The BROADEN Distributed Tool, Service and Data Architecture

Martyn Fletcher, Tom Jackson, Mark Jessop, Stefan Klinger, Bojian Liang, Jim Austin.

Advanced Computer Architectures Group,
Department of Computer Science,
University of York, YO10 5DD, UK.

Abstract

The organisation of much industrial and scientific work involves the geographically distributed utilisation of multiple tools, services and (increasingly) distributed data. A generic Distributed Tool, Service and Data Architecture is described together with its application to the aero-engine domain through the BROADEN project. A central issue in the work is the ability to provide a flexible platform where data intensive services may be added with little overhead from existing tool and service vendors. The paper explains the issues surrounding this and explains how the project is investigating the PMC method (developed in DAME) and the use of Enterprise Service Bus to over come the problems. The mapping of the generic architecture to the BROADEN application (visualisation tools and distributed data and services) is described together with future work.

1. Introduction

This paper describes an integrated and Distributed Tool, Service and Data Architecture developed for use in aero-engine health monitoring domain. The architecture has been developed to support the requirements of the BROADEN (Business Resource Optimization for Aftermarket and Design on Engineering Networks) project – part of the UK Department of Trade and Industry (DTI) Technology Innovation Programme. BROADEN is a follow on to the DAME (Distributed Aircraft Maintenance Environment) Grid pilot project [1].

Typically, as with many industrial applications and scientific research domains, the aero-engine domain needs to provide for distributed users with access to tools and distributed data. This coupled with Grid technologies [2] provide an opportunity to derive commercial and scientific benefit from distributed data. Such domains are often characterised by:

- Geographically distributed data – potentially in vast amounts.
- Geographically distributed users - not necessarily distributed to the same places as the data.
- Legacy Tools which are standalone but may need to interoperate with other tools.
- Tools which are designed to interoperate with other tools or services.
- Distributed services used by tools or other services.

A generic Distributed Tool, Service and Data Architecture is being developed to satisfy the above with aero-engine health monitoring as the exemplar domain. Modern aero engines operate in highly demanding operational environments with extremely high reliability. However, Data Systems & Solutions LLC and Rolls-Royce plc have shown that the adoption of advanced engine condition monitoring and diagnosis technology can reduce costs and flight delays through enhanced maintenance planning [3]. Such aspects are increasingly important to aircraft and engine suppliers where business models are based on Fleet Hour Agreements (FHA) and Total Care Packages (TCP). Rolls-Royce has collaborated with Oxford University in the development of an advanced on-wing monitoring system called QUICK [4]. QUICK performs analysis of data derived from continuous monitoring of broadband engine vibration for individual engines. Known conditions and situations can be determined automatically by QUICK and its associated Ground Support System (GSS). Less well-known conditions (e.g. very early manifestations of problems) require the assistance of remote experts (Maintenance Analysts and Domain Experts) to interpret and analyze the data. The remote expert may want to consider and review the current data, search and review historical data in detail and run various tools including simulations and signal processing tools in order to evaluate the situation. Without a supporting diagnostic

infrastructure, the process can be problematic because the data, services and experts are usually geographically dispersed and advanced technologies are required to manage, search and use the massive distributed data sets. Each aircraft flight can produce up to 1 Gigabyte of data per engine, which, when scaled to the fleet level, represents a collection rate of the order of Terabytes of data per year. The storage of this data also requires vast data repositories that may be distributed across many geographic and operational boundaries. The aero-engine scenario is also typical of many other domains, for example, many areas of scientific research, healthcare, etc.

We describe the development of an architecture which integrates geographically distributed users (tools), services and data. The following sections describe the generic Distributed Tool, Service and Data Architecture.

2. Architectural Issues and Development

This section provides an overview of the main characteristics of the tools, services, and data including the issues and rationale behind the development of the architecture.

2.1 Tool characteristics

A major issue within BROADEN has been the need to allow users to add tools to the system with the minimum of change to the tools. Our analysis has show the types of tools used can be characterised as follows:

- Tools designed to operate standalone; this is typical of legacy tools.
- Tools not designed to interoperate with other tools; again this is typical of legacy tools.
- Tools designed to use remote services.
- Tools designed to interact with other tools.

The main requirement is that the architecture should allow all the above tool types to interact with one another as necessary, with a minimum of change both to the tools and the system architecture (preferably none).

