# Model-Based Debugging of Embedded Software Systems

**Padma Iyenghar**,
Elke Pulvermueller, Clemens Westerkamp,
Michael Uelschen and Juergen Wuebbelmann

**Project - funded by**

**Federal Ministry of Economics and Technology (BMWi), Germany**

&

**Industrial Partner**

**Willert Software Tools GmbH**

# Outline

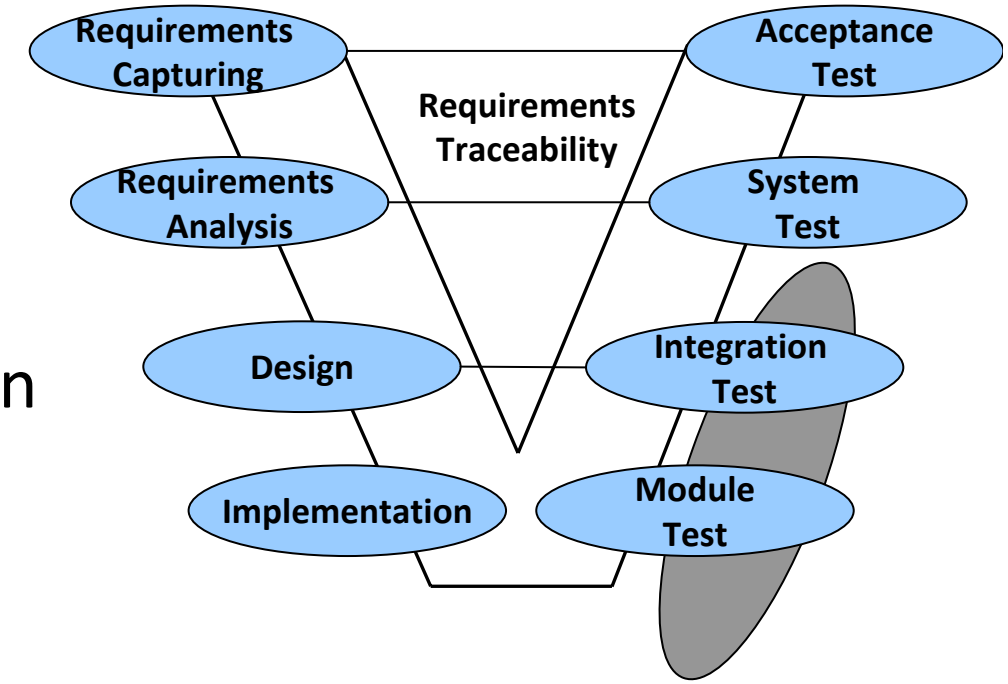
1. Introduction
2. Model-Based Debugging Approach- Concept
3. Model-Based Debugging – Prototype
4. Illustrative Example
5. Performance metrics
6. Summary and Conclusion

# Outline

1. Introduction
2. Model-Based Debugging Approach- Concept
3. Model-Based Debugging – Prototype
4. Illustrative Example
5. Performance metrics
6. Summary and Conclusion

