# Dependability in Grids

**Grid-FIT has been implemented specifically to test SOAP based Web Service systems and Globus Grid systems.**

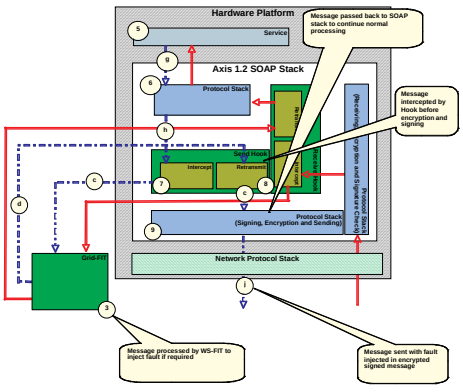
## Introduction

The Distributed Systems and Services (DSS) group at the University of Leeds has for many years established a track record and an internationally leading reputation in service-based software architectures and dependable distributed computing.

Two of the technologies developed by this group are described in this leaflet.

## Grid-FIT: Dependability Analysis of Grids

Grid-FIT (Grid – Fault Injection Technology) is a fault injector that utilizes network level fault injection to assess Grid systems. Grid-FIT has been implemented specifically to test SOAP based Web Service systems and Globus Grid systems. A detailed description of its design and operation are given in our internationally published research.




Our method and tools are designed in such a way that they can be used to test both standard Web Service systems and Globus systems.

Network level fault injection is usually based upon, the more or less random, corruption of bytes within a network packet. Our method extends this method to make meaningful perturbation to a middleware message, i.e. our method can target a single parameter within a middleware message sequence and change it.

The method uses an instrumented SOAP API that includes small pieces of hook code. A hook intercepts a message, transmits it via a socket to the fault injector engine and receives a modified message back from the fault injector. The modified message is then transmitted normally to the original destination. There are hooks to intercept both incoming and outgoing messages.

By instrumenting the SOAP stacks on strategic machines this method can be used as part of the certification testing for individual components within a production system without the need for a test harness.

Whilst a number of existing fault injectors could be used to do this, notably DOCTOR and Orchestra, these tools are designed for general purpose protocol testing. Grid-FIT has been designed around an engine to decode SOAP messages and presents an interface at the script level with the information included in a SOAP RPC easily accessible.

## Test Case Construction

The Grid-FIT Tool also includes an easy to use GUI to allow test cases to be constructed and executed. Skeleton test cases can be automatically constructed from the WSDL of a service. Grid-FIT then uses a customizable fault model to allow easy construction of concrete test cases. A customizable set of failure modes is also included into the tool and these can be linked to a generated test case to test the outcome of a test case. Finally the built-in profiling tools can be used to determine fruitful areas to concentrate testing.

**FT-Grid seeks to facilitate the development of design-diversity fault-tolerance in Grid and e-Science environments**

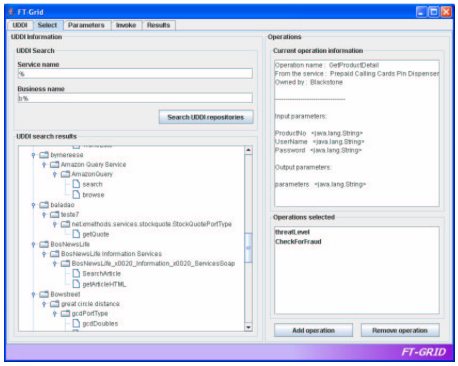
## Test Cases

Our research using Grid-FIT has shown that Grid-FIT can effectively be used to detect defects in system designs by exercising seldom used code pathways via fault injection techniques.

We have successfully applied our tools to a number of constructed test web service systems with promising results. Our next step will be to apply our tools too a number of real world systems and determine the scalability of our method.

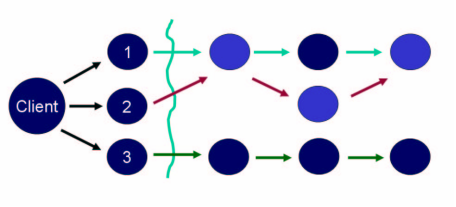
## FT-Grid: Design-diversity Fault-Tolerance in Grids

A traditional way to increase the dependability of distributed systems is through the use of fault-tolerant techniques. We have developed a java-based Fault Tolerance framework called FT-Grid which seeks to facilitate the development of dynamic, multi-version fault-tolerant service-based systems. It does this by allowing a developer to quickly employ ultra-late binding to search UDDI repositories, find services, invoke them and vote on the results – all at run-time.




A screenshot of FT-Grid

A potential problem when considering multi-version fault tolernace in a SOA is that the implementational details of a service are typically hidden from a client. This is an issue as services that initially appear disparate may – during the course of their execution – invoke one or more identical, "shared" services. In the case of figure 3, a client invokes 3 seemingly separate service. However, in the course of their workflow, service 1 and service 2 both invoke "common" services. Should a fault occur in one of these common services, then the potential exists for them to both fail in an identical manner, thus causing common-mode failure.




Shared services

One way to avoid this problem is to use the technique of provenance. Provenance is the documentation of the process that leads to a result, and acts as a record of how a piece of data has been created.

If we assume that data provenance is recorded, it will allow a fault tolerance scheme to build up a "view" of how each result it receives has been constructed. By possessing this view, weightings can be assigned to each service based upon how closely related it is to another service.

We have integrated the PreServ provenance scheme developed as part of the PASOA project at the University of Southampton with FT-Grid, and have extensively tested the resulting system against a large test system, into which we injected faults. We have then compared this data against both a single-version and a 'traditional' multi-version approach. The results are as follows:

| Approach | % correct results | % common-mode failure |
|---|---|---|
| Single-version system | 83.6% | n/a |
| Traditional MVD approach | 92.3% | 6.6% |
| Topological aware MVD | 99.4% | 0% |

## Further Information


Contact:

Mr Nik Looker (email: nlooker@comp.leeds.ac.uk)
Dr Paul Townend (email: pt@comp.leeds.ac.uk)
Prof. Jie Xu (email: jxu@comp.leeds.ac.uk)



e-Science
University of Leeds
The University Of Sheffield.
The University of York
Sun microsystems Center of Excellence
Yorkshire Forward
Yorkshire & Humber Regional Development Agency