2.2 Service characteristics

Underlying the use of tools is the provision of services. These follow the requirements of the tools in section 2.1. The services used can be broken down into the following categories:

- Services may be centralised. This is typical in legacy systems.
- Service may be distributed – particularly located near the data sources.
- Service may be autonomous e.g. data input services triggered by data arrival.

The architecture should accommodate all the above services.

2.3 User characteristics

The system may have many geographically dispersed users, who may not be located in the same places as the data repositories. The architecture therefore should accommodate the use by geographically distributed users.

2.4 Distributed Data

The scenario, which we will describe, is one in which every time an aircraft lands, vibration and performance data is downloaded to the system from the QUICK system fitted to each engine. In future deployed systems, the volume of the Zmod data downloadable from the QUICK system may be up to 1 GB per engine per flight. Given the large volume of data and the rate at which it is produced, there is a need to consider how the data is managed.

In order to illustrate the requirements of the aero-engine application, consider the following scenario. Heathrow, with its two runways, is authorized to handle a maximum of 36 landings per hour [5]. Let us assume that on average half of the aircraft landing at Heathrow have four engines and the remaining half have two engines. In future, if each engine downloads around 1 GB of data per flight, the system at Heathrow must be capable of dealing with a typical throughput of around 100 GB of raw engine data per hour, all of which must be processed and stored. The data storage requirement alone for an operational day is, therefore, around 1 TB, with subsequent processing generating yet more data.

With such a vast amount of data being produced a centralised repository may place unacceptable demands on data bandwidth and will not provide a scaleable solution. Portions of data could be moved to services but this would represent only a solution in some limited application as most services will need the ability to use full data sets. Therefore, it is desirable to store data in distributed nodes and then the services which act on the data must also be distributed with the data – to avoid having to move the data to the services [6].

The architecture should permit the use of distributed nodes to store data (and services),

assuming other issues such as security, etc. are satisfied.

3. The Generic Distributed Tool, Service and Data Architecture

The scenario used for the DAME and BROADEN projects envisages geographically dispersed users using a workbench and tools which access distributed services and data.

Figure 1 provides an overview of the generic architecture that has been developed. The elements shown are:

- The Graphics and Visualisation Suites which contain a set of tools at a particular user location.
- The Enterprise Service Bus to enable tool interactions.
- The distributed nodes encapsulate the data and high performance services - see figure 2.
- The global workflow manager provides service orchestration on demand from the individual users or as an automatic activity in response to a predetermined stimulus.

Figure 2 shows the generic overview of the generic architecture:

- The Process Management Controller (PMC) is a distributed component which manages the distribution of service requests and collection and aggregation of results for return to the requester (tool).
- The Processing Services act on local data and provide high performance search, signal extraction facilities, etc.
- Distributed Data Management provides access to local and distributed data through Storage Request Broker abstraction mechanisms.
- The Local Workflow Manager Management provides automatic and requested workflows to be enacted with a single distributed node. A local workflow may also be part of a global workflow controlled by the global workflow manager.
- The Local Resource Broker manages selection of local resources in keeping with specified Service Level Agreements (SLAs).
- Data Loading Services populate the local data stores in each node. These services may also perform other tasks such as data integrity checking, error correction, etc. on input data.
- Databases / Data stores are provided for each type of data resident in the node.

