Model-Driven Tool Integration (MDT):
Telelogic DOORS® with …

Prof. Dr. rer. nat. Andy Schürr
andy.schuerr@es.tu-darmstadt.de

Technische Universität Darmstadt
Dept. of Electrical Engineering & Information Technology
(Associate Member of Department of Computer Science)
The Motivating Example

Automotive System Development

System requirements
SW-Functionality
HW-Design
ECU-Housing
Product Data
Function Test

Real-Time System Lab
MDT Development Scenarios

- Analysis and “Live” Modification of Local Tool Data:
  - System requirements: e.g. DOORS

- Bidirectional “Live” Tool Integration (Checking/Translation):
  - Delta
    - Datif
    - Mate
  - System requirements: e.g. DOORS
  - Software models: e.g. Matlab/Simulink
1. The DELTA Project

(DOORS Environment with traceability Linking and Tool integration API)

(DaimlerChrysler, Model Engineering Solution, TU Darmstadt)
Application Scenarios

• Analysis and “Live” Modification of Tool Data:

  - system requirements
e.g.: DOORS

  - software models
e.g.: Matlab/Simulink

• Bidirectional “Live” Tool Integration:

  - system requirements
e.g.: DOORS

  - software models
e.g.: Matlab/Simulink
• Instantiate requirements definition patterns (e.g. Use Case style)

• Check local completeness of requirements definitions (based on templates)

• Propagate local changes (e.g. referenced use case names)

• Export to / import from XML

• …
### 1 Use Case

#### 1.1 Detect Cruise Control Lever Position

**1.1.1 Attributes**

This use case describes how to compute changes at the cruise control lever.

**1.1.2 Description**

The cruise control lever which is actuated by the driver has three buttons: ON, OFF, and RESUME that control the activity of speed control.

- ON: pressing this button activates cruise control.
- OFF: by pressing this button the cruise control is switched off in any driving and operating condition.
- RESUME: pressing this button when cruise control is deactivated, reactivates cruise control with the last memorized speed.

**1.1.3 Precondition**

Ignition have to be activated.

**1.1.4 Postcondition**

The driver is able to choose another cruise control function.

**1.1.5 Category**

- high priority

**1.1.2 Includes**

**1.1.3 Extends**

**1.1.4 Actors**

- 1.1.4.1 Driver
Model Analysis and Transformation

- DOORS Database
- API Adapter
- DOORS Objects
- Model Analyser
- Model Transformer
- Analysis Results

MOF LON Version 0.9

Real-Time System Lab
DOORS Adapter Requirements

• Read/write access to tool data for modern object-oriented programming language (Java)

• Strongly typed tool data interface as well as untyped reflective interface

• Usage of an accepted meta modeling standard (MOF2.0 standard of the OMG)

• Usage of an accepted data repository API standard (JMI JSR040, Sun’s Java Meta Data Interface standard)

• Export / import of tool data using standard format (XMI standard of the OMG)
DOORS Adapter Architecture

- Metamodel of Tool Data
- Generated Interfaces
- Implementation Layer – JMI CAPI Bridge
- JMI Compliant Tool Adapter
- C++
- Generated XMI Representation
- XMI Reader/Writer
- Transformation Code (3rd Party)

MOF2.0 - MOFLON

Telelogic DOORS®

- DXL Server
- DXL Scripts
- DXL

Data Server

MOF2.0 - MOFLON

Real-Time System Lab
DXL Based Tool Meta Model

cut-out of MOF 2.0 meta mode
(Meta) Model Definition with MOFLON
Generated JMI Compliant API

Generated Code Structure
Excursion JMI

MOF Meta Model

JMI - Java Mapping

OutermostPackage

A

AKnowsB

B

RefPackage

RefClass

RefObject

RefAssociation
JMI Adapter Overhead

Adapter vs. DXL

Project
Formal Module
Formal Object
Attribute

Adapter vs. DXL

Adapter
DXL

Prozent
0
10
20
30
40
50
60
70
80
90
100
110
120

Project
Formal Module
Formal Object
Attribute
2. The MATE Project

(MATLAB Simulink & Stateflow Analysis and Transformation Environment)

(CARMEQ, Daimler AG, Model Engineering Solution, TU Darmstadt, University of Paderborn, … )
MDT Application Scenarios

- Analysis and “Live” Modification of Local Tool Data:
  - System requirements: e.g.: DOORS
  - Software models: e.g.: Matlab/Simulink

- Bidirectional “Live” Tool Integration (Checking/Translation):
  - System requirements: e.g.: DOORS
  - Software models: e.g.: Matlab/Simulink
Matlab Simulink/Stateflow

- Model-driven development of electronic control unit (ECU) software
- Specification of control algorithms
- Executable block diagram and statechart models
Some Standard Guidelines

GL1: The name of a subsystem consists of characters, digits, and _ ; it starts with a character …

GL2: The name of an Enable Block should be the same as the name of the related Enable Signal

