HIGH AVAILABILITY TELEMATIC MANAGEMENT SYSTEM

Tiago Dias[1], Jorge Lopes[1], Gonçalo Abreu[2], Eduardo Lopes[3]

1: Brisa Auto-estradas de Portugal
Quinta da Torre da Aguilha, 2875-599 São Domingos de Rana, Portugal
e-mail: {tiago.dias, jlopes}@brisa.pt, web: http://www.brisa.pt

2: MakeWise, Engenharia de Sistemas de Informação
Rua Dr. Francisco Sá Carneiro nº. 4 R/C Esq. 2500-206 Caldas da Rainha
e-mail: goncalo.abreu@make.com.pt, web: http://www.make.com.pt

3: Armis, Sistemas de Informação
Rua Eugémio de Castro, nº 248, Esc. 144, 4100-225 Porto, Portugal
e-mail:eduardo.lopes@armis.pt, web: http://www.armis.pt

ABSTRACT

An efficient use of ITS resources depends on the information and communications architecture that enables the optimal use of technology, information and, increasingly, services, available across the variety of ITS applications and systems. This is especially true for quality of service requirements, namely availability.

The current article focuses on architectural concerns in the development and deployment of the VMS operation component of Traffic Atlas. This component is cross-cutting in regards to the multi-layer service oriented architecture of Traffic Atlas; in order to simplify our approach a VMS-centric approach is taken at each layer.

KEYWORDS: Telematics, High Availability Solutions, ITSIBus, Service Oriented Architectures, Traffic Management

1. INTRODUCTION

The rapid evolution of technologies, competitive market and lack of standardization has given rise to the creation of products which generally prohibit the integration of legacy systems without significant reengineering [1]. Because of this, the various systems which together provide overall traffic and toll plaza management are normally operated independently from dedicated workstations. This however inhibits the operator's decision-making abilities, especially in real time situations, impacting on the end service level provided.

In their development of Traffic Atlas, Brisa launched a process whereby data from disparate systems is collected, harmonised, aggregated and enhanced to form a single, optimised interface. The innovative aspect of Traffic Atlas lies in achievement of the real time integration of fragmented data streams sourced from fundamentally differing systems possessing varied operational functionalities. The challenge was heightened by the diversity of these systems in functionality, application (including CCTV cameras, emergency systems, weather stations and toll plaza management systems) and generation. The Traffic Atlas system represents a novel application of various ICTs (information and communication technologies) which build a common platform middleware from which the transmission, receipt and recording of data, voice and video information is integrated and displayed in a standard format on a map-based user interface.

2. THE TRAFFIC ATLAS SOFTWARE PRODUCT LINE

Brisa Auto-Estradas de Portugal, founded in 1972, is the largest Portuguese motorway operator and an important player in the traffic sector in Europe. Brisa currently operates, on a concession basis, a network of 11 motorways constituting the main Portuguese road links, connecting the country from north to south and from east to west. The total length of motorways in the network is more than 1100km. Brisa owns various companies specialising in motorway services aimed towards increasing in its own operating efficiency and improving the quality of the service provided to customers. The company is known internationally for the deployment of the automatic pass-through toll systems.

Traffic Atlas, shortly named "Atlas", is a web-based product for motorway operation and management. It's designed for use in a wide area of operation-related environments. It has been developed by Brisa for internal usage and is currently the main working tool for Brisa's control room (operations and tolling), maintenance teams, operation managers, external law forces and call-center.