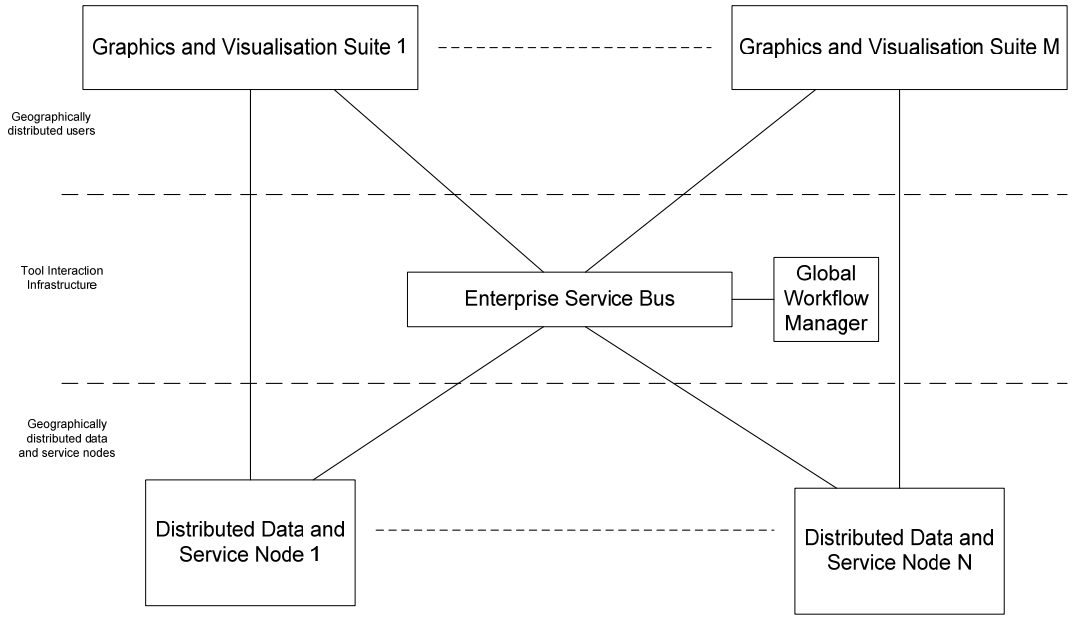

Figure 1 Overview of the Generic Tool, Service and Data Architecture

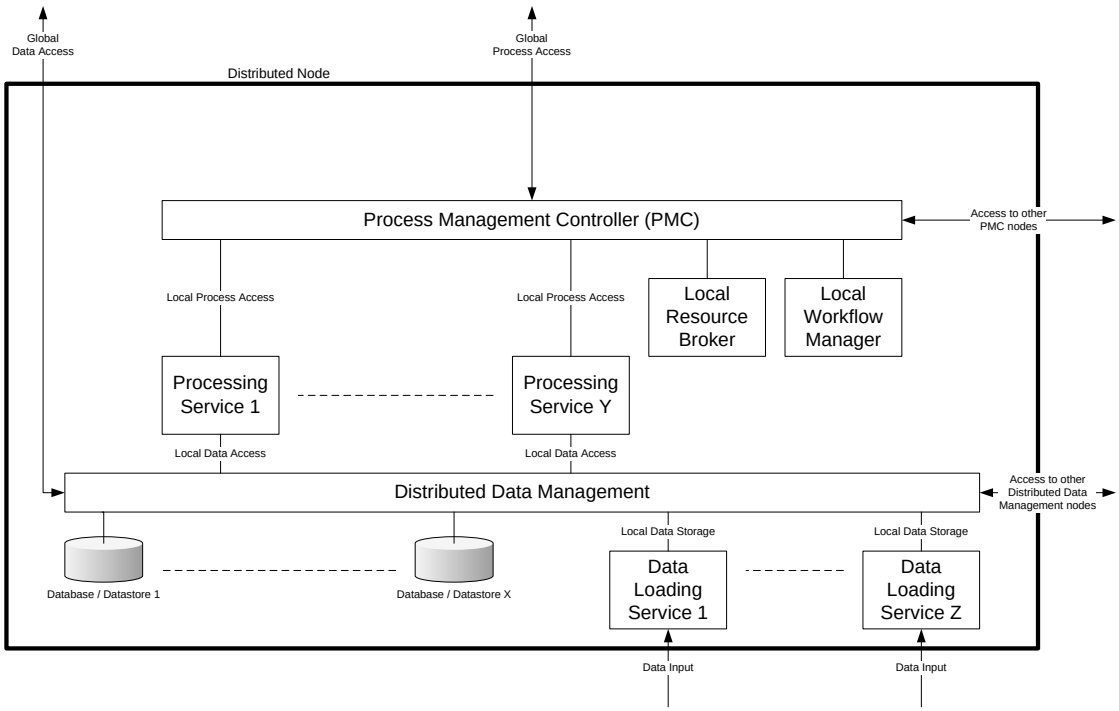

Figure 2 Overview of a Generic Distributed Service and Data Node

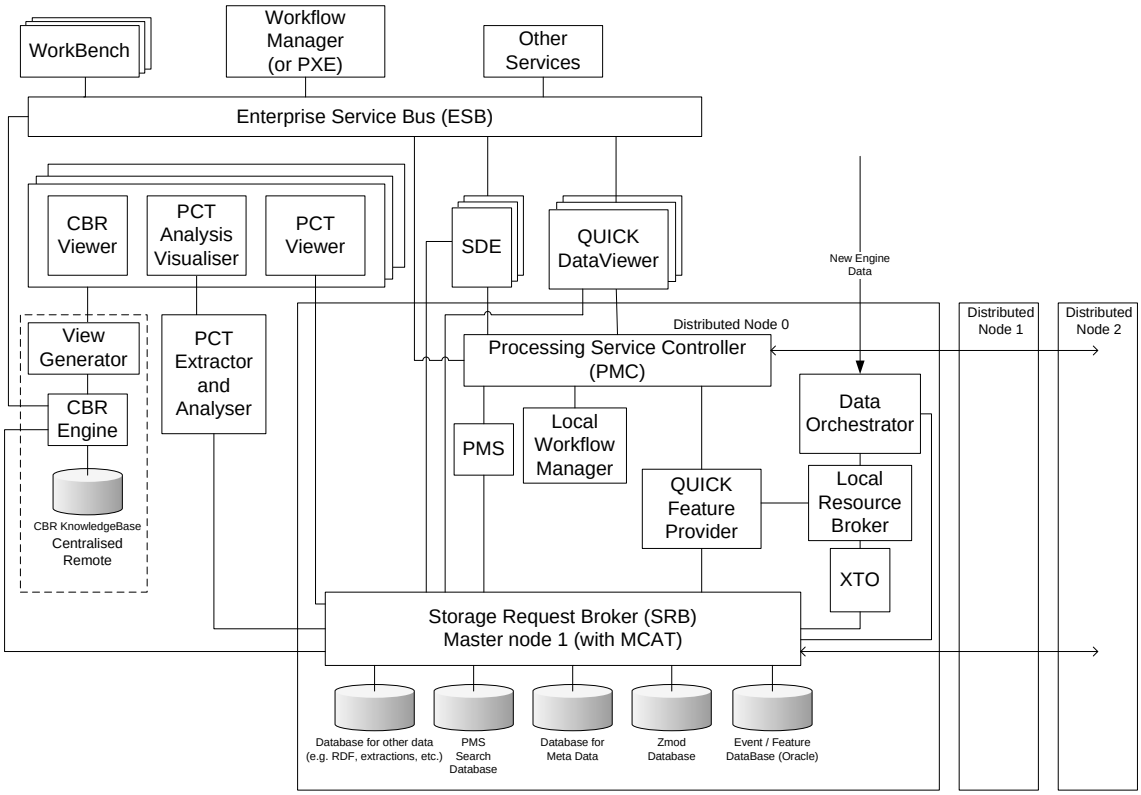

Figure 3 Overview of BROADEN Tool, Service and Data Architecture

4. Tool Interaction and Distributed Service and Data Management

This section describes the main middleware elements we have selected which allow the tools to interact and which facilite the management of distributed data and services. As shown in Figure 3, there are a number of tools that communicate with each other. These originate from different vendors (in our case Oxford Biosignals, York University, Sheffield University, Leeds University and Rolls-Royce). There are many other tools and services that could be added from other suppliers. The main challenge has been to allow these tools and services to communicate with each other without having to change the tools or services themselves, that is, to provide an open and extensible architecture. This is driven by the recognition that tool and services vendors are unlikely to want to alter their software just to fit into the proposed architecture, or to communicate with a new tool (in the project we have simulated this constraint). The simplest mechanism to solve this problem would be to implement adapters between all the services and tools that translate the communications between the tools. Unfortunately with a large number of tools and services the number of adapters would potentially be very large. This raises the need for a middleware layer to allow a configurable translation mechanism. This middleware must also address work orchestration issues such as starting, stopping, monitoring and dynamic workflow capabilities. Our work has investigated the Enterprise Service bus (ESB) and our own PMC to achieve this.

4.1 Enterprise Service Bus

An Enterprise Service Bus [7] typically provides XML messaging, XML transformation, intelligent routing, connectivity, support for service oriented architectures, etc. An instance of an ESB may be used within the architecture as a simple messaging and translation mechanism between the tools of the system. Tools are able to register with the ESB providing information about their type of service, their functions and the respective data formats. Translation specifications are provided to the ESB in a standard format and the ESB provides translation facilities on a tool pair basis and even on a function pair basis, if necessary.

The ESB keeps a registry of all connected tools, and routes messages between tools and translator components. Tools, once registered, might become unavailable, move to different locations or change their data formats without informing the ESB. Therefore a method of continuous verification and notification of each service will also be implemented.

Errors and exceptions are logged for an administrator to resolve problems. In addition to this, an informative message is returned to the client with information about the failure of any requests.

