Development of a Self-diagnosis Tests System Using a DSL for Creating New Test Suites for Integration in a Cyber-physical System

Ricardo B. Pereira, José C. Ramalho, Miguel A. Brito
ricardo-97-pereira@hotmail.com, jcr@di.uminho.pt, mab@dsi.uminho.pt

10th Symposium on Languages, Applications and Technologies
Agenda

● Contextualization of the project
● Problem
● Motivation and Objectives
● Contributions
● Proposed Architecture
● Implementation, more focused on the DSL
● Conclusions and Future Work
Contextualization

- TSIM - Test System Intelligent Machines
- Consortium:  
  - CONTROLAR  
  - University of Minho  
  - Centro de Computação Gráfica
- Bosch Supplier Club Initiative Program
- Intelligent Functional Testing System Machine:  
  - Cyber-physical system  
  - Functional tests on car radios
Failures in several car radios in a row

Bosch tries to find the problem on the machine

Repair or replace the machine, which leads to financial losses for Controlar and Bosch.

Distrust in the correct functioning of the machine

Causing damage to the machine itself until it is completely unusable
Motivation and Objectives

How can we overcome this problem?

- Integrate a self-diagnosis tests system in the machine
  - Integration of software in Cyber-physical system
  - System to manage the tests and their execution

- Develop an architecture for the system
  - As suitable as possible for the problem
  - As generic as possible regarding what a testing system is

- Provide a methodology to integrate the system into the Cyber-physical system
State of the Art Contribution

1. A new modular and extensible architecture for self-diagnosis tests systems
2. A methodology to integrate a self-diagnosis tests system into a cyber-physical system
3. Implementation of a self-diagnosis tests system that validates the proposed architecture
Self-diagnosis Tests System Architecture
Self-diagnosis Tests System Architecture

**FRONTEND TIER**
- OPERATOR
- TEST MANAGER
- TESTS EXECUTION
- SYSTEM CONFIGURATION
- USER INTERFACE SERVER (REACT)

**BACKEND TIER**
- API FRAMEWORK (EXPRESS)
- BUSINESS LOGIC SERVER (NODEJS)
- TESTS MANAGEMENT (KDT+DSL)
- OBJECT DATA MODELING (MONGOOSE)

**DATABASE TIER**
- MONGODB

HTTP Requests

Queries

HTTP Responses

JSON Data
Self-diagnosis Tests System Architecture

FRONTEND TIER
- OPERATOR
- TEST MANAGER
- TESTS EXECUTION
- SYSTEM CONFIGURATION
- USER INTERFACE SERVER (REACT)

BACKEND TIER
- API FRAMEWORK (EXPRESS)
- BUSINESS LOGIC SERVER (NODEJS)
- TESTS MANAGEMENT (KDT+DSL)
- OBJECT DATA MODELING (MONGOOSE)

DATABASE TIER
- MONGODB

HTTP Requests
HTTP Responses
Queries
JSON Data
Self-diagnosis Tests System Architecture

FRONTEND TIER
- OPERATOR
- TEST MANAGER
- TESTS EXECUTION
- SYSTEM CONFIGURATION
- USER INTERFACE SERVER (REACT)

BACKEND TIER
- API FRAMEWORK (EXPRESS)
- BUSINESS LOGIC SERVER (NODEJS)
- TESTS MANAGEMENT (KDT+DSL)
- OBJECT DATA MODELING (MONGOOSE)

DATABASE TIER
- MONGODB

HTTP Requests
- Queries

HTTP Responses
- JSON Data
Self-diagnosis Tests System Architecture

**FRONTEND TIER**
- **OPERATOR**
- **TESTS EXECUTION**
- **SYSTEM CONFIGURATION**
- **USER INTERFACE SERVER (REACT)**

**BACKEND TIER**
- **API FRAMEWORK (EXPRESS)**
- **BUSINESS LOGIC SERVER (NODEJS)**
- **TESTS MANAGEMENT (KDT+DSL)**
- **OBJECT DATA MODELING (MONGOOSE)**

