Train Management Platform for advanced maintenance of Passenger Information Systems (PIS)

Stijn Verstichel (Stijn.Verstichel@intec.UGent.be)
A. Baart, G. Lievens, S. Latre, F. Deturck and F. Vermeulen
Introduction

- **Context : Trains**
  - Dynamic
  - Complex

- **Problem : Maintenance**
  - Preventive ➔ Replace at regular intervals
  - Corrective ➔ Replace when broken

⇒ Can be optimized
Introduction

From Idea towards Architecture

Ontology

Results and Conclusions
“We want to design an information-infrastructure, capable of correlating the available status information, in order to perform cost-effective maintenance”
How?

1. Perform measurements per component
   - On demand
   - Scheduled
2. Store measurements
3. Correlate the information
4. Draw conclusions

Advantages!

- Robust
- Efficient maintenance
- Less expensive maintenance
- Prediction of faults and failures
- Self-descriptive
- ....

Use of Semantic Web’s First-Order Logic Ontology
Functional Building blocks

- Obtain status-information from components
- Analyze communication between components
- Impose consequences
- Provide feedback to the Users
- Easily install now model of the system
- Collect information from imposed consequences
Scenarios

Obtain status-information from components

- Component
- Application gateway
- System

Request Measurement
Result

Filter
Store
From Idea to Architecture

"topdown"-design method

Message Mapper  Bus Mapper  Communication Mapper

Audio  Display  Heating  Lights  GPS-3  Unit

Message forwarder

Data Storage

Communication

UnitServer

Message Interface

Communication Interface

GUI Interface

GUI Mapper

Communication splitter

Business Logic Interface

Data Storage

Business Logic


p. 8
From Idea to Architecture

Business Logic

- Communication Interface
- Communication Router
- Communication Controller
- Measurement Controller
- Consequences Controller
- Wagon Controller
- GUI Adapter
- Data Storage Router
- Data Storage Interface

Data Storage

From Idea to Architecture

Functional Building blocks → Scenarios → Architecture

Data Storage Controller

- Business Logic Interface
- Reasoner Controller
  - Racer Local Mapper
  - Racer Local Interface
  - Reasoner B Local Mapper
  - Reasoner B Local Interface
  - Racer Remote Mapper
  - Racer Remote Interface
  - Reasoner B Remote Mapper
  - Reasoner B Remote Interface
- Ontology Controller
  - Ontology Interface Remote
  - Ontology Interface Local

Introduction to ontology

- **RDF (Resource Description Framework)**
  - Basic expressiveness

- **OWL (Ontology Web Language)**
  - built on top of RDF
  - Lite: classification hierarchy and constraint features
  - DL (Description Logic): maximum expressiveness without losing computational completeness and decidability
  - Full: maximum expressiveness and syntactic freedom with no computational guarantees
Key Features of Ontology

- **Self-descriptive**

- **Application Logic inside the model**

- **1st – order – logic**

- **Automatic**
  - Validation
  - Classification
  - Inference

  *Reasoner*
Overview TIO.owl

Overview TIO_remote.owl
Results

Maintenance Tool
Results

Maintenance Tool
Conclusions

- Currently:
  - Maintenance = straightforward BUT inefficient
  - Maintenance = \( f(MTBF) \)
    - failure

- Proposed maintenance application
  - Maintenance optimized per component
  - Maintenance tool for visualisation

  - Maintenance = simple AND more efficient
  - Maintenance = \( f(MTBF, \text{cost}) \)
Conclusions

- Value for Money

  **Operator/Owner**
  + Financially interesting
  + Improved robustness

  **Maintainer**
  + Easier maintenance
  + Improved planning

  **Passengers**
  + Greater uptime for components and systems
  + Improved and up-to-date information

- Ontology

  + dynamic, reasoning, self-descriptive,…
  - complex and (currently) not yet performant
Thank you for your kind attention!