Proceedings of Applied Informatics - AI'99, Innsbruck (Austria), February 1999

AUTOMATIC CONFIGURATION FOR REMOTE DIAGNOSIS AND MONITORING OF RAILWAY EQUIPMENTS

TXOMIN NIEVA
Institute for computer Communications and Applications (ICA)
Department of Computer Science (DI)
Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland
E-mail: txomin.nieva@epfl.ch

This article is sponsored by the FNRS (Swiss National Science Foundation) program as part of the NePESM (New Paradigms for Embedded Systems Management) project, and ABB Corporate Research Ltd (Switzerland).

Keywords: Automatic Configuration, Remote Diagnosis and Monitoring, Embedded Devices, Web Technology

Abstract: Today, there is no standard system for performing maintenance on heterogeneous railway equipments. Maintenance staff needs a standard application capable of diagnosing and monitoring heterogeneous on-board equipments. We propose to build a web-based diagnosis and monitoring system. The main advantage of our approach is that it offers a well known and user friendly interface: a web browser (such as Netscape Navigator or Microsoft Internet Explorer). In addition, it gives the possibility for remote diagnosis and monitoring railway equipments. Thus experts, not necessarily on site in the depot, can make diagnosis and detect faulty components. They can inspect, more than one vehicle at a time, directly from their office. The on-board communication system and related network management services provide a unified entry point to the train data for diagnosis. We have built an automatic configuration approach for such a maintenance system. As a result, this maintenance system can be installed on any vehicle without previous device-specific knowledge about that vehicle. This paper is intended for people concerned with industrial applications to the Internet and especially for those developing remote monitoring tools for embedded devices.

❑ INTRODUCTION

Over the last several years, a strong demand has grown among railway operators for a modern, versatile communication system on-board trains, both to interconnect equipment located inside a railway vehicle and to allow communication between different vehicles. Such a network was specified by IEC as the "Train Communication Network" [1].

Based on this standard network, under the 5th Program, the European Project ROSIN (Railway Open System Interconnection Network) defined standard applications. In particular, the RoMain project addressed the monitoring of heterogeneous on-board equipments to support maintenance work. I have been working on this project, in collaboration with *ABB Corporate Research Ltd. (Switzerland)*, focusing my research on a web-based and automatic configurable architecture.

This paper is organized as follows: First, we introduce the RoMain approach. Then, we explain the system architecture focusing on the automatic configuration capability. In this part, we emphasize the information that we need to implement this capability and the technique for storing and retrieving it. Later, we explain the internal architecture of our approach. Finally, we draw conclusions from the actual work.

❑ THE ROMAIN APPROACH

Today, there is no standard system for performing maintenance on heterogeneous railway equipments, due to the diversity of low level communication protocols, equipment manufacturers, and operating systems used by maintenance and commissioning staff. This impedes standardization. Each equipment manufacturer has to build its own software to monitor its systems taking into account these constraints. This practice is wasteful and redundant.

We have built a standardized web-based remote diagnostic and monitoring system that supports maintenance. Our aim was to develop a framework in which the equipment manufacturers provide the equipment web pages for monitoring and performing maintenance on their equipments. The vehicle assembler

