

# Testing Grid Software on the Grid

S. Newhouse, C. Walker, J. Bradley

Open Middleware Infrastructure Institute,  
Suite 6005, Faraday Building (B21), Highfield Campus,  
Southampton University, Highfield, Southampton, SO17 1BJ, UK  
{sn,cw,jb4}@ecs.soton.ac.uk

## Abstract

The middleware being deployed to support grid communities is increasing in complexity. No longer can a single development team provide the entire software infrastructure needed to support a deployment to meet the needs of a particular community. Therefore, the assembly, testing and verification of this deployed middleware stack becomes increasingly important across different platforms and deployment environments. This paper and associated presentation will describe the Open Middleware Infrastructure Institute's approach to testing its own middleware distribution, assembled from different software providers, and how it is using conventional software testing infrastructures and working internationally to develop grid specific testing frameworks.

## 1. Motivation

The Open Middleware Infrastructure Institute (OMII) is tasked with integrating a robust and reliable software stack across a variety of platforms. A key element in this work is the quality assurance of the resulting stack. The OMII currently uses a combination of manual and automated testing to verify the behaviour of the software across different platforms.

This paper provides a brief summary of our current testing strategies (for OMII 1.2) as we increase the automation and thereby reduce the cost of testing within our current release cycles. The testing infrastructure used to support OMII 2.0 and its future evolution will be described in the presentation.

## 2. Manual Testing

Manual testing remains a key activity within OMII's quality assurance process. This has had a particular benefit in improving the quality of the installation, its associated documentation, and the feedback provided to the user when things go wrong. By packaging the distribution into several bundles (base & extensions, services, application and client) there is a natural interdependency between these. For instance, the base & extensions bundle needs to be installed before the services bundle.

OMII's System Testing comprises the usual functional and non-functional elements but in addition the focus of the OMII Test Team continues to be firmly fixed on finding the highest risk bugs and on being the "voice of the

customer". The functional side of system testing concentrates heavily on requirements and task-based test cases whereas the non-functional side continues to grow in order to assess the effects of various "environmental" factors upon the OMII user.

## 3. Platform Testing

The platform testing uses the 'build and test framework' developed through the National Middleware Initiative in the USA by the Condor group at Wisconsin (<http://www.cs.wisc.edu/condor/nmi>). This uses the Condor framework to co-ordinate execution of build and test commands across multiple platforms and environments. Key to this is the Condor framework which provides the multi-platform execution environment and workflow management through DAGMan. The core framework is configured to build and test a particular set of software through the provision by the user of a set of 'glue' scripts that are invoked by the framework. These glue scripts can be called at various points in a multi-platform workflow: on startup or shutdown of the framework, or before or after the task that this it to be executed on the build platform, and to define the build task itself.

As a demonstration of this framework we use it internally to build and deploy the OMII distribution and its supporting databases and Perl modules, across a range of platforms provided by the NMI pool at Madison and a local Condor pool within OMII. These binary bundles are then redeployed and tested using the same framework. Binary bundles that pass these tests are marked for download on an internal

website. Integration of the OMII's software into the NMI build and test framework was relatively straightforward once provision had been made within the OMII installation process for automated installs through the provision of sensible defaults that could be picked up from the environment.

#### **4. Automated Testing**

Various elements of the OMII's testing strategy already rely heavily on automated test tools. The software components within the main distribution are provided with JUnit tests that enable the validation of interfaces and their expected behaviour during code development. This functional stability is verified through prolonged stress testing of a deployed server and a number of clients on the main code base.

The client interaction with the web services is monitored during testing to verify that the message exchanges comply with the standards specified by the WS-I standard. The current distribution is also checked out nightly from CVS and installed on a test machine to verify the integrity of the current code base.

#### **5. Stress Testing**

While the testing strategies outlined so far will ensure that the software can be reliably installed in a variety of different environments from the supplied documentation to provide a working installation, no issues surrounding the performance of the deployed system have been addressed. As part of the in-house stress testing we regularly deploy a server installation and subject it to an ever increasing number of client connections. This has already helped to discover race conditions in the interactions between the OMII\_1 services necessary to execute an application on an OMII\_1 grid, and to identify improvements in the structure and interaction with the database tables that hold the state of the OMII\_1 services.

#### **6. Installation Testing**

During the installation of each software bundle within the distribution we verify its successful installation before proceeding to the next phase. For instance, when installing the service provider we verify that the Tomcat container can be started, is running, and can be stopped before attempting to deploy Axis. Likewise, we verify that Axis is up and running before attempting to deploy any web services into the hosting environment.

We also verify the integrity of the client installation by using service providers hosted by the OMII. During the installation we verify that the client is able to make an un-secured and then a secured invocation on a remote test web service. Later on in the installation we check the connection to the four main services hosted in the OMII service provider, before using these services to run a test application. Finally, if the user installs the application pack on their client machine, they can use the Cauchy client to run simulations using the Cauchy application installed on the service provider.

#### **6. Summary and Open Issues**

The OMII's testing regime continues to evolve as our product matures. We have already explored a variety of strategies from manual testing to automatic stress testing. While we expect testing to continue to occupy a significant proportion of our internal resources, we expect to do more testing with these resources as we make greater use of automated testing. An issue that we have only begun to explore is the effective testing across distributed resources of a complete deployed grid system.