Developing New ATM Network Management Systems with External Partners

A White Paper

EUROCONTROL DNM

May 2012

EUROCONTROL

EXECUTIVE SUMMARY

This White Paper describes how EUROCONTROL's Directorate Network Management (DNM) has addressed the issue of becoming more open in the development of its Network Technical Systems, being able to work in partnership with external organisations and to integrate new software developments, while continuing to master the full architecture.

This paper is primarily for ANSPs who would like to know more about how this has been achieved and the opportunities it offers.

Over the last few years, the Network Technical Systems have been moving towards a service approach in which the services provided by the different backend systems are accessed via a service layer and can be combined with other services to offer added value to the user.

Recently, DNM has developed a framework that allows new systems to be developed by partners outside DNM, while using the existing infrastructure and support facilities. This is called the Java Backend Server (JBS) framework and its main features are described in this paper.

The JBS framework offers standard ways to deal with generic IT system requirements, such as exception handling, load balancing, data persistence, monitoring and deployment. It allows new backend systems to be hosted seamlessly within the architecture and the DNM infrastructure. In addition, making use of the standard functionality provided can bring significant savings in development costs and leaves the JBS developer free to concentrate on the specific business functionality.

The paper provides some information about the real-life use of this framework – for the development of the Call Sign Similarity Tool (CSST) – and a potential new opportunity offered by JBS: service hosting.

For further information, please contact:

Mr Jean-Pierre Aiguier,
Head of Network Technical Systems Division, DNM, EUROCONTROL
jean-pierre.aiguier@eurocontrol.int

INTRODUCTION

If you find it difficult to master the architecture of your software developments in partnership with external organisations or find it painful to integrate such developments into your environment, you might be interested to know how EUROCONTROL's Directorate Network Management (DNM) has addressed the problem for its Network Technical Systems.

DNM, like many other organisations, has been confronted with the need to work with external partners on its software system developments, to be able to cope with the demand for new services. We have often been faced with these questions:

- How can we master the architecture of the network management systems when they are developed by partners outside DNM?
- How can we minimise problems when integrating external developments into our environment?
- How can we minimise the impact on operational support?
- How can we support our partners who would like to develop services while making use of our infrastructure?

The solution to these questions (and to the challenge to move the Network Technical Systems towards new technologies) was to adopt a service approach, resulting in the development of a service-oriented framework that allows new systems to be developed by partners outside DNM, while using the existing infrastructure and support facilities to provide network management services in two complementary ways:

- **B2C (business to client) - via the Network Operations Portal (NOP)**
  which provides a consolidated view of the different aspects of the Network Operations Plan and gives access to a set of services to support the associated planning processes (see https://www.public.cfmu.eurocontrol.int/PUBPORTAL/gateway/spec/index.html); and
- **B2B** (**business to business)- via NOP web services**,
  which provides access to both data and services via a system-to-system interface, allowing customers to call these services and exchange corresponding information from their own systems directly; example B2B services are Flight Plan Validation, Route Generation and eAIM (which provides access to the consolidated European Airspace Use Plan).

BACKGROUND

The original Central Flow Management Unit's (CFMU) systems were developed in the early 90s, using the technologies available at the time. One of the key strategies of the DNM in support of business-IT alignment and cost-efficiency is service orientation. DNM has adopted service-oriented principles not only at the software architecture level (SOA[1]), but at other levels too: service orientation impacts skills, processes, organisation, documentation and culture.

Adopting a service-oriented approach for software architecture cannot be achieved from one day to the next, especially when there is a set of very complex systems that needs to be kept in operational use and upgraded, while addressing the overall architecture evolution. The strategy that we developed had to take into consideration the existence of our current systems; it is based on a step-wise approach, resulting in a layered architecture (see Figure 1 below):