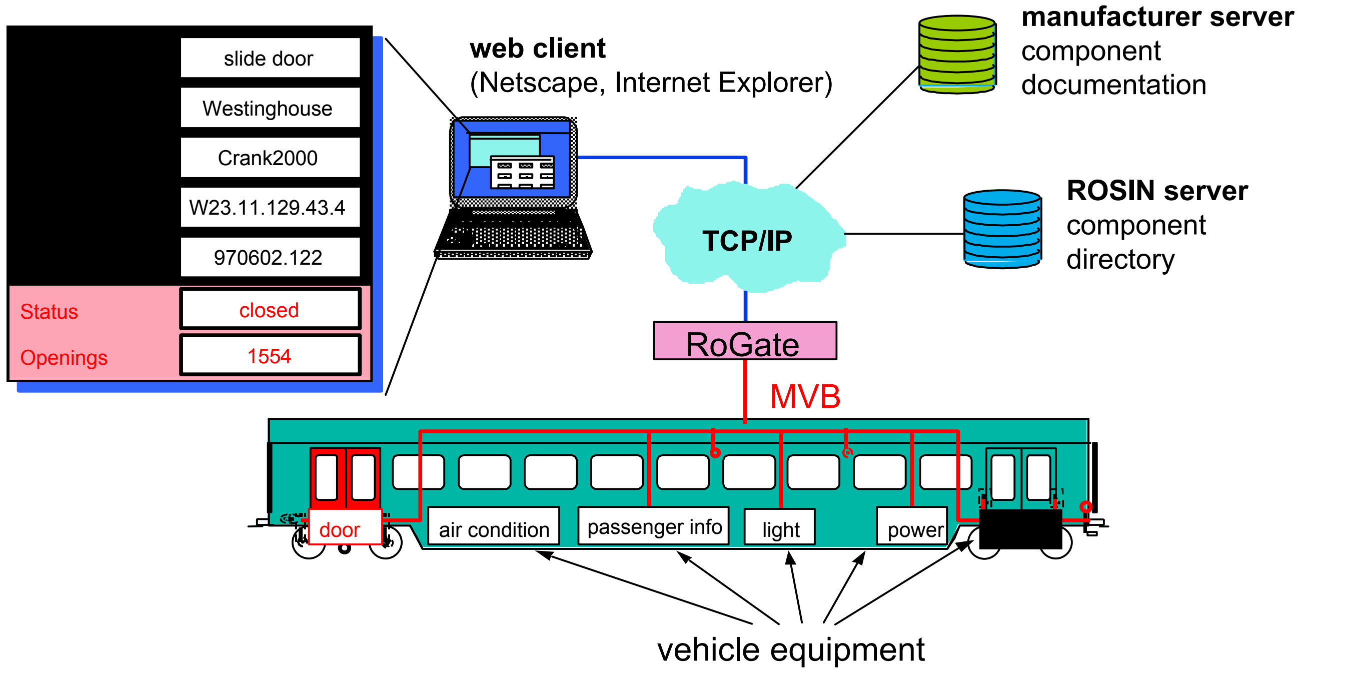

FIGURE 1: SYSTEM ARCHITECTURE

uses these pages to build a model of the entire vehicle, offering to the maintenance people a user-friendly way to monitor all the equipments on a vehicle. We have built an automatic configuration approach for this maintenance system. With this approach, this maintenance system can be installed on any vehicle without having any previous device-specific knowledge about that vehicle.

Our approach offers a well known and user-friendly interface: a web browser (such as Netscape Navigator or Microsoft Internet Explorer). Moreover, it allows remote diagnosis and monitoring. Thus experts, not necessarily on site in the depot, can make diagnosis and detect faulty components. They can inspect, more than one vehicle at a time, directly from their office. The experts may inspect, directly from their office, more than one vehicle at a time. Taking advantage of the on-board communication system and related network management services, our system provides a unified entry point to the train data for diagnosis.

I. SYSTEM ARCHITECTURE

In our architecture, see FIGURE 1, an element called ***RoGate*** (*Railway Open GATEway*) is connected to a fieldbus, which interconnects several devices (such as sensors, PLCs, …) that implement equipments (like doors, brakes, HVAC …). The core of the *RoGate* is the *GLASS* server, which is a remote monitoring system based on Internet technology. In this server, each component (vehicle or equipment) is represented by a *proxy* object [2]. This *proxy* retrieves the actual data from the equipment.

The component documentation (web pages, Java applets, etc…) is stored in a web server (typically the manufacturer web server) located somewhere on the Web. The manufacturer registers its web pages in a known server (ROSIN server) by giving to the server the component identifier and the hyperlink of its documentation. The server offers a means to get this documentation from the Internet.

II. AUTOMATIC CONFIGURATION

Maintenance staff does not want to spend time installing or configuring anything in order to monitor their systems. They just want to perform maintenance. Consequently, our idea was to have an automatic configuration, which enables the maintenance system to configure itself when it is attached to a new vehicle.

To do this, some "extra" configuration information for maintenance must be stored in the vehicle itself. We refer to this information as "*maintenance information metadata*", which allows:

- The retrieval of the vehicle and equipments documentation from somewhere on the Internet.
- The automatic generation of the vehicle and equipments *proxies* on the *RoGate.*

These proxies allow the *RoGate* to access the equipments through the specific fieldbus and update their web pages with the actual value of their variables.