**DATABASE TIER**
- **MONGODB**

HTTP Requests

HTTP Responses

Queries

JSON Data
Self-diagnosis Tests System Architecture

Frontend Tier:
- Operator
- Test Manager
- Tests Execution
- System Configuration
- User Interface Server (React)

Backend Tier:
- API Framework (Express)
- Business Logic Server (Node.js)
- Tests Management (KDT+DSL)
- Object Data Modeling (Mongoose)

Database Tier:
- Mongodb

HTTP Requests
- To Frontend Tier
- To Backend Tier

HTTP Responses
- From Backend Tier
- From Database Tier
Self-diagnosis Tests System Architecture
Self-diagnosis Tests System Architecture
Self-diagnosis Tests System Architecture

**FRONTEND TIER**
- OPERATOR
- TEST MANAGER
- TESTS EXECUTION
- SYSTEM CONFIGURATION
- USER INTERFACE SERVER (REACT)

**BACKEND TIER**
- API FRAMEWORK (EXPRESS)
- BUSINESS LOGIC SERVER (NODEJS)
- TESTS MANAGEMENT (KDT+DSL)
- OBJECT DATA MODELING (MONGOOSE)

**DATABASE TIER**
- MONGODB

HTTP Requests
- From USER INTERFACE SERVER
- From TESTS EXECUTION

HTTP Responses
- To USER INTERFACE SERVER
- To TESTS EXECUTION

Queries
- To MONGODB

JSON Data
- From MONGODB
Self-diagnosis Tests System Architecture

**Frontend Tier**
- Operator
- Test Manager
- Tests Execution
- System Configuration
- User Interface Server (React)

**Backend Tier**
- API Framework (Express)
- Business Logic Server (Node.js)
- Tests Management (KDT+DSL)
- Object Data Modeling (Mongoose)

**Database Tier**
- MongoDB

HTTP Requests
HTTP Responses
Queries
JSON Data
Self-diagnosis Tests System Architecture
Self-diagnosis Tests System Architecture

FRONTEND TIER
- OPERATOR
- TEST MANAGER
- TESTS EXECUTION
- SYSTEM CONFIGURATION
- USER INTERFACE SERVER (REACT)

BACKEND TIER
- API FRAMEWORK (EXPRESS)
- BUSINESS LOGIC SERVER (NODEJS)
- TESTS MANAGEMENT (KDT+DSL)
- OBJECT DATA MODELING (MONGOOSE)

DATABASE TIER
- MONGODB

HTTP Requests
- OPERATOR
- TEST MANAGER
- HTTP Responses
- TESTS EXECUTION
- SYSTEM CONFIGURATION
- USER INTERFACE SERVER (REACT)
- HTTP Requests
- Queries
- JSON Data
- MONGODB
General Architecture for Cyber-Physical System

- **Electronics Test Drivers**
  - Executable Driver
  - Metadata

- **Device Under Test**
  - Tsim Machine (Hardware)
  - Car Radio

- **Self-Diagnosis Tests System**
  - Api Framework (Express)
  - Business Logic Server (Node.js)
  - Tests Management (KDT+DSL)
  - User Interface Server (React)
  - System Configuration
  - Tests Execution
  - DATABASE (Mongodb)
  - Object Data Modeling (Mongoose)

- **User Types**
  - Test Manager
  - Operator
General Architecture for Cyber-Physical System

CYBER-PHYSICAL SYSTEM ARCHITECTURE

- ELECTRONIC TEST DRIVERS
  - EXECUTABLE DRIVER
  - METADATA

- DEVICE UNDER TEST
  - TSIM MACHINE (HARDWARE)
  - CAR RADIO

SELF-DIAGNOSIS TESTS SYSTEM

- API FRAMEWORK (EXPRESS)
- BUSINESS LOGIC SERVER (NODEJS)
- TESTS MANAGEMENT (KDT+DSL)
- OBJECT DATA MODELING (MONGOOSE)
- DATABASE (MONGODB)