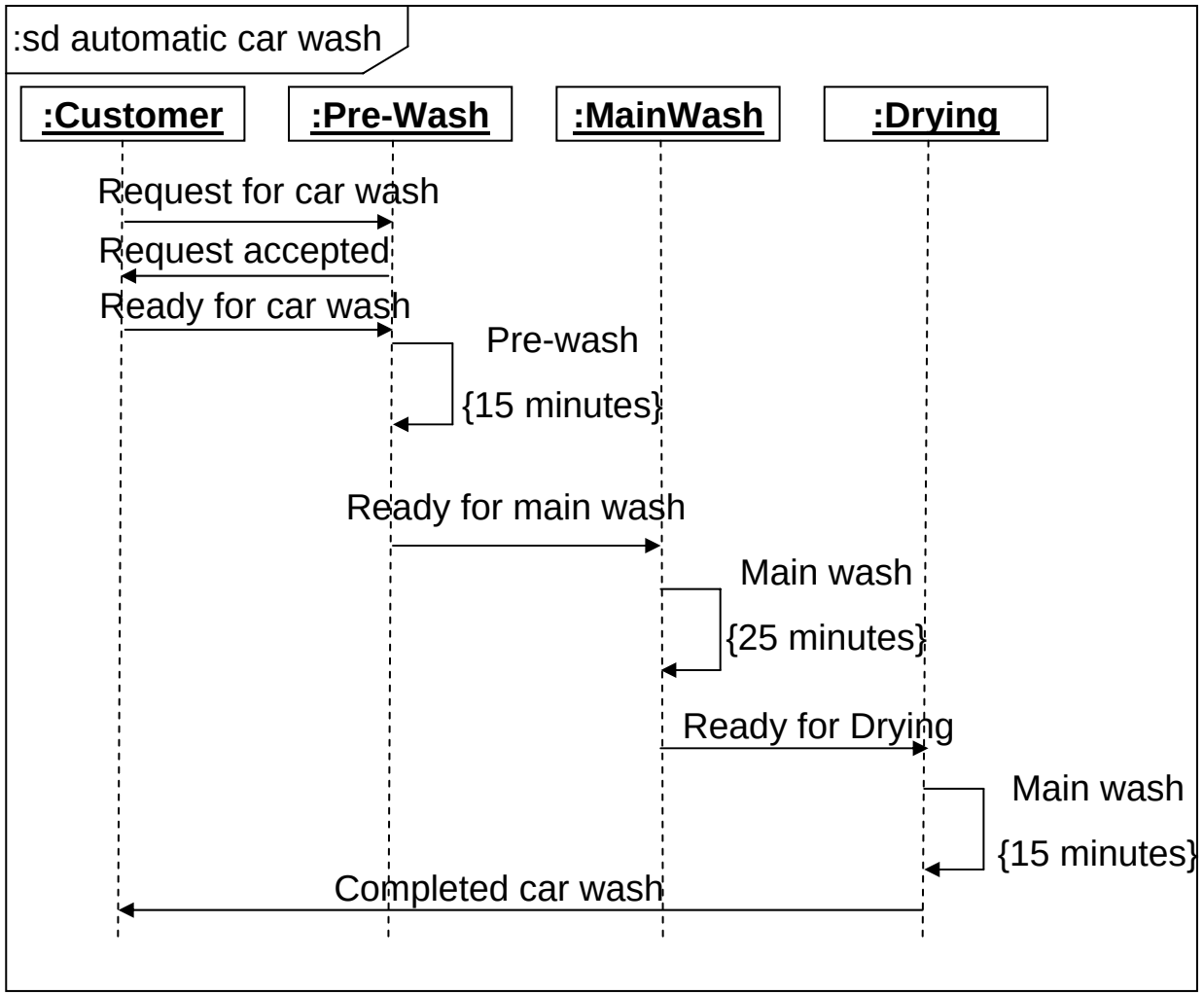
# Introduction

- Embedded Systems – Ubiquitous
  - Memory size, speed and real-time constraints
- Traditional approach
  - Debugging methods & tools
- Model Driven Architecture (e.g. UML diagrams in design model)
  - Model Driven Development (MDD)
  - Design-level debugging (visualize target behavior) → UML sequence and timing diagrams