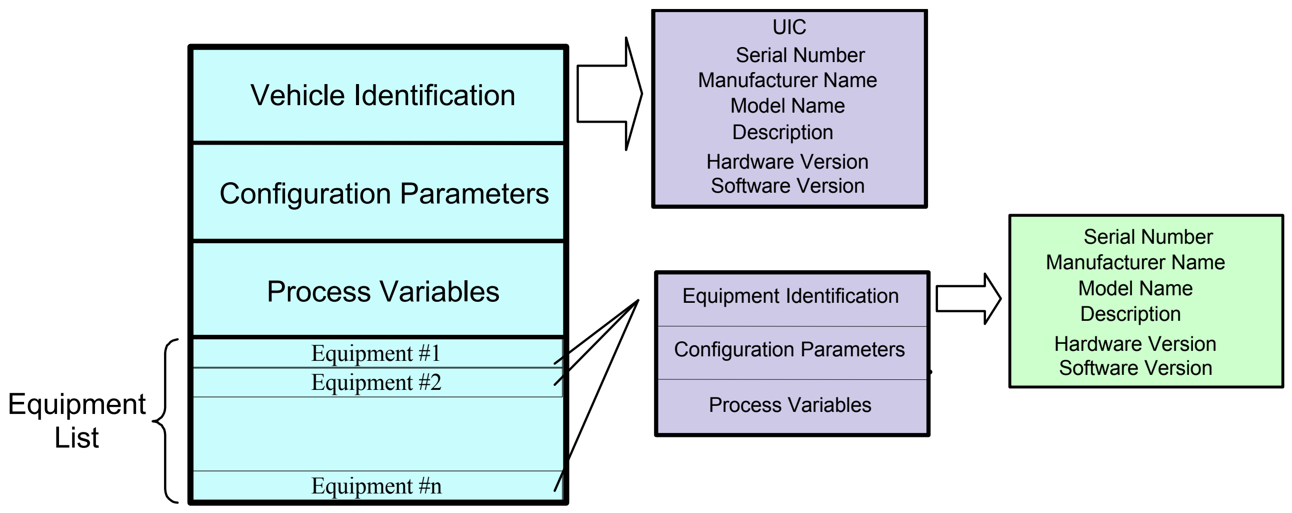

FIGURE 2: MAINTENANCE INFORMATION METADATA STRUCTURE

A. MAINTENANCE INFORMATION METADATA

The most flexible solution is to store, on the vehicle, the smallest dataset sufficient to identify the different components. It must contain enough information to automatically build component proxies. This dataset should be defined by equipment manufacturers and vehicle assemblers during the vehicle building process, and it is stored on a dedicated device, usually the train bus 'node', located in every vehicle. The *maintenance information metadata* structure is, graphically shown in FIGURE 2, and described briefly as follows:

➢ Vehicle Identification

- **UIC Number**: unique identifier for the vehicle, which is universally assigned by the UIC (Union Internationale des Chemins de Fer) international association.
- **Serial Number**: serial number of the vehicle. This field typically identifies a vehicle inside a company.
- **Manufacturer Name**: name of the vehicle manufacturer.
- **Model Name**: name of the model of the vehicle.
- **Description**: textual field that describes the vehicle.
- **Hardware Version**: version of the hardware where the vehicle is implemented.
- **Software Version**: version of the software, which implements the vehicle.

➢ *Configuration Parameters*

Specific information about the vehicle (e.g. the last revision date, etc …). A configuration parameter is described through:

- **Name**: name of the parameter.
- **Type**: type of the parameter.
- **Description**: textual field that describes the configuration parameter.
- **Value**: actual value with free format and length.

➢ *Process Variables*

The list of network variables exported by this vehicle, which corresponds to the internal application variables useful for maintenance. Each process variable is described with the following fields:

- **Name**: logical name for the variable on the bus.
- **Type**: type of the process variable.
- **Description**: textual field that describes the process variable.
- **Address**: network address of a process variable.

➢ *Equipment List*

The list of equipments installed on the vehicle with, for each equipment, the following fields:

➢ Equipment Identification

Similar to the *Vehicle Identification*. There is not a unique identifier for an equipment, thus we built it with a combination of *Manufacturer Name*, *Model Name*, *Hardware Version* and *Software Version* fields. This *UID* is analogous to the *UIC Number* on a vehicle.