The central design in Atlas has been that of a software product line. Software product lines are set to create a managed set of components meeting the general requirements of a market segment, instead of developing a customer-centric application. This design enables Atlas to easily adapt to new requirements and configuring solutions for different deployment scenarios. Support for a different CCTV product or integrating a new kind of road-side equipment is easily achieved in Atlas due to the pervasive usage of two essential design-patterns: Provider/Adapter and Observer [2]. The Provider pattern is used where Atlas should not be bound to a specific external system type; examples are Providers for video-walls, CCTV or Variable Message Signs (VMS). It works basically by defining an interface that all providers must follow and, by means of a configuration point, the actual provider is designated for each element. The Observer pattern is used for decoupling cross-cutting actions from operational code. This patterns works by defining an event which is triggered throughout the code, then handlers for this event are registered, each doing a specific task, some logging actions or alarms to database others to change the state of a particular equipment or generate a human alert. The events defined by these means are available for future usage for extensions and don't require changes to existing code. We are also using a Hook Operations [3] design-pattern when the action taken outside the main code has to return an effect to the original code (namely when validations are required). A more generic approach could be obtained using aspect oriented programming [4]. Our Provider design-pattern approach is consistent with the higher-level of multi-layer service orientation in Atlas. Typically for each Atlas component there is a web service that is available for the use of the interface, internally this service is composed of distinct data-access and logic layers and may also consume an external service by means of a provider.

The Atlas interface is fully web-based in order to avoid the need for other client-side software other than a standard web-browser. This feature simplifies deployment of new versions as well as makes it simpler to provide redundancy and fault-tolerance. Additionally, in order to achieve a richer and more responsive client interface without requiring specialized client software, Atlas is making an increasing use of Ajax (Asynchronous Javascript And XML). When compared with most ITS interface clients (which are GIS based) Atlas has a different approach for its' mapping interface. Instead of basic geographic maps, Atlas also provides cognitive mapping [5] which are schematic representations of a motorway network. The schematic nature of such representations makes it easy to capture user attention to specific road-elements, incidences or equipment. Due to its' simple geometric nature, this approach is

also more appropriate for implementation as a web-based interface than a GIS based approach.

3. THE ITSIBUS ARCHITECTURE

ITSIBus - Intelligent Transportation Systems Interoperability Bus [6] is a Service-Oriented Architecture originally developed in ISEL (Instituto Superior de Engenharia de Lisboa). Its main motivation is to facilitate integrated solutions by defining a technologically independent, open standard, of an architecture that focuses on systems and services.
Brisa's toll plazas are managed by a solution that was developed according to ITSIBus and is in production since 2004. The elected technological binding for this product was the Java framework, using Jini for the service-oriented implementation.

4. TELEMATIC MANAGEMENT SYSTEM

Telematic Management System (TMS) is the Traffic Atlas solution for the operational management of different types of telematic equipments. The main objective was to evolve from having heterogeneous and monolithic systems, unconnected and unaware of each other, to a comprehensive and integrated solution, based on open standards, to which developers and vendors can seamlessly contribute. The whole system was developed according to ITSIBus (open) standards, and currently is managing the entire set of Brisa's network of VMS equipments, weather stations and other equipments already in the works (visual incidence detection agents, etc.).

Physical devices are integrated in the ITSIBus architecture through the use of adapters – software components which abstract the details of the device's communication mechanism and are an ITSIBus service in their full right. These adapters are also a concretization of the Adapter design pattern seen in [2]. Adapters are developed for each particular VMS model/vendor. An adapter represents a VMS in the system, providing its functionality through a service; while originally, this functionality would only be available as a proprietary (lower level) protocol.

The VMSs (in the form of their adapters) are managed by a Telematics Management Server which constitutes the main orchestrator of the entire system, processing requests from external systems and performing management and maintenance operations on the VMSs. The first approach towards the TMS architecture included an instance of an adapter for each VMS, running in dedicated hardware located on the field, near each VMS.

