved during unit testing, integration testing could be allowed to live up to its name – namely testing the integration of the software units. The team, with the agreement of the HA, determined that 100% entry-point coverage was suitable to demonstrate the completeness of the integration tests. This is in fact exactly in-line with the 61508 requirement [7.4.8.3] to demonstrate that, "all software modules... interact correctly to perform their intended function..."

SYSTEM LEVEL TESTING

Following sub-system testing the team carried out a further series of System Integration tests which were formally documented by IPL QA staff. These tests mainly served as a dry-run to gain confidence before going into the customer-witnessed Factory Acceptance Tests. For the most recent delivery of the project FATs took 20 days to run, much of which time was occupied by reconfiguring of the system between each successive test run. It was noteworthy that the system integration tests ran smoothly and only revealed a few design anomalies.

The last layer of testing before the NASS was allowed to be installed at the RCC was called Interaction testing. This was carried out at the offices of another HA contractor, Peek Traffic, and involved running NASS on a rig which included NMCS2 equipment in exactly the same configuration as the live NMCS2 system. The intention here was to ensure that NASS could interact correctly with the other live systems at the RCC.

Lastly, Site Acceptance tests were held at the RCC to demonstrate that the live NASS was in fact operating correctly and safely with the rest of the NMCS2 system.

See Table 3 for a summary of testing levels within the NASS development project

Name of tests	Testing	Tool	Comments
Module/Unit tests	C++ classes	Cantata++	283 in total. 100% Entry-point, statement and condition coverage. Test plan is in the class specification.
Component tests	Components (individual executables or DLLs)	Cantata++	23 of these in total. 100% Entry-Point coverage. Test plan is in the component specification.
Integration testing	Aspects of ADS and SSDS	IPL-developed simulators	
System Test	Entire System	IPL-developed simulators	Dry-run of Factory Acceptance Tests, witnessed by IPL QA staff
Factory Acceptance Test	Entire system	IPL-developed simulators plus HA 'Portable Standard'.	Formal run of System Tests at IPL offices, witnessed by HA/consultants
Interaction tests	Entire system	Test Rig at Peek Traffic	Formal run of tests at Peek, witnessed by HA/consultants
Site Acceptance Test	Entire system	Live at RCC	Formal run of tests at RCC, witnessed by HA/consultants
[Regression tests	Re-runs of Unit and Component tests	Cantata++ and IPL-developed framework	Run nightly]

Table 3. Complete hierarchy of testing on NASS project, at third phase of the project

CONCLUSION

This article shows that a properly run 61508 project, even at a relatively low SIL can be demanding on test time. We wish to reinforce the point that testing is, on one level, about providing reassurance to developers that they can move with reasonable confidence from one stage of the project to another. At a different level it can provide confidence to other stakeholders (e.g. the customer) that the system will work safely and reliably when installed on site. A good project will work from a test strategy (i.e. determine in advance at what stages and levels in the lifecycle testing should be carried out), and will demand that test plans exist for each entity to be tested; testing should not be *ad hoc*! Testing needs to generate results as evidence of test completion (i.e. be self-documenting). Furthermore, testing needs to be conducted in a repeatable fashion because the one thing you can be sure of is the need to run and re-run tests at all levels many times.