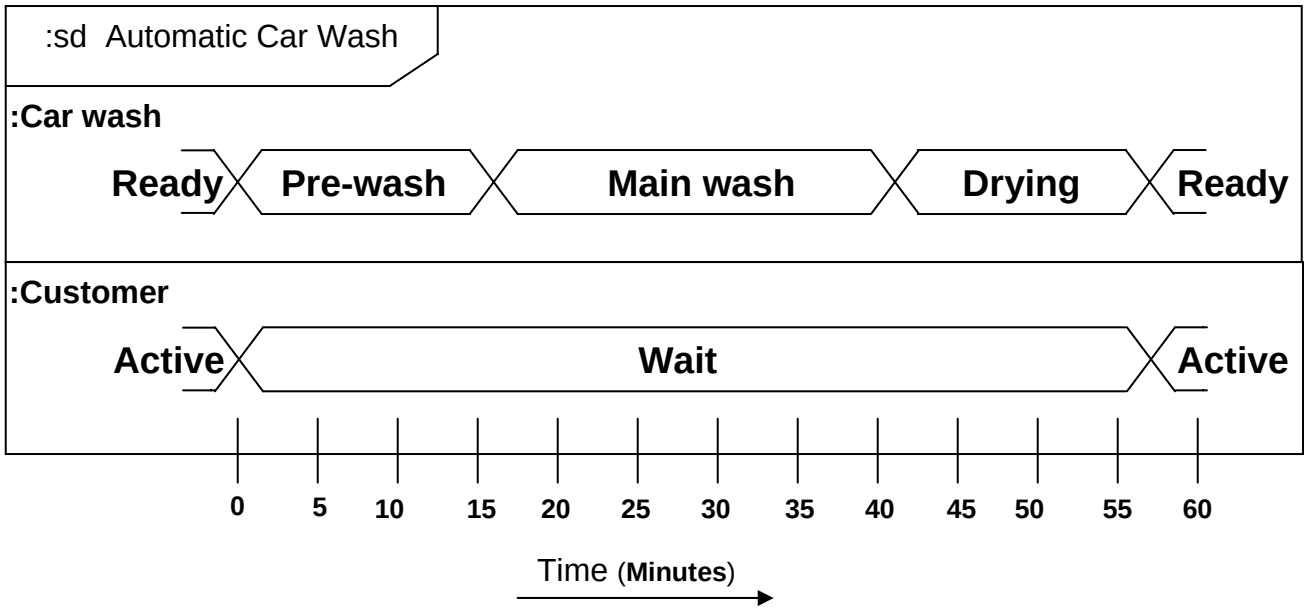



**Debugging**
**(Data monitor, instrumentation trace, debug breakpoints, etc)**

# UML Interaction Diagrams – Sequence diagram



# UML Interaction Diagrams – Timing diagram



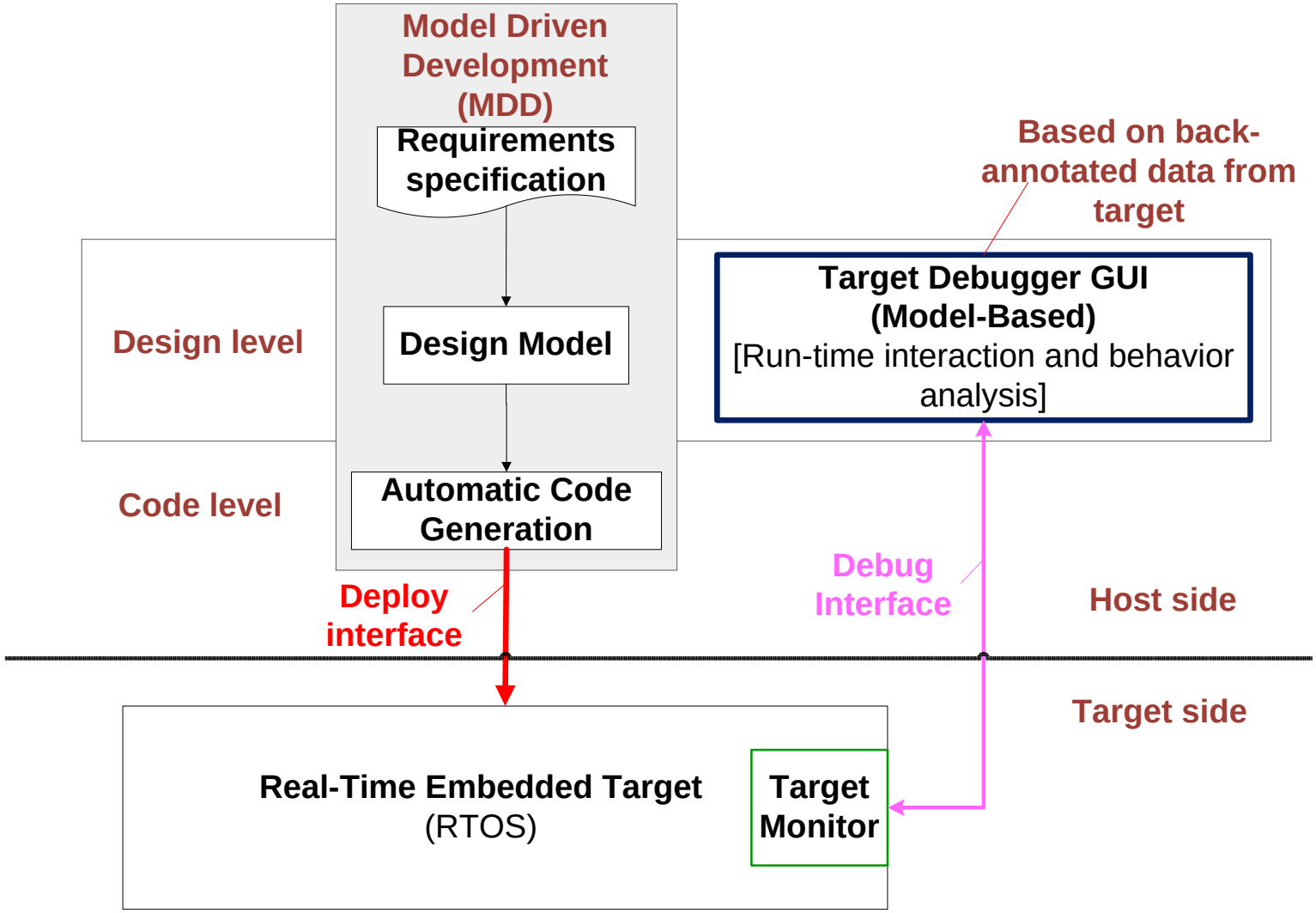
# Introduction

- Existing approach: Dynamic source code instrumentation
  - E.g. Rhapsody (MDD tool): suitable for large systems
- Required: Robust model-based debugging with MDD
  - Minimal overhead
  - Possibility to leave debug code in production code
  - Visualize target behavior in real-time  even for small platforms
- Proposed: Design-level debugging approach
  - Minimally intrusive
  - Enhanced UML diagrams with real-time information
    - Time annotated UML sequence diagram
    - UML timing diagram

# Outline

1. Introduction
2. Model-Based Debugging Approach- Concept
3. Model-Based Debugging – Prototype
4. Illustrative Example
5. Performance metrics
6. Summary and Conclusion