Workflow, for use in the Global Workflow Manager, can be expressed in BPEL4WS and in order to make the update of workflows more user-friendly, a GUI will be included in the Workbench. Our concerns with ESB are performance related, given its requirement to communicate primarily through XML schemas.

4.2 Process Management Controller

We have developed a complementary technology in DAME, which is termed Process Management Controller (PMC). PMC is a lightweight middleware application for managing services which act on distributed data in a generic manner and for providing high performance communication [6]. The PMC is independent of the data types and algorithms used and provided by the services.

The PMC service provides an interface that allows client applications (tools or other services) to manage services and retrieve results from a network of PMC nodes, without knowledge of how many nodes exist or where they are located. Each service operates on data held within its local node.

The network of PMC nodes is organised in a peer-to-peer network of independent nodes. When a distributed service activity is initiated at a PMC node, this node becomes the master for that activity. All other nodes are designated slaves for that activity. The master node replicates the service request to all the slave nodes. Each node then uses the appropriate local service to perform the activity and the slave nodes return their results to the master node, where they can be collected by the client application. Subsequent or parallel service activities can be initiated at any node in the network.

On start-up each PMC node reads its local configuration file which contains parameters that adequately describe its state, including its *seed* PMC node. The seed node is a key aspect to the PMC network initialisation process. In an empty network, one initial node is designated the seed node, and this node is started first. All

subsequent nodes can use this node as their seed. At later points in time, it may be desirable to use additional seed nodes. The initialisation process for both new and existing nodes will be synchronised and measurable. The PMC nodes will form a peer-to-peer network where the nodes are loosely coupled and only form connections in response to user requests.

As new nodes are added and they register with all other nodes in the network, each node will maintain a persistent register of known nodes.

It may also be necessary to temporarily remove nodes from the network e.g. for maintenance. In these cases the node will not be logically removed, so as to maintain the data coverage statistics for search operations. When these nodes are restarted, they will need to ensure that their node lists are correct and up to date. If a node is to be permanently removed from the PMC network, then it must use the de-register method on each PMC in its contact list.

4.3 PMC v ESB

We currently see the capabilities of PMC and ESB being complementary; one offers flexibility at the cost of performance, the other reduces the flexibility but give a gain in performance. An analysis is underway to measure these issues within the BROADEN application. We see the eventual possibility of migrating the functional aspects of ESB that we determine as essential into PMC, once these have become clear through trial deployment.

4.4 Distributed Data Management.

As the data within the system may be distributed between many compute nodes it is necessary to provide a mechanism to virtualise the storage across the nodes. Our current choice is the SDSC Storage Request Broker (SRB) [8]. SRB is a tool for managing distributed storage resources from large disc arrays to tape backup systems. Files are entered into SRB and can then be referenced by logical file handles that require no knowledge of where the file physically exists. A metadata catalogue is maintained which maps logical handles to physical file locations. Additional system specified metadata can be added to each record.

The SRB can operate in heterogeneous environments and in many different configurations from completely stand-alone, such as one disc resource, one SRB server, and one MCAT, to completely distribute with many resources, many SRB servers, and completely federated MCATs. In this configuration, a user could query their local SRB system and yet work with files that are hosted remotely.

When data is requested from storage, it is delivered via parallel IP streams, to maximize network throughput, using protocols provided by SRB. In current implementations, a single MCAT is deployed inside a Postgress database. Real world implementations would use a production standard database, such as Oracle or DB2. With this setup, each nodes's SRB installation is statically configured to use the known MCAT.

The use of a single MCAT provides a single point of failure that would incapacitate the entire distributed system. To alleviate this, a federated MCAT would be used, allowing each node to use their own local MCAT. In large systems, this would provide a performance benefit, as all data requests would be made through local servers. However, this would require an efficient MCAT synchronization process.

5. The BROADEN Tool, Service and Data Architecture Implementation

The previous two sections described the generic architecture. This section describes the actual tools and services used when the architecture is applied to the BROADEN project. Figure 3 shows the architecture populated with the BROADEN tools and services.

The visualisation tools provided are:

- Case Based Reasoning Viewer [9] including Performance Curve Test (PCT) Viewers.
- Signal Data Explorer (SDE) is a visualisation tool used to view vibration spectral and extracted time series data [5, 10] and performance data. It also acts as the front end to the high performance pattern matching services.
- QUICK Data Viewer is a visualisation tool which allows a user to view spectral data, extracted features and events occurred [4].