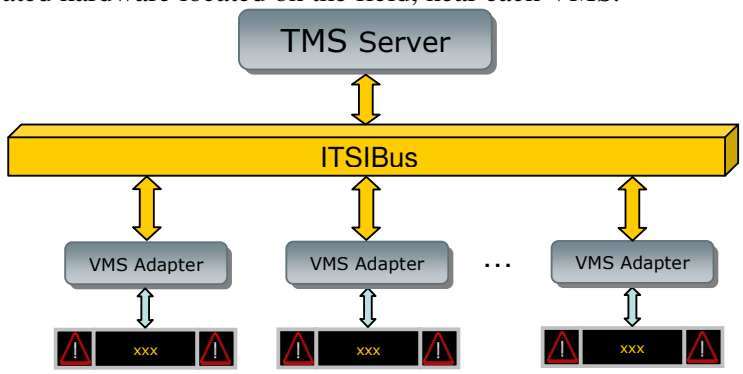

Diagram 1: Initial TMS Architecture

With the objective of lowering deployment and maintenance costs, adapters running near each VMS were moved to a central location inside TMS by virtualization as a pool of the adapters for on-demand usage. The adapters run within the TMS Server context and, every time there is a need to communicate with a VMS, an adapter is obtained from the pool and properly configured to establish a connection with the VMS.
Nevertheless, in case a particular VMS solution doesn't support a networked communications channel for its interaction protocol, the local adapter solution can coexist in TMS with this centralized adapter pool solution.

Shifting our view from VMSs to the central system, the entire TMS system is a distributed application comprised of several sub-systems. Bellow is a schematic representation of the entire system:

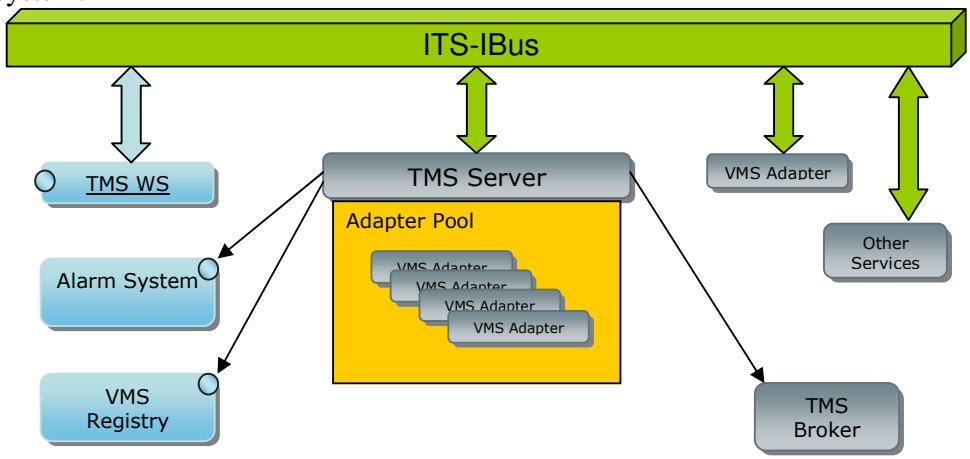

Diagram 2: Current TMS Architecture

As stated before, the **TMS Server** is responsible for managing several VMSs, implementing the lower level business logic and processing operational requests from other systems. It contains a pool of VMS adapters that provide communications with physical VMSs.
Amongst other things, the Server is responsible for detecting VMS content changes; placing default messages in a VMS; periodically adjusting the VMS's internal clock; monitoring hardware alarms.

The **TMS WS** component provides the ITSIBus TMS service functionalities through Web Services. It is a gateway that non-ITSIBus systems can use to interact with the TMS. Note, however, that the TMS WS communicates with the TMS through ITSIBus with all the inherent advantages (service location, redundancy, etc).

The TMS Server starts without absolutely any knowledge about the VMSs present in the network. One could find all the VMSs through an auto-discovery process, as defined in the ITSIBus standard, if the adapters were standalone services. However, since the adapter pool approach had obvious advantages in this particular scenario, a **VMS Registry** was implemented with the purpose of supplying information about the VMSs installed in the network. It is inquired as part of the VMS Server start-up process, supplying it with information about every VMS, such as: IP Address, Port Number, Default Content Message, Topology and Supplier (to choose the correct adapter implementation). In order allow adding new VMSs during TMS runtime, when there is a request regarding a VMS which is yet unknown, TMS inquires the Registry to obtain the new device details.

The **Alarm System**, available through Web Services, allows the delivery and notification of Alarms and Events generated by TMS. The processing and application of higher-level business rules in response to an event or alarm is handled by Traffic Atlas alarm module. A Java based JMX monitoring platform will be introduced in future TMS versions to allow for a more integrated and open solution in regards to the ITSIBus standard.