➢ Configuration Parameters

The same as in a vehicle.

➢ Process Variables

The same as in a vehicle.

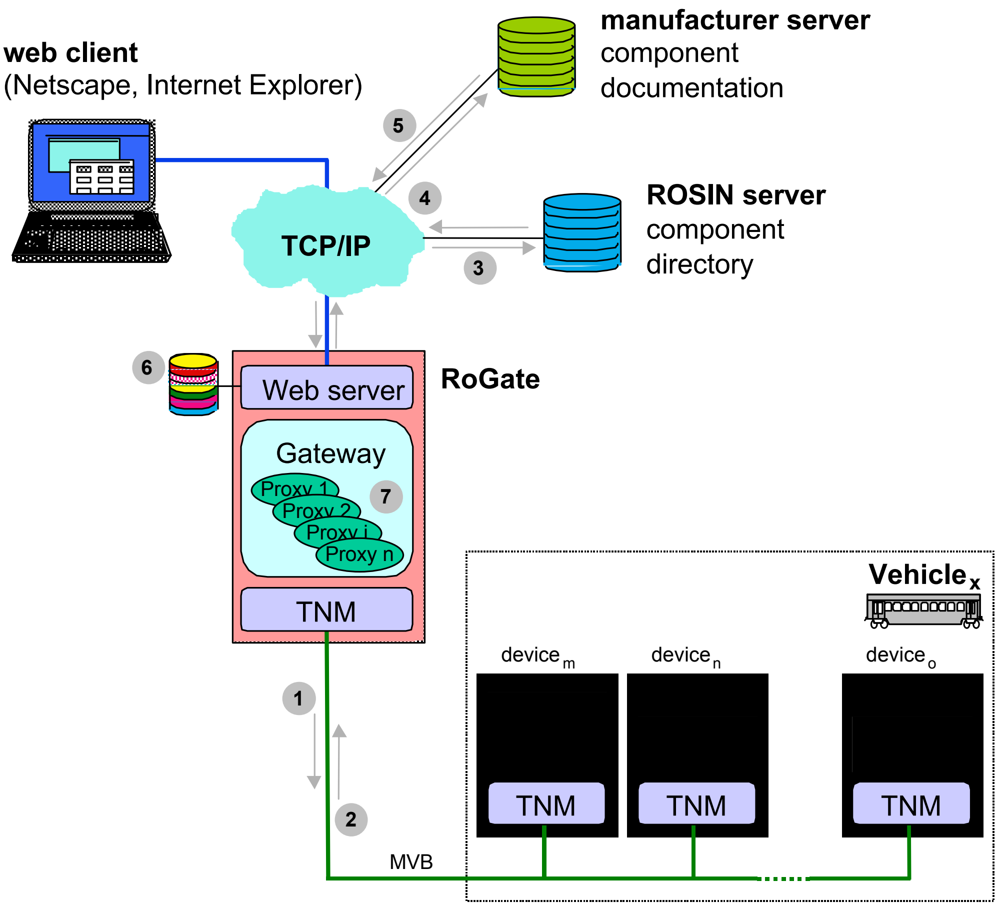

FIGURE 3: ROGATE INITIALIZATION

B. SYSTEM INITIALIZATION

When a *RoGate* is connected to a new vehicle the initialization process is started. After this process, the *RoGate* presents to the maintenance staff the vehicle and equipment web pages with actual values. The initialization of the *RoGate* is shown in FIGURE 3 and consists of the following steps:

1. The *RoGate* asks the vehicle 'node' for its maintenance information structure. This is made through a TNM (Train Network Management) message called *Call_Read_Maintenance_Information*.

2. The vehicle 'node' replies to him with its maintenance information structure using the message called *Reply_Read_Maintenance_Information*.

3. The *RoGate* asks a *component directory* for the component documentation (web pages, Java applets and so on). This is done using the combination of *ManufacturerName*, *ModelName*, *Hardware Version* and *Software Version* as a key in the case of an equipment, and using the *UIC* number in the case of a vehicle.

4. The *component directory* redirects the call to the actual location of this documentation.

5. The response to this call is this documentation in a single *Zip* file.

6. The *RoGate* extracts the files from this Zip file and saves them locally.

7. The *RoGate* generates a local *component proxy*, which is responsible for updating the web pages with the actual state of the variables running on the real component.