The data repository at each node is responsible for storing all raw engine data along with any extracted and AURA encoded data. Each node is populated with:

- The Pattern Match Service (PMS) which is based on Advanced Uncertain Reasoning Architecture technology [11] and provides high performance pattern matching for use with the SDE or in workflows.

- The QUICK Feature Provider service which provides features and events to a client such as the QUICK Data Viewer.
- The XTO (eXtract Tracked Order) service which extract specific time series from the raw spectral vibration data.
- The Data Orchestrator is the Data Loading Service and is responsible for "cleaning" and storing all raw engine data.

Also included in the BROADEN architecture is a centralised CBR Engine and Knowledge Base [9].

6. Future work

The generic architecture has been developed to be application neutral. It will be implemented for the BROADEN application during the course of the project. This will allow us to assess the strengths and weaknesses of PMC and ESB for the task.

Future work will include:

- The testing and demonstration of the generic architecture in the BROADEN application domain.
- The testing and demonstration of the generic architecture in other application domains.
- The introduction of fault tolerance as appropriate (inter and intra node).
- The exploration of fault tolerance techniques within the ESB.
- Performance testing would be helpful to discover issues early in the process.

7. Conclusions

This work builds on the tools and services developed during the DAME project. A generic architecture has been developed, which integrates:

- Geographically distributed nodes containing data repositories and services.
- Geographically distributed users.
- Legacy Tools.
- Purpose designed tools.

It is a Grid based architecture used to manage the vast, distributed, data repositories of aero-engine health-monitoring data. In this paper we have focused on the use of a generic architecture to enable the general concept to be used in other application areas and with varying degrees of legacy systems and designs.

The middleware elements are generic in nature and can be widely deployed in any application domain requiring distributed tools and services and data repositories.

8. Acknowledgements

The work reported in this paper was developed and undertaken as part of the BROADEN (Business Resource Optimization for Aftermarket and Design on Engineering Networks) project, a follow-on project to the DAME project. The BROADEN project is part-funded via the UK Department of Trade and Industry under its Technology Innovation Programme and the work described was undertaken by teams at the Universities of York, Leeds and Sheffield with industrial partners Rolls-Royce, EDS, Oxford BioSignals and Cybula Ltd.

9. References

[1] J. Austin et al., "Predictive maintenance: Distributed aircraft engine diagnostics," in The Grid, 2nd ed, I. Foster and C. Kesselman, Eds. San Mateo, CA: Morgan Kaufmann, 2003, Ch. 5.

[2] I. Foster and C. Kesselman, Eds., The Grid, 2nd ed. San Mateo, CA: Morgan Kaufmann, 2003.

[3] Data Systems and Solutions Core Control™ technology. http://www.ds-s.com/corecontrol.asp.

[4] A. Nairac, N. Townsend, R. Carr, S. King, P. Cowley, and L. Tarassenko, "A system for the analysis of jet engine vibration data," Integrated Computer-Aided Eng., vol. 6, pp. 53–65, 1999.

[5] D. O'Connel, "Pilots predict air chaos," The Sunday Times, p. 3.3, Mar. 14, 2004.

[6] J. Austin et al., "DAME: Searching Large Data Sets Within a Grid-Enabled Engineering Application". Proceedings of the IEEE, vol. 93, no. 3, March 2005.

[7] David A. Chappell. "Enterprise Service Bus – Theory in Practice". O'Reilly. ISBN 0-596-00675-6.

[8] SDSC Storage Request Broker [Online]. Available: http://www.sdsc.edu/srb/

[9] M. Ong, X. Ren, G. Allan, V. Kadirkamanathan, H. A. Thompson, P. J. Fleming (2004). "Decision support system on the Grid". Proc Int'l Conference on Knowledge-Based Intelligent Information & Engineering Systems, KES 2004.

[10] Martyn Fletcher, Tom Jackson, Mark Jessop, Bojian Liang, and Jim Austin. "The Signal Data Explorer: A High Performance Grid based Signal Search Tool for use in Distributed Diagnostic Applications." CCGrid 2006 – 6th IEEE International Symposium on Cluster Computing and the Grid. 16-19, May 2006, Singapore.

[11] The AURA and AURA-G web site, Advanced Computer Architectures Group, University of York, UK. http://www.cs.york.ac.uk/arch/NeuralNetworks/AURA/aura.html.