Intelligent Maintenance System Based On Ontologies and Web Services

Osama F. Zaki and Emilio M. Miguelanez*

Higher Institute of Computing and Managerial Science, Thebes Academy, Maddi, Egypt.

Abstract: The technologies and architecture presented in this paper was proposed and implemented by the author(s) in a real life research project for the European Union. The project objective was to improve the performance of the European railway system, by means of better information exchange between all parts of the system. As maintenance processes have an important role in defining the railway performances, improving maintenance can give an essential contribution to the overall project objectives. Therefore the system proposed here deals with the definition, specification and development of an Intelligent Maintenance System for railways (IMAIN). The overall architecture approach is based on the idea of distributed nodes; distributed services approach. This means Intelligent Maintenance Services are able to request additional information or functionality from other nodes, like: legacy maintenance management systems, monitoring services or even integrated information directly from accessible reasoners.

Keywords: Software Architecture, Distributed AI, Maintenance, Railways.

1. Introduction

The main purpose of this system is to be used as a tool to improve the railway services – passengers and freight – in line with their customers’ needs [2]. The architecture consists of number of scalable nodes and the functions of the system are allocated across different modules. There are three main activities (subprojects): Intelligent Rolling Stock Maintenance, Intelligent Rolling Stock/Infrastructure interaction and Intelligent Infrastructure Maintenance. The following steps are the implementation of such system:

- Data Sources: Data manifest, describing the one key point in the development which is the description of the data to be employed for the implementation and demonstration.
- Maintenance Ontology Development. This part describes the design of the ontology that encapsulates the maintenance concepts used in all maintenance applications.
- Implementation of Rolling Stock maintenance applications.
- Intelligent depot and tools
- Implementation of interfaces, which are responsible to enable the communication and exchange of information with other nodes. The IMAIN have interfaces to others systems. The other systems implement lean and condition based maintenance [4].

2. Overall view of the system

The overall architecture approach is based on the idea of distributed nodes; distributed services approach [5][9]. The diagram below shows some of the modules implemented and how the Intelligent Maintenance System (IMAIN) is linked to the other parts of the Intelligent Integration of Railway System.
The Intelligent Maintenance system is integrated with other systems such as the infrastructure, traffic management, and the operation management. Figure 2 depicts the system architecture for IMAIN for Rolling stock. Each one of this application will be explained in a dedicated subsection. Those are the applications that make the IMAIN system: Intelligent Depot Tool (IDT), Condition Analyser Onboard (CA Onboard), Condition Analyser On-ground (CA), Unplanned Event Manager (UEM), Predictive Maintenance Server (PMS), Lean Maintenance Optimiser (LMO).

Each application within IMAIN is composed of layers [6]. The layers are: back-end (e.g. triple store), business logic (e.g. reasoning mechanism), and front-end (IDT) IMAIN framework is explained in more in details in the following sections. Figure 3 shows the different layers that make an application within the IMAIN system.
In Figure 3, the back-end is mapped with repositories and front-end is mapped with interfaces (implemented using IMAIN framework – see following section). Business logic, however, is mapped into three different layers. Those layers representing the business logic and they are arranged differently according to the direction of the process. For example, in outward direction business logic is mapped into, Adaptor, Reasoning, and (other) processing. In the inward direction business logic is mapped into (other) processing, reasoning and Populating. Some of those layers are optional. For example, Reasoning for inward direction is optional and Adaptor for outward direction is optional.

The layers represented above are true for all applications within IMAIN system except for the Condition Analyser Onboard and the IDT. Some extra layers (or perhaps some less layers) can be altered for CA and IDT. IMAIN framework layer exits for all applications including the Condition Analyser Onboard and the IDT. An application may or may not access a legacy system (relational databases for example). This is done via adaptors. Main adaptor used is D2RQ [1]. A relational database can be accessed via D2RQ mapping. If information/knowledge is stored in a triple store format then adaptor is not needed.

Reasoning [8] is the mechanism to check the consistency of the knowledge base as well as performing re-classification on the knowledge.

Reasoning is performed via programming library called PELLET[8]

3. Triple Store and Ontology

IMAIN applications are driven by Ontology. Coreontology[7] is developed, which can be expanded in order to include several domains. The Ontology is further extended in the direction of maintenance processes. This means to include concepts like faults and their consequences, such as operational prescriptions (e.g. speed limit) and needed actions (e.g. maintenance required, part to be replaced, etc.), which can affect the train operation and the maintenance planning. Figure 4 shows an extract from the ontology.
Ontology is a way to represent the knowledge whether domain knowledge or environmental knowledge. Ontology is expressed in OWL language [8]. Individuals of each concept on the ontology are stored in a triple store format in any relational database management system. Triple stores are manipulated via a programming library called JENA [3].