# Model-Based Debugging Approach - Concept (1)



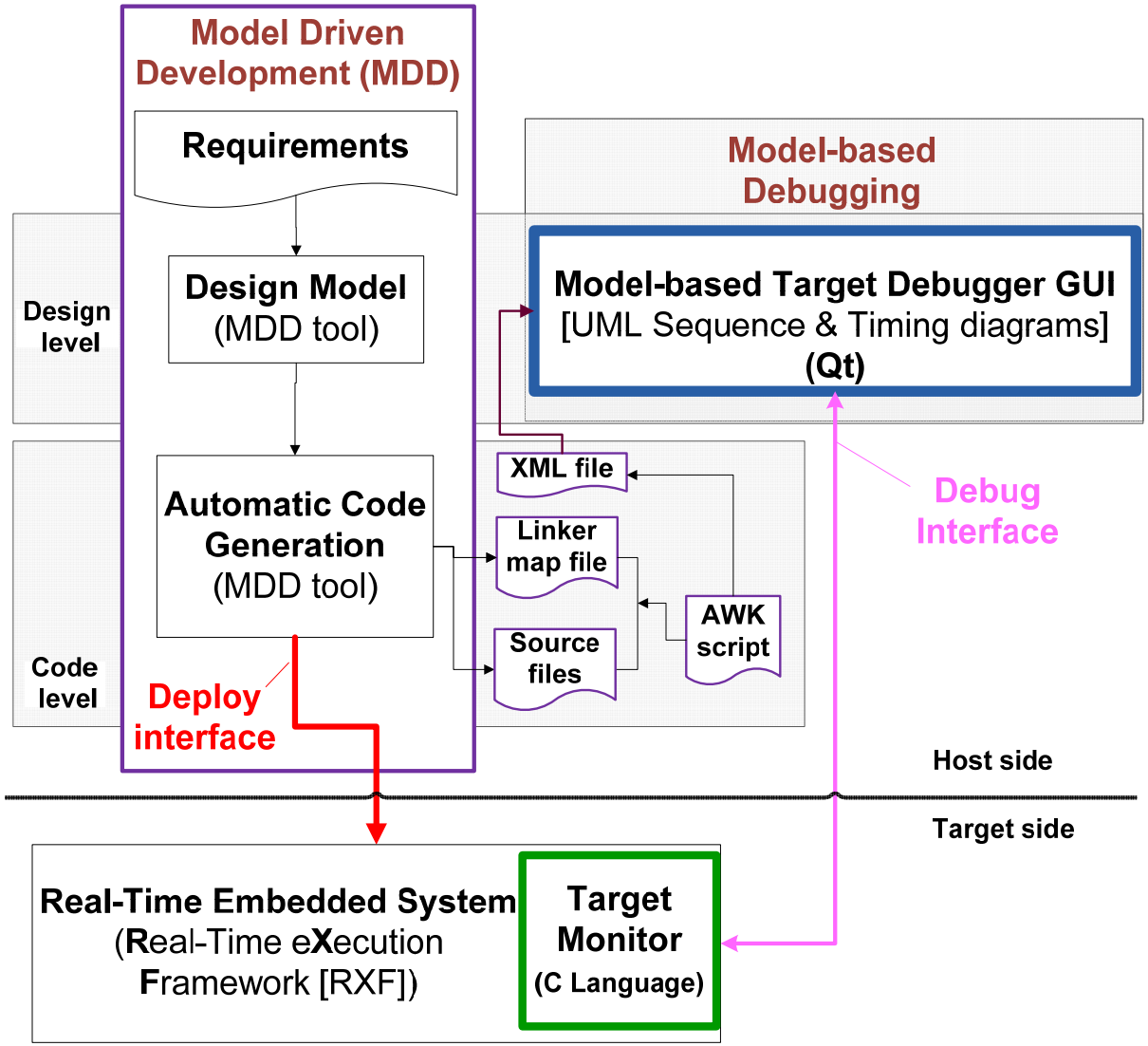
# Model-Based Debugging Approach- Concept (2)

- Idea
  - Host Computer
    - Extract A Priori knowledge from code generation
    - Source, header files, linker map file, etc...
    - Build symbol table references, store in intermediary format (e.g. XML)
    - Decode trace data from target
  - Embedded System
    - Target monitor: Sends data about target behavior using pre-defined protocol format → reduce overhead

# Outline

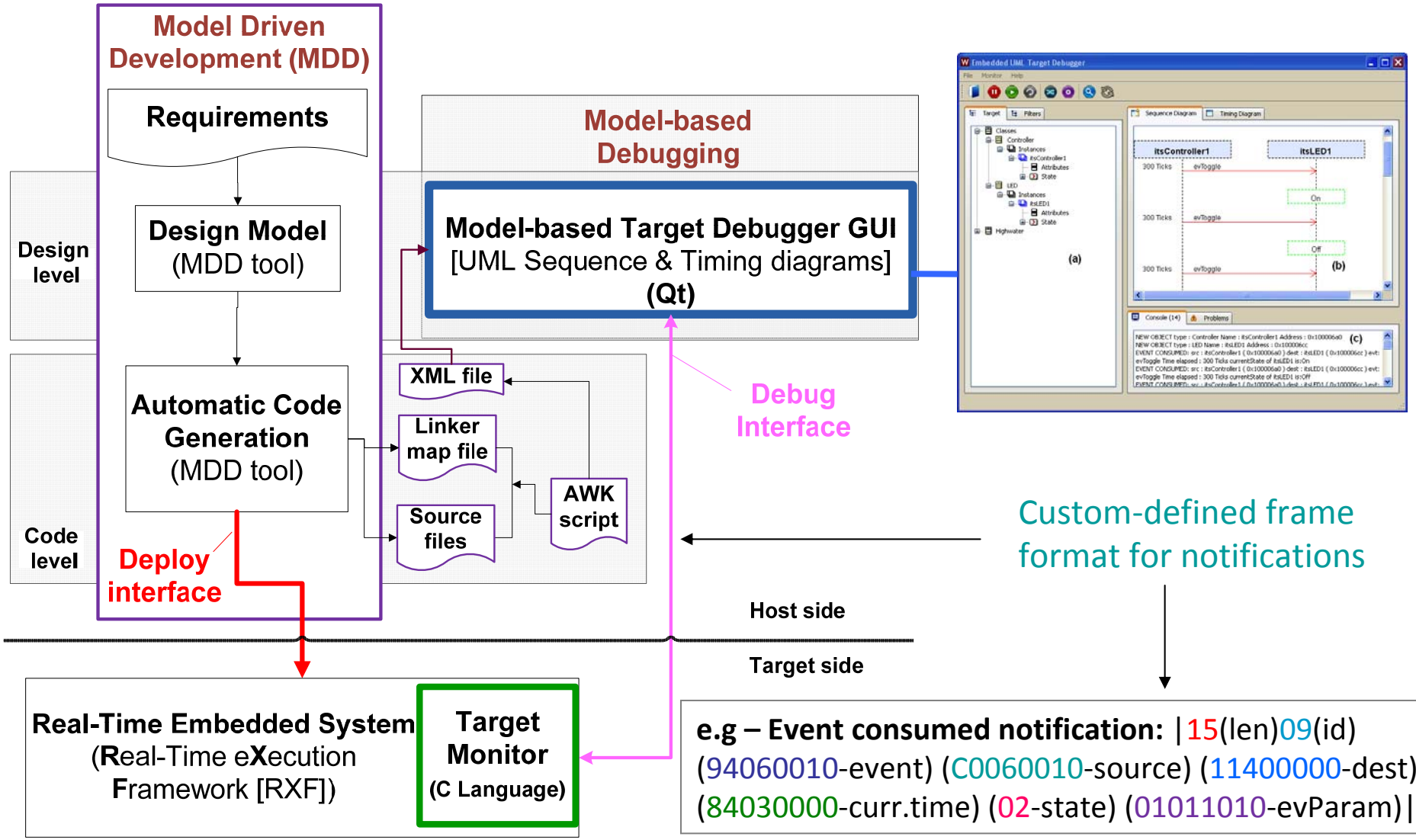
1. Introduction
2. Model-Based Debugging Approach- Concept
3. Model-Based Debugging – Prototype
4. Illustrative Example
5. Performance metrics
6. Summary and Conclusion