Given the architecture presented this far we can see that there is no significant impediment for having several TMS Server instances running at the same time, eventually each managing a different set of panels. But what about having more than one Server instance managing the same set of panels, for redundancy purposes? This is also a possible scenario with no drawbacks other than the fact that each instance would be monitoring the whole set of VMS, causing unnecessary network traffic since they would be doing repeated work.
To address this issue the **TMS Broker** was conceived. The Broker provides a lease to a Server, allowing it to be monitoring the VMSs. The first Server to get the lease is responsible for monitoring, while subsequent Server instances are denied the lease. Thus, only one Server can be monitoring at a given time. This Server must periodically renew the lease, otherwise (ex. if the server goes down) it becomes available again to supply to another Server.

5. VARIABLE MESSAGE SIGNS SUPPORT IN TRAFFIC ATLAS

Atlas works as an intelligence layer over the VMS. It takes a flat panel and turns it into a powerful operational tool by giving the VMS extended functionalities, both in the operational and maintenance fields. One of the purposes of the Atlas VMS intelligence layer is to be pro-activate by providing suggestions for the most effective message according to real-time traffic and weather situations. During normal VMS operation several actions (ex. messages signalled, namely human or automatic), alarms and states occur, to provide for auditing and business analysis every single status and action taken is logged (using the observer design-pattern from the Atlas framework already presented in section 2). Making use of the provider design-pattern implementation, also presented back in section 2, and a modular interface, different types (models/vendors) of VMS can be operated, maintained and provide for analysis with Atlas. To achieve this, besides a client interface for manual VMS interaction, several layers of applications and services where created.

In order for the Atlas VMS interface to be user-friendly and optimize interactions it centres on an editable display (depending on state and permissions) of the VMS. State information is presented when appropriate (ex.: the VMS is not reachable due to maintenance). A near-by CCTV with coverage of the VMS is available for visual validation if required (usually for maintenance purposes). CCTV access is one of a set of functionalities that are at a distance of one click: erasing the current message (sets the default message); accessing the most used messages for the current VMS or a set of context-based pre-defined messages; even navigating to other VMS (a list is always present). For integration with the incidence management platform that accompanies Atlas, or for using on context of alerts regarding dynamic context changes (namely weather), a message suggestion interface extends the normal interface. This interface provides the operator with a message suggestion with computed distance indications and context based text obtained from a rule based system. The rule based system is quite simple and uses a set of parameters like incidence typification and location (relative to the VMS). For periodic automated tasks related with VMS, namely a screen-saver that changes the position of the current time display in inactive VMSs, Atlas uses an agent implemented as a Windows Service. This agent is also responsible for the pooling of VMS providers that aren't synchronous (which isn't the case of TMS) and for the

evaluation of context conditions that may trigger message suggestion alerts (incidence and weather related).

Alarm information is available in real-time from TMS at the Alarm System Web Service. This service uses an event system (observer design-pattern) to take actions on the alarms; one of the actions is generic and persists this information for future reference. Atlas also offers interfaces for auditing actions and alarm status (and history). These are generic interfaces Atlas also provides for other times of equipments (like CCTVs). A specific history interface is available only for the VMS component in Atlas in order to provide access to the full history of messages present at a particular VMS (or set of VMSs).
As equipments are deployed to the field (and sometimes removed) Atlas had a requirement of enabling real-time addition and edition of VMS equipment. This is done through a CAD like interface which is simple enough for non-specialized usage. This interface enables mapping the positioning of a VMS in the infrastructure as well as editing the VMS configuration parameters (addresses, model/vender for adapter selection at run-time, etc.). This enables testing VMSs right after these are deployed in the field.

6. A MULTIPLATFORM, HIGHLY AVAILABLE ARCHITECTURE

Prior to the integration of TMS in the Traffic Atlas deployment at Brisa, other VMS management software was used. This provided no service level integration with Atlas[1] and had serious availability and reliability issues. TMS was created to overcome these limitations, the ITSIBus reference architecture was chosen for its proven results in Brisa's tolling implementation and because ITSIBus matched the requirements of both VMS integration and integration with Traffic Atlas.