### 4. IMAIN Framework

IMAIN Framework (Figure 5) is a set of modules that implement the interfaces between SP3B (IMAIN system) applications and also it is used to interface with SP3A (IMON system) and SP3C (ODDS). The IMIN Framework consists of the following Modules:

- **Module 1**: IMAIN Registration Server side - it allows application X to accept and store queries into the local registry database which is sent by application Y.
- **Module 2**: IMAN Registration Client side: allows application Y to register queries with the application X.

#### 2.1 IMAIN Web GUI Registration

#### 2.2 IMAIN Manual registration (bypassing server and client)

- **Module 3**: IMAIN Dynamic Push - allows application X to push the result back to application Y.
- **Module 4**: IMAIN Receiver – it allows application Y to receive data from application X
- **Module 5**: IMAIN Registry database Scanner - allows an application to scan its local registry database and then sends the query to the business logic
- **Module 6**: IMAIN Populate - it allows application Y to receive data from application X and populated into local registry

Some modules from the above are slightly modified to work on the locomotive (Condition Analyser Onboard). For example, Condition Analyser Onboard does not use registry database or database scanner. This means that most fields in the registry database are hardcoded.

IMAIN allows two applications (within SP3B) to communicate together using service-oriented architecture (web services). For example, an application X (e.g. the Unplanned Event Manager) registers a query with an application Y (e.g. the Condition Analyser), and whenever data is available to X it will be sent to Y - *push mechanism*. It also allows an application X (i.e. the Intelligent Depot Tool) to ask for data which is currently available to any of the SP3B applications - *pull mechanism*. 
The registration of events is done using relational database. Single table is used to handle the registration of events. Fields in the table are:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryID</td>
<td>VARCHAR(512)</td>
</tr>
<tr>
<td>ApplicationID</td>
<td>VARCHAR(128)</td>
</tr>
<tr>
<td>SPARQLQuery</td>
<td>VARCHAR(5024)</td>
</tr>
<tr>
<td>ResultFormat</td>
<td>VARCHAR(45)</td>
</tr>
<tr>
<td>URLEndPoint</td>
<td>VARCHAR(5024)</td>
</tr>
<tr>
<td>LeaseTime</td>
<td>INTEGER</td>
</tr>
<tr>
<td>FrequencyScan</td>
<td>INTEGER</td>
</tr>
<tr>
<td>QueryStatus</td>
<td>TINYINT(1)</td>
</tr>
</tbody>
</table>

ApplicationID is the ID of the application that registered the event. SPARQLQuery is the SPARQL query to be executed. ResultFormat determines the format of the result to be sent back (XML or RDF). URLEndPoint determines the endpoint address (where the result to be sent). LeaseTime determines how long the event is valid for. FrequencyScan determines the scan frequency (how often the event is executed). QueryStatus is used for internal process to check whether the event has been scanned or not.

Each application manages a registration table. A process is implemented to scan the events at regular basis. The interval period to scan the registration table (events) is determined independently (programmatically) by each application. Once the event is scanned, it is passed to another process to execute the event based on its frequency scan value and it will runs for the period equivalent to the lease time. This process runs all events in parallel but they are asynchronous manner.

For IMON/IMAN interaction, IMAIN (via the Condition Analyser) register queries with three different IMON nodes (that is it; BT On ground, FAR systems, and CDDoor). In actual fact, IMAIN registers only with BT on ground, after that using the IMON distributed reasoning mechanism other IMON nodes are explored automatically. The Condition Analyser registers and then receives information from different IMON nodes. Accordingly, the Condition Analyser sends data to the UEM. This is explained in another section.

IMAIN allows interaction with external applications SP3A (IMON) and SP3C (ODDS). For example, ODSS registers an event query with UEM, and then results are pushed back to ODSS. However, for ODSS, the IMAIN Receiver directs the result to another ODSS process instantaneously.

5. The Condition Analyser

The On Ground Condition Analyser considers all symptoms and environmental data in order to identify and classify system faults [11] according to their priority, and their nature (existing or incipient [10]) for a single vehicle or across the fleet. Furthermore, the On Ground Condition Analyser application is also able...
to infer information about the current condition (i.e. status) of the train and individual coaches with regard to its being available to operate based on the fault and context information. The main advantage of this application is based on the representation of the structure of the vehicle, the operational context of the vehicle and the relationships between diagnostic information as an ontology model, which structures the heterogeneous knowledge related to the system and to enable the common understanding between software applications. The main source of information is ground repository storing the diagnostic information corresponding to relevant process and event data. The On Ground Condition Analyser is capable to work with standard legacy systems, such as relational databases, as well as ontology based repositories.

Therefore the main tasks for On Ground Condition Analyser are (at vehicle or fleet level):

- Identification and classification of faults according to their types
- Inferring the status (availability to operate) of the train and individual coaches

Figure 6 shows the interaction of other applications connected to the IMAIN framework.

5.1 The Predictive Maintenance Server

The PMS analyses huge amounts of historical data related to several vehicles, trains and fleets, in order to extract new knowledge on devices behaviour and fault possibility.