# Model-Based Debugging Approach – Prototype (1)




- RXF (RTOS): scheduler to handle events
  - Target monitor invoked by RXF.
  - Notified about consumed events.
- Communication overhead: XML file (intermediate format).

# Model-Based Debugging Approach – Prototype (2)

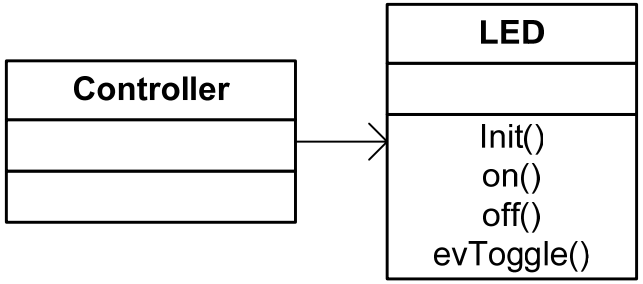


# Outline

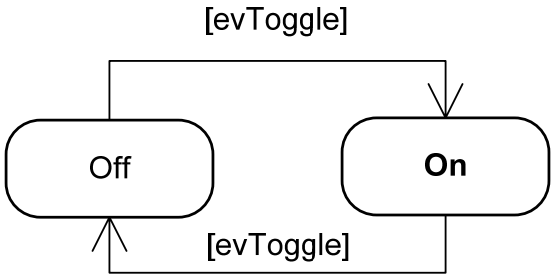
1. Introduction
2. Model-Based Debugging Approach- Concept
3. Model-Based Debugging – Prototype
4. Illustrative Example
5. Performance metrics
6. Summary and Conclusion

# Illustrative Example (1) – LED toggling

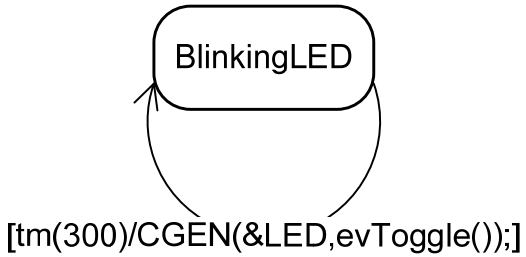
## Class diagram – *Controller* driving an *LED*