[1] Service Oriented Architecture

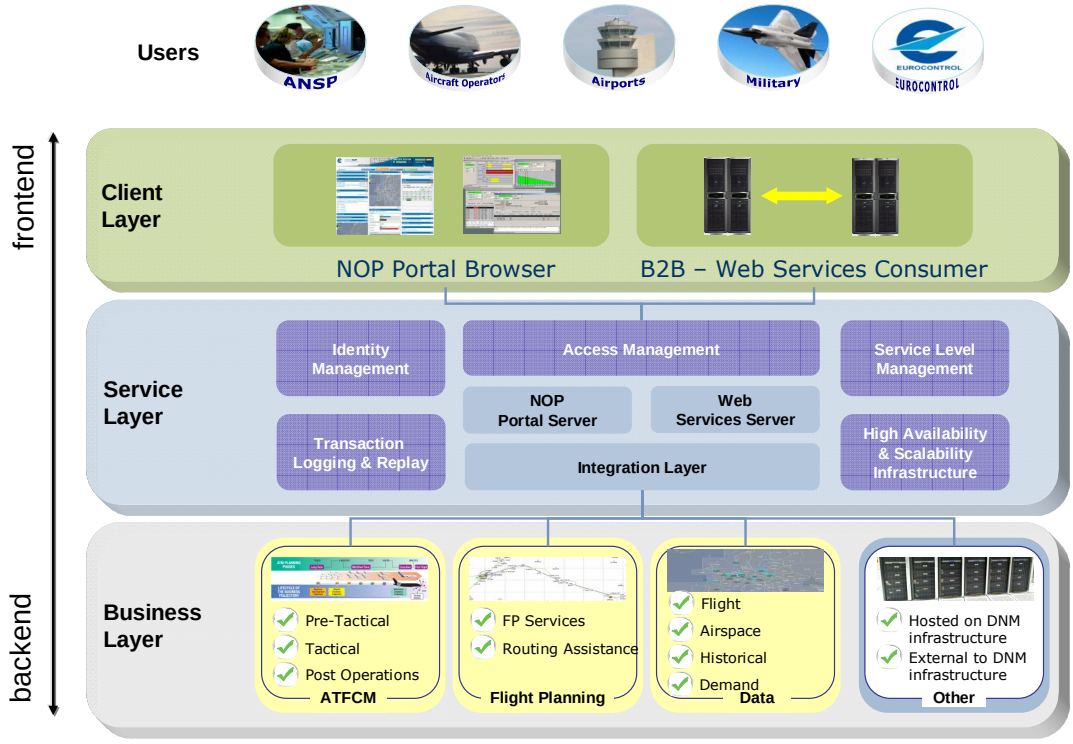

Figure 1: Overview of the Layered Architecture

At the frontend, the Users access the Network Technical Systems through the client layer which provides services in two complementary ways. The NOP Portal provides services directly to end users via a website[2]. The NOP B2B web services provide an interface that can be accessed directly by the end users' systems, allowing them to build their own applications.

At the backend, the business layer contains the underlying systems providing the business data and functionality needed to provide the services to the users. It includes existing systems for Air Traffic Flow and Capacity Management (ATFCM), Flight Planning and data storage. One objective of the SOA framework introduced by DNM is to enable other systems to be developed in partnership with external organisations, so that they can be integrated into the architecture alongside these existing systems.

The service layer is part of the SOA approach that DNM is following; it is described in the next section.

MOVING TOWARDS SOA

As stated above, DNM has taken a step-wise approach to the introduction of service orientation.

The **first step** was to implement a service layer on top of the backend systems that make up the business layer. This service layer, developed by DNM, is the cornerstone of the service-oriented architecture and the interoperability with other systems, and is in line with the evolutions necessary for SWIM[3] . This step was started in 2004 and now provides:

- **Service Integration:** The services provided by the different backend systems in the business layer are accessed from the service layer and potentially combined with other services to offer added value to the user. For example, showing on a single geographical map: the trajectory of a

[2] More specifically, a web-based application desktop.
[3] System Wide Information Management

filed flight plan obtained from one system; the NOTAMs[4] affecting that flight obtained from another system; and weather information obtained from an external weather provider.

- **Formal modelling of services:** The service layer uses a model-driven approach where the service interfaces are formally modelled and parts of the software are automatically generated from these models.
- **Service management:** Ensuring that the different SLAs[5] we have with different customers are respected in a cost-efficient way. It includes: service monitoring, measuring and reporting, logging, replay, load balancing and overload protection.
- **Security:** Providing authentication, single sign-on, authorisation, confidentiality, user access segregation and identity management.

The **second step** was to establish a framework for the development of backend systems, in order to enable service-orientation in the overall architecture. This step was achieved in 2009 with the development by DNM of the Java Backend Server (JBS) framework (see below).

The **third step** was to develop new backend systems using this framework. This was started in 2010, with the development of the first backend server developed in Java within a service-oriented architecture: the Call Sign Similarity Tool (CSST). The CSST is also our first backend system to have been developed by an industrial partner.

The rest of this white paper concentrates on the JBS framework (step 2), but also provides some information on its first practical use (step 3).

ENABLING SEAMLESS INTEGRATION

The second step in the DNM introduction of service-orientation is about providing a framework to enable seamless integration of developments by external partners: the JBS framework.