USER TYPES

- TEST MANAGER
- OPERATOR

TEST MANAGER

TESTS EXECUTION

SYSTEM CONFIGURATION

USER INTERFACE SERVER (REACT)
General Architecture for Cyber-Physical System

- **EXECUTABLE DRIVER**
- **METADATA**
- **TSIM MACHINE (HARDWARE)**
- **CAR RADIO**
- **REST API**
- **API FRAMEWORK (EXPRESS)**
- **BUSINESS LOGIC SERVER (NODEJS)**
- **TESTS MANAGEMENT (KDT+DSL)**
- **USER INTERFACE SERVER (REACT)**
- **SYSTEM CONFIGURATION**
- **TESTS EXECUTION**
- **DATABASE (MONGODB)**

**SELF-DIAGNOSIS TESTS SYSTEM**

**USER TYPES**
- **TEST MANAGER**
- **OPERATOR**

**DEVICE UNDER TEST**

**ELECTRONIC TEST DRIVERS**
- **EXECUTABLE DRIVER**
- **METADATA**
- **EXECUTABLE DRIVER**
- **METADATA**
- **EXECUTABLE DRIVER**
- **METADATA**
- **EXECUTABLE DRIVER**
- **METADATA**
- **EXECUTABLE DRIVER**
- **METADATA**

**GENERAL ARCHITECTURE FOR CYBER-PHYSICAL SYSTEM**

- **SELF-DIAGNOSIS TESTS SYSTEM**
- **USER INTERFACE SERVER (REACT)**
- **BUSINESS LOGIC SERVER (NODEJS)**
- **TESTS MANAGEMENT (KDT+DSL)**
- **API FRAMEWORK (EXPRESS)**
- **REST API**
- **DATABASE (MONGODB)**

**USER TYPES**
- **TEST MANAGER**
- **OPERATOR**
General Architecture for Cyber-Physical System

**Cyber-Physical System Architecture**

- **Electronic Test Drivers**
  - Executable Driver
  - Metadata

- **Device Under Test**
  - Tsim Machine (Hardware)
  - Car Radio

- **Self-Diagnosis Tests System**
  - API Framework (Express)
  - Business Logic Server (NodeJS)
  - Tests Management (KDT+DSL)
  - User Interface Server (React)
  - System Configuration
  - Tests Execution
  - Object Data Modeling (Mongoose)
  - Database (MongoDB)

**User Types**

- **Test Manager**
- **Operator**
General Architecture for Cyber-Physical System

**CYBER-PHYSICAL SYSTEM ARCHITECTURE**

- Device Under Test
- Tsim Machine (Hardware)
- Car Radio

**Electronic Test Drivers**

- Executable Driver
- Metadata

**Business Logic Server** (Node.js)

**Tests Management** (KDT+DSL)

**Object Data Modeling** (Mongoose)

**Database** (Mongodb)

**Self-Diagnosis Tests System**

- API Framework (Express)
- REST API

**User Interface Server** (React)

**Tests Execution**

**System Configuration**

**User Types**

- Test Manager
- Operator
General Architecture for Cyber-Physical System
Database

- Document Database:
  - Configurations
  - Tests
  - Packages
  - Reports
  - Schedules
Backend

● Structure:
  ○ Models - perform validations related to data types
  ○ Controllers - perform all system operations
  ○ Grammar - domain-specific language developed
  ○ Routes - handle the requests from the clients
## Domain Specific Language - Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>keyword</strong></td>
<td>Keywords in the script</td>
</tr>
<tr>
<td>-&gt;</td>
<td>“Next” symbol, which means that after this symbol comes the next block of execution</td>
</tr>
<tr>
<td>?</td>
<td>Comes after the conditional expressions and it means that the next block of execution will be executed when the previous conditional expression is true</td>
</tr>
<tr>
<td>(</td>
<td>Opening parenthesis, which means that will begin a conditional expression.</td>
</tr>
<tr>
<td>&amp;</td>
<td>Logical operator “And”, that means intersection</td>
</tr>
<tr>
<td></td>
<td>Logical operator “Or”, that means union</td>
</tr>
<tr>
<td>)</td>
<td>Closing parenthesis, which means that ended the conditional expression</td>
</tr>
<tr>
<td>:</td>
<td>“Else” symbol, which means that the next block of execution will be executed when the previous conditional expression is false</td>
</tr>
<tr>
<td>;</td>
<td>End of the script</td>
</tr>
</tbody>
</table>
Domain Specific Language - Example

( Connect & Open ) ?