Note: the "*Read_Maintenance_Information*" message has been formally specified, using the "*Rosin Representation and Notation*" [3], and proposed as an extension to the Train Network Management standard [4].

C. DEVICE VS. EQUIPMENT

We define a device as a hardware unit connected to one or more buses. Otherwise, we define an equipment as a train function like doors, lights, HVAC… Usually, an equipment is implemented on a device. However, a complex equipment could be implemented on several devices, where one of them would act as "leader". On the other hand, because devices are expensive hardware components, several equipments could be implemented on a single device. Regardless, the maintenance information metadata is independent of the allocation of equipment to devices.

D. ROGATE INTERNAL ARCHITECTURE

A *RoGate* is composed, as shown in FIGURE 4, of the following modules:

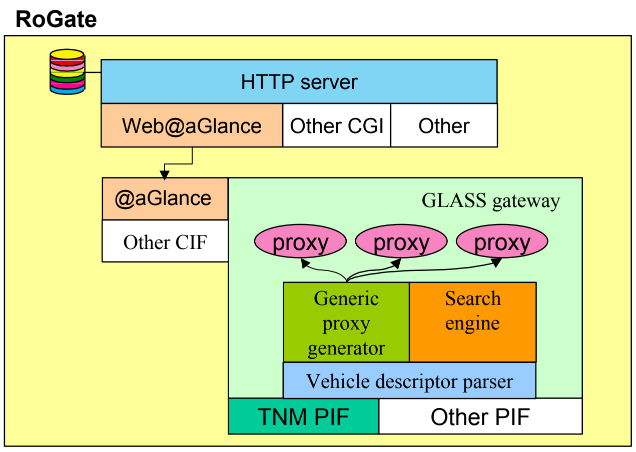

FIGURE 4: **ROGATE INTERNAL ARCHITECTURE**

- ***HTTP Server***: an off the shelf web server.
- ***GLASS Gateway***: connects to clients, process buses, and database systems through independent interfaces and manages *proxy* objects that represent real devices [GLASS98]. *GLASS Gateway* is based on an off-the-shelf web-based monitoring system: *web@aGlance* / *@aGlance* (http://www.aglance.com), by *Intuitive Technologies*.
- ***Vehicle Descriptor Parser***: offers a user-friendly interface to the *maintenance information metadata* structure.
- ***Generic Proxy Generator***: uses the *Vehicle Descriptor Parser* to read and dynamically create *proxies*, for a vehicle and its equipments, from the *maintenance information metadata*.
- ***Search Engine***: makes a local copy of the vehicle and equipments documentation from the Internet, using the *maintenance information metadata*. It forms an HTTP request that it sends to the known common server for all component manufacturers. The response to the request is a Zip file, which corresponds to the component. Once the file is stored locally, it extracts and installs the included files.

❑ CONCLUSION

The RoMain tool provides a unified entry point to the train data for diagnosis and maintenance with the following features:

- Universal access
- Well-known and user-friendly interface
- Low-cost
- On-line documentation and help files

In addition, an automatic configuration approach makes it possible to install the system on any vehicle without previous device-specific knowledge about that vehicle.

We implemented a prototype of this web-based maintenance system. A demonstration of this prototype is at: http://icawww.epfl.ch/nieva/romain.htm.

❑ ACKNOWLEDGMENTS

Special thanks to Hubert Kirrmann for his support during my stage at *ABB Corporate Research*; to Monica Perez who did a very good job capturing the needed information for maintenance and developing tools to treat them; to the GLASS team who helped me understand how the GLASS server runs and how I could use it for our purposes; and to all the members of the ROSIN WP4 who brought a real framework to the discussions and the implementation of our hypotheses.

❑ REFERENCES

[1] IEC-61375. *Train Communication Network Clause 1: General.* November 1995.

[2] Robert Itschner, Claude Pommerell and Martin Rutishauser. *GLASS: Remote Monitoring of Embedded Systems in Power Engineering*. IEE Internet Computing, Volume 2, Number 3, May-June 1998.

[3] ROSIN WP04. *Rosin Representation and Notation, V2.0*. May 1998.

[4] IEC TC9 WG22. *Train Communication Network Part 5: Train Network Management, V1.1.3*. April 1998.