Overview of the Java Backend Server (JBS) Framework

DNM has developed the Java Backend Server (JBS) Framework to address two objectives:

- to allow the development of an overall service-oriented architecture; and
- to allow the development (in Java) of backend systems by external partners, while operating that software within the DNM deployment infrastructure (see below).

These backend systems are called Java Backend Servers because:

- they are written in Java;
- they are “backends” in the sense that they do not offer any graphical user interface, i.e. do not support direct interactions with the user (this is provided in the client layer); and
- they are “servers” in the sense that they contribute to the service rendered via one or more client applications built on top of the JBS.

[4] Notice to Airmen
[5] Service Level Agreement

The JBS framework is made up of 3 key elements:

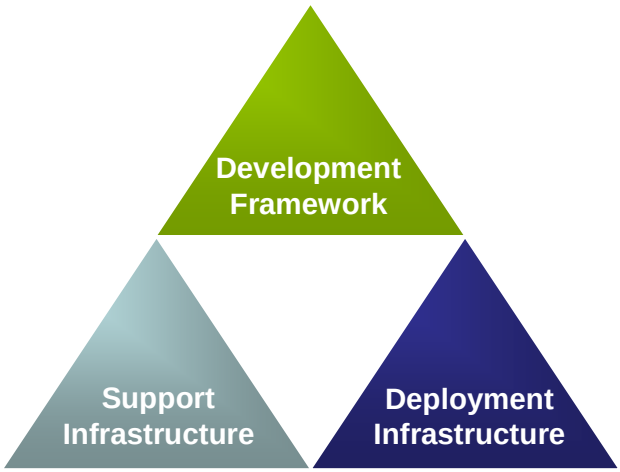

Figure 2: JBS Framework

- The **development framework** can be compared to J2EE in the sense that it offers standard ways of dealing with generic IT system requirements (such as initialisation, exception handling, load balancing, high availability, concurrency management, configuration, logging, data persistence, monitoring and deployment). J2EE is generic, but in practice is heavy to use and not rich enough for our purposes. The JBS development framework created by EUROCONTROL is specifically adapted to guarantee an overall architecture alignment and the smooth integration of new JBS software into the Network Technical Systems architecture. It is also a means for standardising the support activities associated with the new backend.

  The development framework comprises:

  - the **JBS Developer's Guide**, a document explaining the different principles, policies and interfaces to be applied to design, develop, test and package a backend system (JBS), so that it can be smoothly integrated into its target deployment environment in DNM;
  - the **JBS DK**[6], which allows the developers to compile, run and test the backend system at their own site and to use their own configuration management tools; it also allows the tests to be easily re-executed at EUROCONTROL;
  - **Training material** to help developers understand and use the JBS framework.

- The **deployment infrastructure** ensures adherence to the Network Technical Systems deployment standards and enables the backend system to use the NTS run-time environment and Service Layer. The infrastructure provided is mission-critical (i.e. has high availability and multi-site replication).

- The **support infrastructure**, ensures smooth integration with the existing monitoring, support and contingency processes and 24/7 technical help desk.

The JBS framework is already in place and is being used to support backend system development in two major areas: CSST (Call Sign Similarity Tool – see next section) and ADR (Airspace Data Repository). It enables Java systems to be developed completely by external partners or for the development to be moved between in-house and external teams. It is the means to ensure that the new backend systems can be hosted seamlessly within the architecture and the DNM infrastructure, where they can be used to support the delivery of new or improved services. By making use of all the standard functionality provided by the service layer, the JBS developer is left free to concentrate on the specific business functionality, which brings cost-benefit in the development.

[6] Java Backend Server Development Kit – includes a "compiler" (actually a tool that helps the developer in generating, compiling the sources and packaging the results consistently), some Java code within which the JBS must run, and a testing framework written in Java (with test data in XML).

The Annex to this paper provides more information on the main principles and concepts behind the JBS development framework summarised here:

- **managed nodes** provide a clear separation between functional aspects (the specific purpose of the node, i.e. the backend business logic, which is in the Java software developed – the JBS itself) and non-functional aspects (i.e. infrastructure that is common to all nodes, see Figure 3 below);
- **model-driven architecture** for service interface and data type definition; and
- a **specialised test framework** that enables the testing of the JBS.