The Predictive Maintenance Server is a ground software application that analyses all historical data in the fleet DB on ground in order to identify patterns which can bring to the occurrence of possible faults. The analysis, related to several vehicles, trains and fleets, is focusing on inferring new knowledge on devices behaviour and fault possibility, and it could be triggered by an external event (symptom), by a maintenance operator (establishing some rules) or by a routine procedure. The Predictive Maintenance server is based on the modelling of the diagnostic rules well known among the maintenance engineers, with the goal to indicate the need to perform a maintenance action at a scheduled point in time when the maintenance activity is most cost effective and before the equipment fails. The main source of information is ground repository storing the diagnostic information corresponding to relevant process and event data. The Predictive Maintenance Server is capable to work with standard legacy systems, such as relational databases, as well as ontology based repositories.

The following functions are implemented by PMS:

- Provides synthesis of diagnostic information from
temporal series, identifies vehicle critical states, and it provides diagnostic rules.

With a strong cooperation with the Condition Analyser application, the Predictive Maintenance Server will help maintenance engineers and operators to determine the condition of in-service equipment in order to predict when maintenance should be performed. Maintenance staff is able to edit and execute diagnostic rules that could be performance at certain point in time, or be triggered by component condition or behaviour.

With the Predictive Maintenance Server, maintenance engineers are able, in a technical and timely manner, to analyse equipment failures and forecast the probability of the same equipment failing in the same vehicle or other units, and consequently report the need to schedule a maintenance action before the failing of the system.

5.2 The Lean Maintenance Optimiser

The LMO tries to apply lean concepts in order to suggest the best decisions for optimised maintenance. The LMO refines the planning of maintenance activities in order to optimise them. Starting from the lean process KPIs (measuring the wastes) the process can be optimised. Data to be analysed include: The vehicle DB (including the FBS and the availability of each vehicle). The Train DB (operator side, linking the operational data as the train ID with the physical composition of the train). E.g.: link between maintenance distance and route distance allows maximizing the distance run between maintenance actions. The HR/Logistics DB (providing availability of maintenance operators and logistics). The Warehouse DB (providing the replacement components and materials availability). The LMO allocates the actions coming from the Unplanned Event Manager in the current planning of the maintenance depot, prioritizing the maintenance interventions. The LMO is the interface of IMAIN (or OMDS) towards maintenance legacy systems (MMS).

5.3 Unplanned Event Manager

This application queues faults and maintain a fault status record, tracing the linked events and actions performed (e.g. fault status asserted, notified, de-asserted; actions as work order issued, under investigation, deferred until later, rectified, work order closed). The UEM evaluates actual/incipient faults. If the fault is critical, it is notified to the ODSS. In all cases, it identifies the needed operational prescriptions and maintenance actions and, in general, it suggests to a human user a list of available possible actions and allows selection. The User can also include other different actions that should be stored and made available as future available action improving the expert knowledge store. New actions should be defined as structured data input, in order to avoid (by using Ontology) that the same action, defined slightly differently, brings to incorrect duplication. The performance reduction of a faulted train comes from prescriptions that are part of Unplanned Event Manager. As a consequence, this application is in charge to do that. It notifies needed actions to the Operator, to inform them about availability restrictions. The needed actions to solve the fault or prevent it generate a working order to be sent to MMS. If there is the Lean Maintenance Optimizer, it receives the request by the UEM and generates the working order after a processing of the other actions to be done to the same vehicle, in order to optimise (in this case = reducing the train stop) the intervention. The Operation Department of Rolling Stock will be informed about the need of Maintenance and the consequent unavailability of the train.

5.4 Intelligent Depot Tool

The Intelligent Depot Tool (IDT) offers a uniform interface to maintenance operators, grouping information coming from different sources and allowing a direct interface to all IMAIN applications.

The IDT is able to address all needed services in the grid and use them to provide answers to operator’s requests. It offers a uniform, easy to use human interface according to the common GUI (Graphical User Interface), providing all the information useful to support the maintenance process. Two different User profiles have been identified: the Maintenance Engineer, in charge to check the consistency of information and their proper usage. This User has the rights to increase the knowledge and modify existing cases coming from previous experience; Depot and Workshop personnel, in charge to use information in order to perform the maintenance interventions. The IDT supports the Users for complex analysing, possibly requiring special set of tools. It offers the User the capability to integrate, modify and complete the suggestions coming from the Reasoner, as reply to queries.
6. Conclusion

This paper presented real life project for railways. It is an application for distributed artificial intelligence.

Ontology has proved to be powerful knowledge representation platform where logic, rules, and semantic nets are integrated. It can be converted and stored in relational database systems. Web services can view as multi-agents system where agents can communicate and exchange knowledge and decisions.

All aspects of maintenance are considered and integrated together: condition analyser, unplanned event manager, predictive maintenance, and leamaintenance optimizer. In addition, an intelligent GUI is developed which reflects the comprehensive amount of works carried beneath. The project was demonstrated to main key players in railways industry in Europe and worldwide and it has been very successful.

References