6.1 Architecture overview

TMS exposes VMS functionality as a Web Service which is consumed by the Atlas service layer. Atlas provides for the service dependencies of TMS, namely a VMS registry interface and an Alarm handler. The following scheme presents these integration points as well as more detailed system architectures for Atlas and TMS:

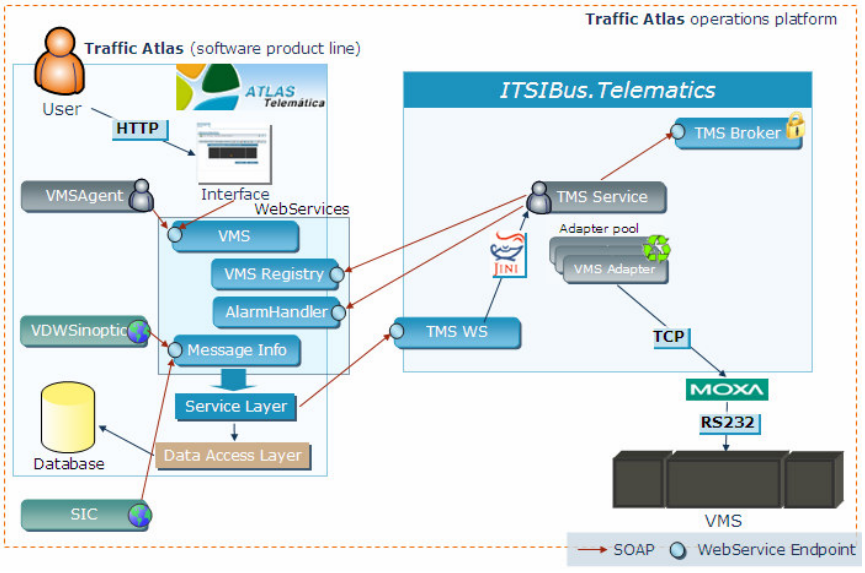

Diagram 3: Traffic Atlas systems architecture

[1] Integration was done at low levels, namely at the Data layer via database triggers and table monitorization.

Atlas' own internal architecture was described in generality back in section 2. The previous scheme details this architecture for the VMS component. The web-based interface consumes a business logic service layer offered by a Web Service (identified by VMS in the Atlas Web Services container box); this Web Service is also used by the VMS Agent as it also needs to make usage of business logic functionalities. Other services also based on this logic may use the functionalities already exposed by the VMS Web Service or create new Web Services, one case is a GIS synoptic system used for video-wall display (VDWSinoptic). Additionally the VMS Registry and TMS Alarm Handler both use the service layer directly as both are exposing additional VMS related functionality. The TMS system architecture has already been detailed in section 4. A determining design choice for TMS which enabled the high-redundant deployment architecture described next was that TMS is not state full and doesn't depend on a persistence store, like Atlas does.

6.2 Deployment architecture

The system architectures of both Atlas and TMS we the enablers of a high-redundancy deployment architecture based on hardware network balancing components. Additionally, as TMS is implemented in Java, deployment of TMS can be done to different operating systems, namely Windows and Linux. The deployment in Brisa has two separate logical servers, one where the Atlas application runs and another server for the TMS Service. For this deployment it was required that when Atlas might be unavailable the TMS monitoring service would be stopped and thus VMS would turn off (after some minutes without being pooled), indicating a malfunction. This way the TMS Broker component was deployed in the same server as Atlas. Each logical server is duplicated in two physical servers (if required, more could be added), the scheme below presents the four resulting servers along with the network load balancer components that enable redundancy (these are explained in more detail next).