Read -> Write -> Close -> Disconnect :

Disconnect ;

Symbols

| keyword | -> | ? | ( | & | | ) | : |

Keywords

| Connect |
| Open |
| Read |
| Write |
| Close |
| Disconnect |
Frontend - UI

Lista de Pacotes

- Pacote AM FM
- Pacote Exemplo
- Pacote Completo
- Pacote Loopback
- Pacote BroadR-Reach
- Pacote Sequência
- Pacote Condicional

Criar Novo Pacote

Nome:
Novo pacote

Descrição:
Novo pacote, apenas para demonstração.

Código:
Power_on ->

Conectores:

LIMPAR  GUARDAR

Lista de Testes

- Send_packet
- Fail_Error
- Power_on
- Power_off
- Set_Tone_fm_frequency
- Fm_seek_right
- AMFM_RX_set_Volume
- Check_fm_signal_quality
- Get_fm_SNR_value
- Get_fm_RSSI_value
Frontend - UI

Relatórios
AGENDAMENTOS
GESTÃO DE PACOTES
DOCUMENTAÇÃO
CONFIGURAÇÕES
SAIR

Lista de Pacotes

- Pacote AM FM
- Pacote Exemplo
- Pacote Completo
- Pacote Loopback
- Pacote BroadR-Reach
- Pacote Sequência
- Pacote Condicional

Criar Novo Pacote

Nome
Novo pacote

Descrição
Novo pacote, apenas para demonstração.

Código
Power_on ->

Conectores: -> ( & | ) ? : :

Limpar Guardar

Lista de Testes

- Send_packet
- Fail_Error
- Power_on
- Power_off
- Set_Tune_fm_frequency
  - Procurar uma frequência válida na banda FM no emulador
- Fm_seek_right
- AMFM_RX_set_Volume
- Check_fm_signal_quality
- Get_fm_SNR_value
- Get_fm_RSSI_value
Frontend - UI

**Lista de Pacotes**

- Pacote AM FM
- Pacote Exemplo
- Pacote Completo
- Pacote Loopback
- Pacote BroadR-Reach
- Pacote Sequência
- Pacote Condicional

**Criar Novo Pacote**

- **Nome**: Novo pacote
- **Descrição**: Novo pacote, apenas para demonstração.
- **Código**: Power_on ->

**Conectores**: -> ( & | ) ? : :

**Listando Testes**

- Send_packet
- Fail_Error
- Power_on
- Power_off
- Set_Tune_fm_frequency
- Fm_seek_right
- AMFM_RX_set_Volume
- Check_fm_signal_quality
- Get_fm_SNR_value
- Get_fm_RSSI_value

**Remover**  **Editar**
Conclusions and Future Work

● New modular and extensible architecture for self-diagnosis tests systems
  ○ Keyword Driven Testing Methodology + Domain Specific Language
  ○ Can perform any of test
● Architecture to extend and integrate the self-diagnosis tests system into a cyber-physical system
● Implement a self-diagnosis tests system to be integrated into a cyber-physical system

● Drag and drop window to the design of new test suites
Development of a Self-diagnosis Tests System Using a DSL for Creating New Test Suites for Integration in a Cyber-physical System

Ricardo B. Pereira, José C. Ramalho, Miguel A. Brito
ricardo-97-pereira@hotmail.com, jcr@di.uminho.pt, mab@dsi.uminho.pt

10th Symposium on Languages, Applications and Technologies