GL3: Blocks may not possess unconnected Inports or unconnected Outports

GL4: …
function f_block_h = guideline_2(system, cmd_s)
    top_h = get_param(bdroot,'Handle');
    f_block_h = [];
    subsys = get_param(find_system(top_h, 'BlockType', 'EnablePort'), 'Parent'), 'Handle');
    for k=1:length(subsys)
        subsys_handle = get_param(subsys{k},'Handle');
        porth = get_param(subsys{k},'PortHandles');
        enable_port_name = get_param(porth.Enable,'Name');
        enableh = find_system(subsys{k},'SearchDepth',1,'BlockType','EnablePort');
        enable_block_name = get_param(enableh,'Name');
        if ~(strcmp(enable_port_name, enable_block_name))
            f_block_h = [f_block_h;subsys_handle];
        end
    end % for
end % function
Enable Signal and EnableBlock must have the same name!

```
Analyzer::analyseAndRepair_gl2 (enableBlock: EnableBlock): Void

enableSignal : Line
dstPort == "enable"

enableBlock
qualifiedName != enableSignal.qualifiedName
```

Check and repair action
3. The DATIF Project

(Darmstadt Tool Integration Framework)
MDT Application Scenarios

• Analysis and “Live” Modification of Local Tool Data:
  - system requirements
    e.g.: DOORS
  - software models
    e.g.: Matlab/Simulink

• Bidirectional “Live” Tool Integration (Checking/Translation):
  - system requirements
    e.g.: DOORS
  - software models
    e.g.: Matlab/Simulink
DOORS – MATLAB Integration Scenario

DOORS

Traceability Links

MATLAB Simulink

1. Automobile Comfort System
   - 1.1 Ice Warner
     - The Ice Warner detects the driver if the temperature at the outside drops below 3°C.
   - 1.2 Seat Belt Warner
     - The Seat Belt Warner informs the driver if any passenger has not fastened his or her seat belt.
   - 1.3 Headlight Recommendation
     - The Headlight Recommendation informs the driver if there should be the headlights due to low light conditions outside.
   - 1.4 Windscreen Wiper Assistant
     - Activates the Windscreen Wipers, if any passenger has not fastened his or her seat belt.

keep consistent
MOFLON Model Integration Scenario

- DOORS Model
  - (Un)Parser Definition
  - (Un)Parser API Adapter
  - DOORS Objects
  - Model Synchronizer
  - MATLAB Objects
  - (Un)Parser Definition
  - MOF Meta Model

- MATLAB Model
  - (Un)Parser API Adapter

- Analysis Results
  - Model Analysis & Refactoring

- OCL Constraints
  - SDM Graph Transformation
    - TGG (QVT-like) Translations
      - generates

- Code-Fragment Modell
- MOFLON Model Integration
- MOFLON Model Integration Scenario
- DOORS Model
- MATLAB Model
- (Un)Parser Definition
- MOF Meta Model
- Analysis Results
- Model Synchronizer
- MATLAB Objects
- (Un)Parser Definition
- MOF Meta Model
- OCL Constraints
- SDM Graph Transformation
- TGG (QVT-like) Translations
- generates
Connecting Meta-Models

DOORS Meta Model

+ Matlab Meta Model
TGG Mapping Rules (QVT-like)

Create Module and Model (n: String):

```
create Module
name := n

create ModuleRealizedByModel

create Model
name := n
```
addNestedRequirementAndSubsystem (n: String)

- r2:Requirement
  - heading := n
- r4:RequirementRealizedBySubsystem
- r5:RequirementRealizedBySubsystem
- s2:Subsystem
- s3:Subsystem

TGG Mapping Rules (QVT-like)
1st step: identify existing context

Generated Forward Transformation Rule

RequirementRealizedBySubsystem::performRequirementTransformation (r: Requirement): Void

r4 : Requirement

r : Requirement

re6 : RequirementRealizedBySubsystem

s4 : Subsystem
Generated Forward Transformation Rule

2nd step: translate/create new elements

```
RequirementRealizedBySubsystem::performRequirementTransformation (r: Requirement): Void
```

- **r4**: Requirement
- **re6**: RequirementRealizedBySubsystem
- **s4**: Subsystem
  - «create»
- **re7**: RequirementRealizedBySubsystem
- **s5**: Subsystem
  - «create»
  - name := r5.name
4. Conclusions
OMG Standards + Graph Transformation

- Velocity, XSLT Code Generator
- OCL 2.x Constraint Def. Language
- TGG (QVT Subset) M2M Translation Def. Language
- MOF 2.x Metamodel Def. Language
- DiaMeta Editor Generator
- SDM Fujaba Graph Transformations

- TU Dresden
- TU Darmstadt
- University of Kassel
- University of Paderborn
- University BW Munich

Real-Time System Lab
MDT Development Scenarios

• Analysis and “Live” Modification of Local Tool Data:

  - system requirements
    e.g.: DOORS

  \[\text{Delta}\]

• Bidirectional “Live” Tool Integration (Checking/Translation):

  - Datif
  - Mate

  - system requirements
    e.g.: DOORS

  - software models
    e.g.: Matlab/Simulink
- Marketing & Support & Vertrieb (Dr. Ingo Stürmer)

- Research & Development (Prof. Dr. Andy Schürr)