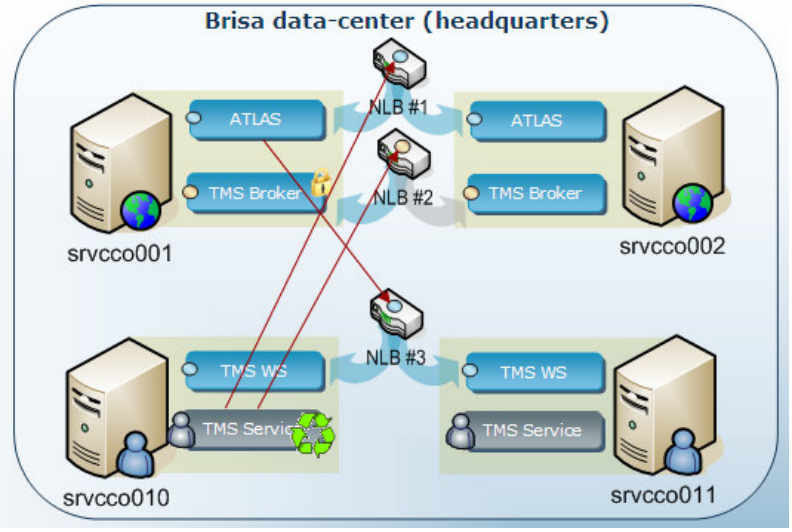

Diagram 4: Deployment architecture in Brisa

Atlas is active in both web-application servers and invokes TMS through means of a network load balancer (NLB #3). Network load balancers have two main modes of operation. One of the modes redirects requests received to a set of redundant servers which share the total load, this is the case of requests to the TMS Web Service (via NLB#3) and to Atlas (NLB#1). As the TMS Service is stateless by design and requires no persistence store, all running instances are available to service Atlas requests but only one is monitoring the VMS pool, the Broker decides which. The identification of the monitoring service is kept in memory by the Broker

(the icon in the scheme). The TMS service instance responsible for monitoring the VMS pool is located in *srvcco010* and represented with an icon. In case of failure of that service the Broker will service a request from a different server for the monitorization lease which will be in charge of monitorization a few moments after the failed service actually failed.
The Broker could not become a single point of failure, so a second mode of NLB operations was required. In such mode an NLB element redirects requests to the first active service in an ordered list of services; in case the first preference has failed the list is followed in search of a working service provider, this is the case of the TMS Broker Service with NLB#2. This way if a Broker service fails requests for leases (by non-monitoring nodes) and lease renewals (by the monitoring node) are redirected to a second Broker instance (at *srvcco002*) and, depending on the order of the requests monitorization is started at a new node or is renewed by the currently active monitorization node.
Additionally to supporting our high-availability architecture, both models of network load balancing allow release deployment without affecting system availability (by deploying separately at each node).

7. CONCLUSIONS

This paper has exposed a specification of a conceptualization used to implement a high-availability solution for traffic and telematic management systems. Here are some conclusions resulting from the deployment process:

- Efficient use of ITS resources depends on an ICT architecture that enables the efficient use of technology, information and increasingly, services available across the variety of ITS applications and systems.
- The use of studied patterns for system design and deployment empowers the solution efficiency and robustness, essential for critical environments.
- Software product lines and the architectures presented (namely ITSIBus) host the ideal environment for other types of telematic equipments, this is why ongoing work includes supporting weather stations in TMS and extending TMS's functional support for any type of telematic message devices (namely fuel price panels).

REFERENCES

[1] J. Lopes, J. Gomes, L. Osório, *"Open Architecture for Road Tolling and Traffic Management Services"*. ITS World Conference 2006, London, 2006.

[2] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *"Design Patterns: Elements of Reusable Object-Oriented Software"*, pgs. 139 and 293. Addison-Wesley, 1995.

[3] S. Black, *"Design Pattern: Hook Operations*". http://www.stevenblack.com/PTN-HookOperations.asp, July, 2000.

[4] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Videira Lopes, C., Loingtier, M., Irwin, J., *"Aspect-Oriented Programming"*. ECOOP, Springer-Verlag, 1997.

[5] J. Lopes, J. Bento, *"Cognitive Mapping for ITS Services"*. ITS World Conference 2005, San Francisco, 2005.

[6] C. Goncalves, B. Antunes, A. Amador, "ITSIBUS: Jini™ and RFID Technologies Enable Interoperability in an Open, Service-Oriented Architecture for Toll Management". JavaOne 2005, San Francisco, 2005