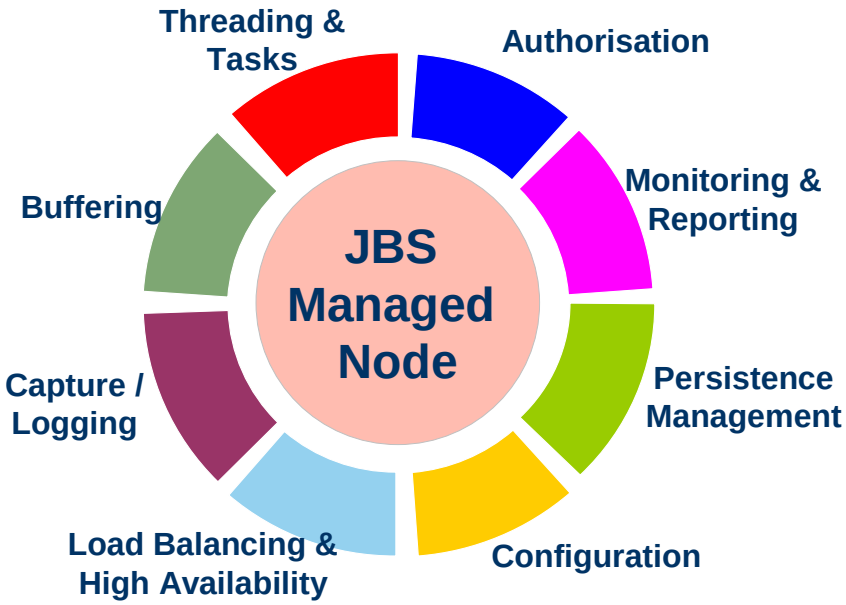

Figure 3: Generic Functionality Provided to a JBS Managed Node[7]

JBS FRAMEWORK IN PRACTICE: REAL-LIFE EXAMPLES

The JBS framework has already been validated as it was used by an external company to develop the first Java Backend Server for EUROCONTROL's services. This JBS is the backend system that implements the business logic part of the Call Sign Similarity Tool (CSST).

The industrial partner development team worked at their premises in Toulouse and this first experience of working with an external partner allowed DNM to improve the JBS development framework, particularly with respect to its documentation and the associated training. The ability to integrate and deploy externally produced Java applications was also improved through this first implementation.

In addition, the JBS framework is also being used for in-house developments, such as in the context of the ADR (Airspace Data Repository).

[7] The Annex provides a brief description of each of the elements on the diagram.

JBS OFFERS A NEW POTENTIAL OPPORTUNITY: SERVICE HOSTING

The JBS framework, however, offers more than just the potential to be more open with respect to where new backend developments are performed.

In order to serve its customers and to deliver business services with the necessary quality of service, DNM already has in place a mission-critical infrastructure that includes 24/7 monitoring, support, contingency and technical help desk. Until recently, this infrastructure was only available for the IT services developed in-house. The introduction of the JBS framework now provides new opportunities for service provision. It enables backend systems developed by external partners to be hosted on the DNM infrastructure, ensuring their seamless integration into our architecture.

These systems can still be managed by the external partner (including further development and maintenance). In addition, the ownership of the external applications can remain with the external partner: oversight, functional specification, user relations, third party contract management, etc.

This hosting approach offers significant economies of scale in terms of infrastructure, deployment and support costs. It also makes it possible to integrate the services offered by the external system into the network applications and make them available to the network users, either via the NOP Portal or via B2B Web Services.

CONCLUSION

In this paper we have explained how EUROCONTROL's Directorate Network Management (DNM) has addressed simultaneously the issues of architecture evolution and support for software development by external partners for the Network Technical Systems.

The architecture approach DNM has put in place allows us to transition to a full service-oriented architecture and to exploit a mixture of new and existing systems through a service layer. It also allows us to offer new client services by integrating services provided by different backends.

In this context, the Java Backend Server (JBS) framework enables us to work with external partners in developing new backend systems that integrate smoothly into the NTS architecture and benefit from the service infrastructure. As a bonus, it has also opened up opportunities for our stakeholders In terms of hosting systems that can be used to provide new or improved services to the users of the ATM network.

For further information, please contact:

Mr Jean-Pierre Aiguier,
Head of Network Technical Systems Division, DNM, EUROCONTROL
jean-pierre.aiguier@eurocontrol.int

ANNEX – MORE INFORMATION ON THE JBS MAIN CONCEPTS

Provision of Generic Functionality - Managed Nodes

The JBS (i.e. the actual piece of Java software developed to implement the new, service-oriented backend system) is executed in one or more "managed nodes", which are processes that take on a number of responsibilities for generic functionality on behalf of the JBS. The generic functionality is identified in Figure 3 above, and further described below. The JBS *must not* implement any of this common functionality, since to do so would not only imply duplicate functions but would also probably provoke defects in the overall service implementation.