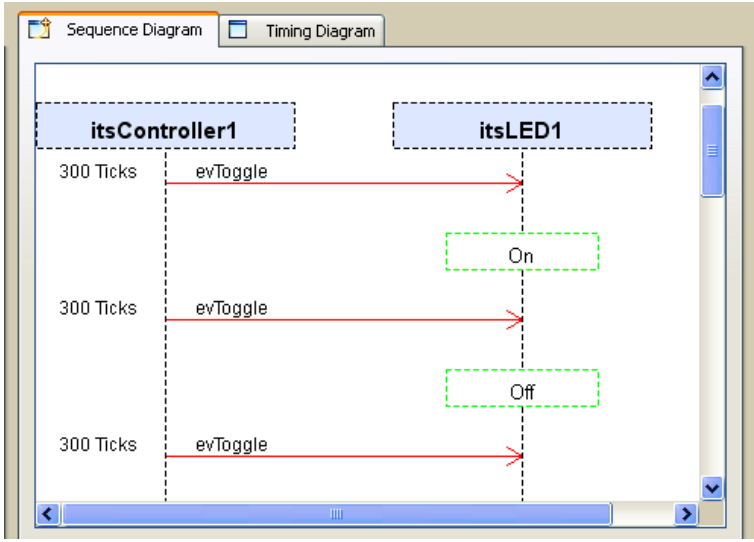

## State chart of *Controller* class



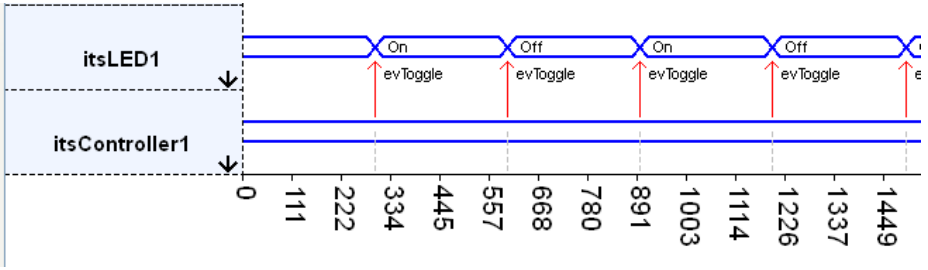

## State chart of *LED* class




## Target debugger GUI – Sequence diagram




## Target debugger GUI – Timing diagram



# Illustrative Example (2) – MIDI System Analyzer

MIDI Keyboard

MIDI