- **Authorisation** – Role-based authorisation is provided; it determines whether a user is authorised to call a JBS method or not.
- **Monitoring and Reporting** – All the activities related to the Service Level Agreement (SLA) implementation and monitoring are provided.
- **Persistence Management** – Regarding storage, the JBS has access to NTS storage, which is made up of: a local file system (used for transient persistence operations that do not need to be shared with other nodes); a shared file system; and a shared database (Oracle).
- **Capture / Logging** –The JBS is provided with a feature to capture and persist the events that occur within a thread lifecycle (e.g. input arguments and return values, exceptions). This is used to support service reporting as well as testing and problem investigation.
- **Configuration** – Within the NTS architecture, a configuration is a part of a formal deployment plan, and therefore obeys some rules. For example, the JBS configuration is typed - it is not a collection of type-free name/value pairs, but an object defined in the IEM (Internal Exchange Model – see the next section). The JBS configuration also comes with an XML representation so that it can be directly modified online by a human being, without going through a dedicated application.
- **Load Balancing & High Availability** – The JBS can be deployed in many nodes. The Service Layer achieves load balancing between these nodes.
- **Threading & Tasks** – Two kinds of threads are identified:
  - Request threads: used to handle requests issued from outside the JBS instance. These are the threads used by the node to dispatch requests to the JBS instance.
  - Controller threads: used to handle calls issued from inside the JBS instance.

  It is the node responsibility to manage the creation and deletion of threads; the JBS is therefore strictly forbidden to do so.
- **Buffering** – Managed nodes rely extensively on multi-threading. When a request is to be dispatched to a JBS instance, the node buffers it if there is no thread available at the moment in the thread pool, and dispatches it to the JBS otherwise.

Details of the Java classes, interfaces and methods relating to these functions and rules to be followed are given in the **JBS Developers' Guide**.

Model-Driven Architecture & Code Generation

The JBS framework makes use of a model-driven approach for handling the interfaces between the JBS and the rest of the architecture (indeed between all managed nodes). The interfaces offered by the newly developed JBS must be described in the **Static Service Model (SSM)**, which is XML based.

All types exchanged via these interfaces are defined via the **Internal Exchange Model (IEM)**, which forms part of the SSM. There is a "Common" part of the IEM providing all exchanged types that are not specific to a given JBS (e.g. DateTime) and which is made available to developers in the JBS DK. The JBS developer adds to the IEM all exchanged data types that are specific to their JBS, according to the meta-model and rules described in the JBS Developer's Guide.

As well as modelling the interfaces of the nodes, the SSM also models: the input and output validation rules; the roles that different users can play; errors; log events; etc. It is then used as the input for code generation tools.

From the SSM, the JBS development framework automatically generates the necessary artefacts, including Java types, but also much more. For example, the JBS developer does not need to take care of communication, as it is all automatically generated from the model. The developer manipulates only Java objects; the node framework takes care of the communication details.

JBS Test Framework

As one of the objectives of the JBS framework is to support software development by external partners, it includes a corresponding JBS Test Framework, developed by DNM, which allows:

- definition of **test data** as data model values (SSM model values that correspond to the equivalent SSM model types);
- definition of **test scenarios** to capture data from the JBS under test; and
- definition of the actual **tests** to be computed from the captured data and the comparison with the expected results.

The JBS test framework is used in two steps:

- first to **execute scenarios**, i.e. to run a number of sequential and/or concurrent actions with the JBS (e.g. database initialisation, method calls) - both the action identifications and the results are captured for further analysis; and
- secondly, to **run tests**, i.e. to process the captured results, compare the actual results against the expected ones, and report.

This specialised test framework enables the testing of the JBS at the developer's premises (FAT[8]) and at EUROCONTROL premises (SAT[9]) with minimal integration effort.

[8] Factory Acceptance Test
[9] System Acceptance Test

Filename: JBS White Paper v15 (2)
Directory: C:\Documents and Settings\cszabo\Local Settings\Temporary Internet Files\OLK168
Template: C:\Documents and Settings\cszabo\Application Data\Microsoft\Templates\Normal.dot
Title: Developing New ATM Network Management Systems with External Partners
Subject:
Author: CDL
Keywords:
Comments:
Creation Date: 31/08/2012 10:17 AM
Change Number: 3
Last Saved On: 31/08/2012 10:17 AM
Last Saved By: CDL
Total Editing Time: 0 Minutes
Last Printed On: 31/08/2012 1:18 PM
As of Last Complete Printing
Number of Pages: 10
Number of Words: 3.260 (approx.)
Number of Characters: 18.584 (approx.)