MIDI Interface

RS-232

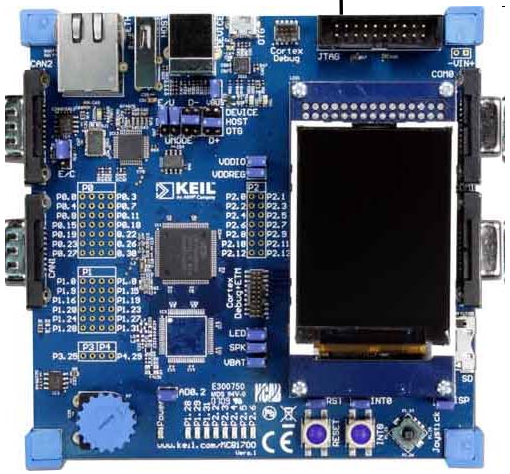
USB

Deploy interface

Host Computer

JTAG

RS-232



RS-232

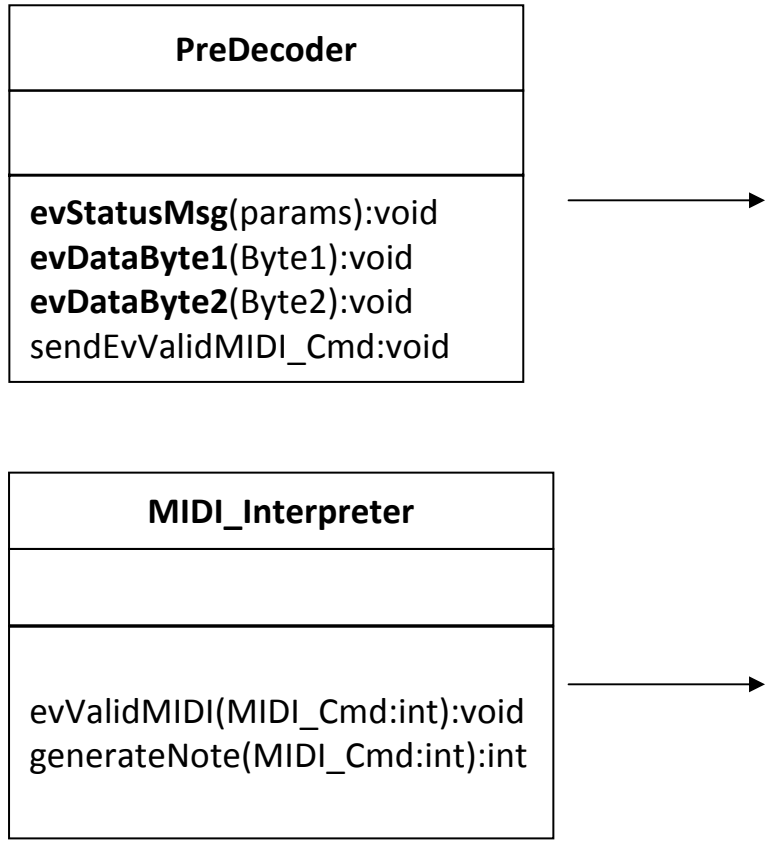
Debug interface

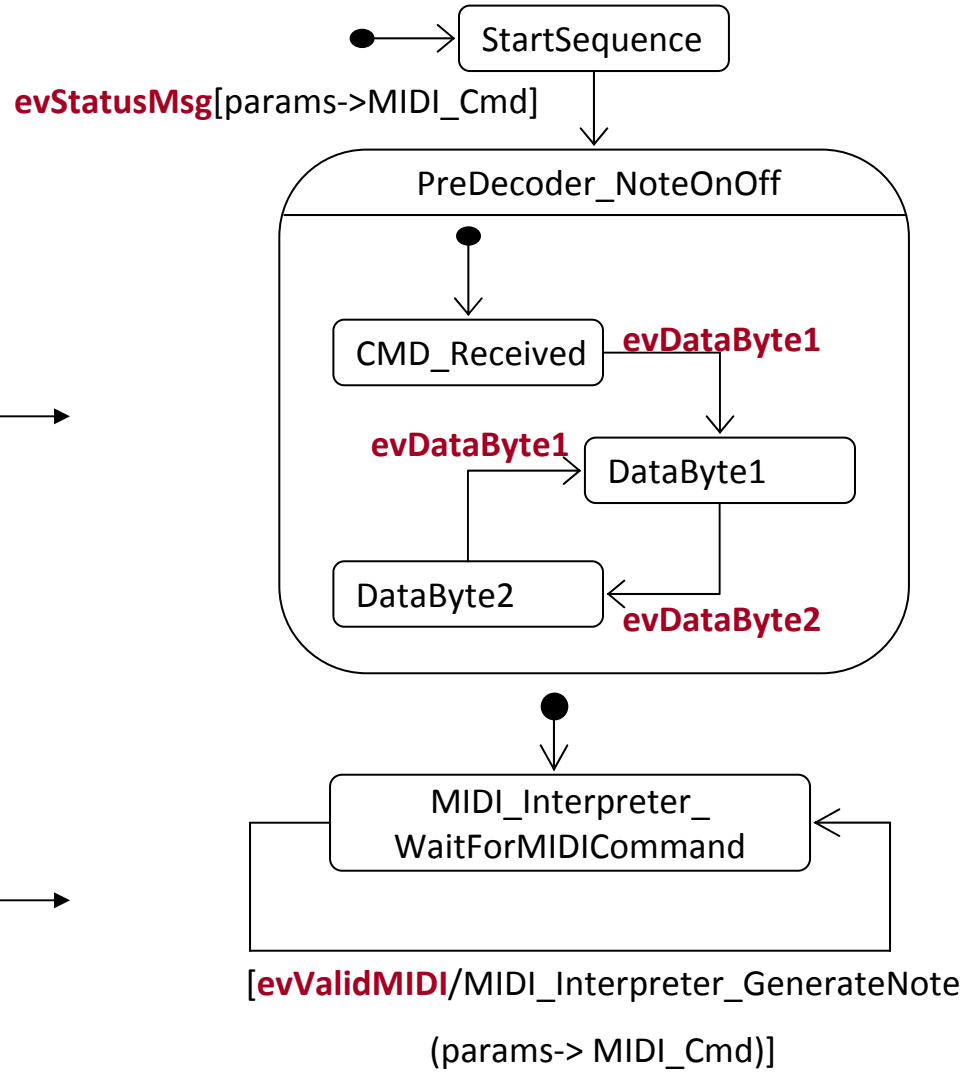
Evaluation Board (Cortex M3)

# MIDI System Case Study – Design Model (subset)

e.g. MIDI message format:

| Command byte | Key Number | Attack Velocity |
|---|---|---|

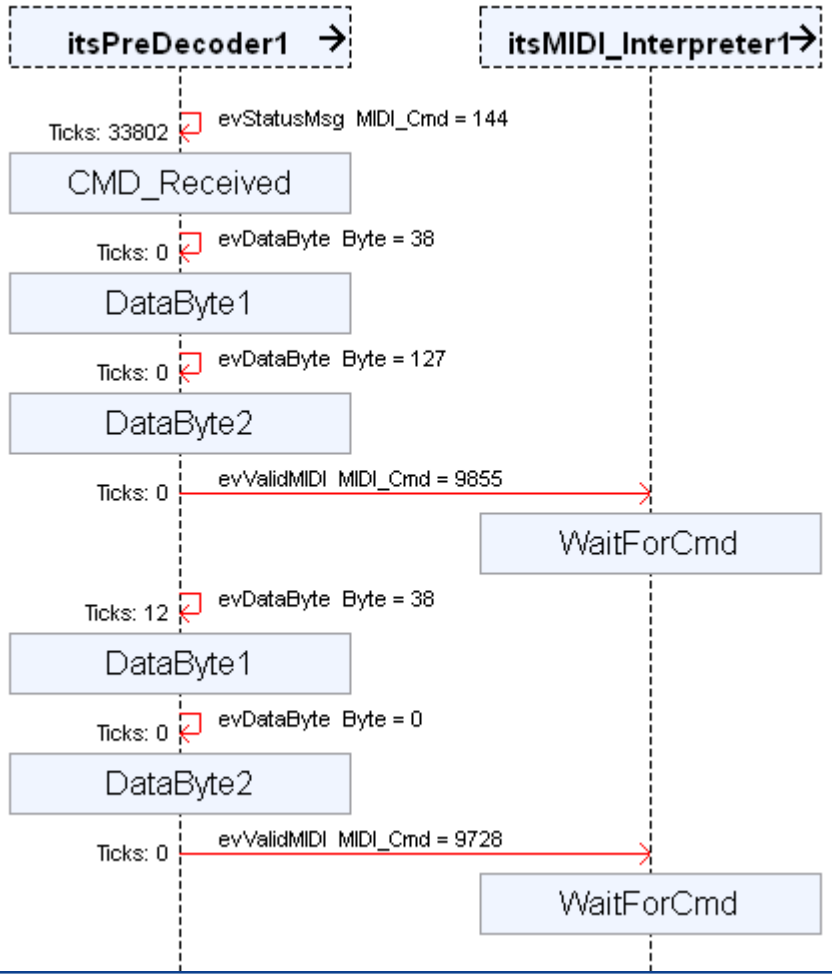


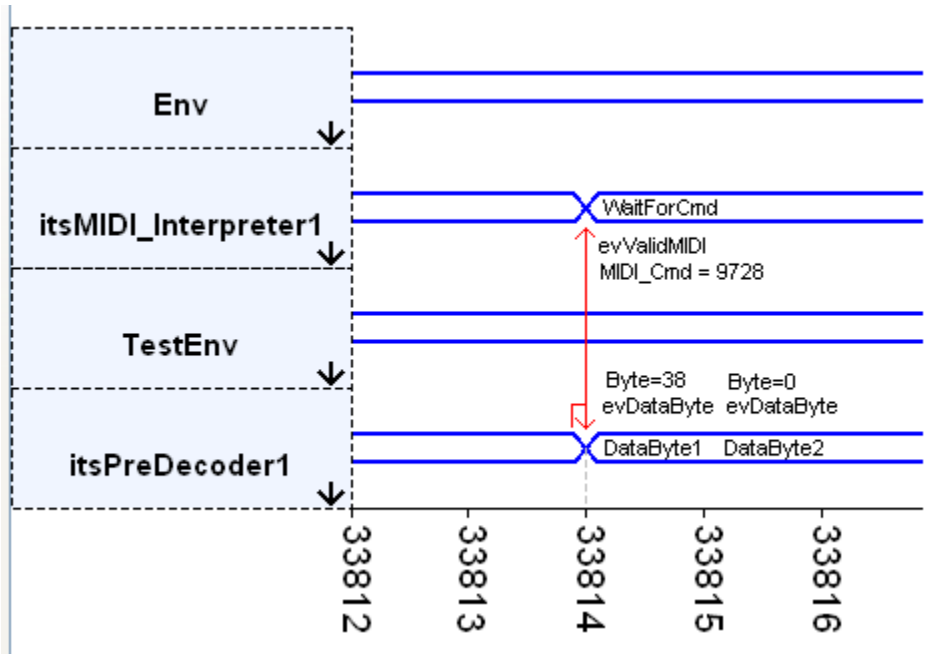
# MIDI System Case Study – Results

## Target Debugger – MIDI system behavior

### UML Sequence Diagram




### UML Timing Diagram



# Outline

1. Introduction
2. Model-Based Debugging Approach- Concept
3. Model-Based Debugging – Prototype
4. Illustrative Example
5. Performance metrics
6. Summary and Conclusion

# Performance Metrics (1)

Experimental Setup

| | |
|---|---|
| **Evaluation board** | MCB1700 [1] |
| **Microcontroller family** | Cortex-M3 (LPC1768) |
| **Max. Clock frequency** | 100 MHz |
| **RAM/ROM size** | 64 Kbyte/512 Kbyte |
| **Debug Interface(s)** | EIA-232 (Generic), µVision (JTAG), Trace32 (JTAG) |

Target Monitor memory footprint for debug interfaces

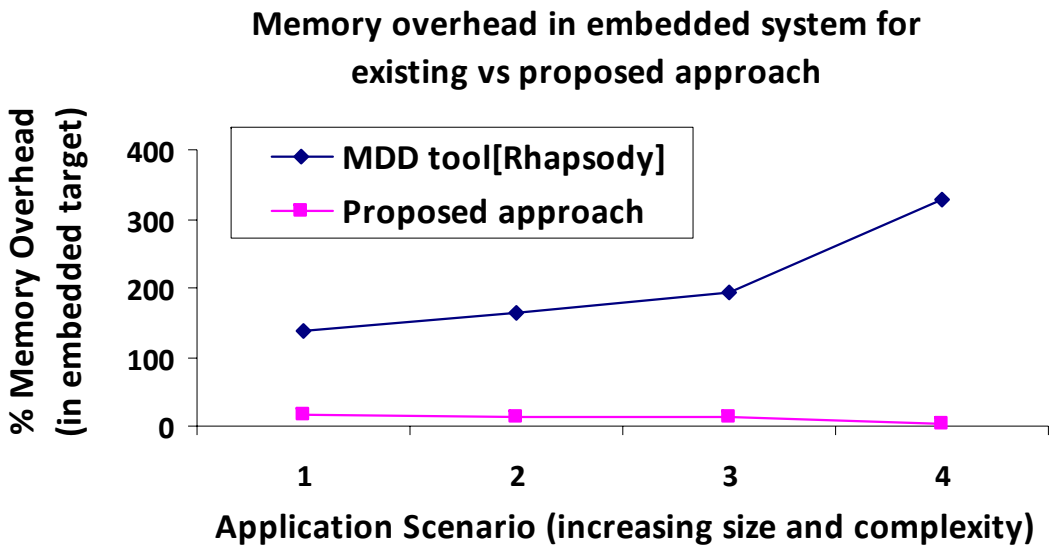
| Debug Interface | ROM (bytes) | RAM (bytes) | Total (bytes) |
|---|---|---|---|
| **EIA-232** | 1856 | 48 | 1904 |
| **µVision (JTAG)** | 1222 | 28 | 1250 |
| **Trace 3 (JTAG)** | 1656 | 20 | 1676 |

Time spent in target monitor (per event consumed notification)

| Debug Interface | Time spent in monitor routine [µs] |
|---|---|
| EIA-232 | 50 |
| µVision (JTAG) | 265 |
| Trace 32 (JTAG) | 7 |

# Performance Metrics (2)

Memory Overhead in Embedded System




**Legend**

| Application Scenario | Number of classes |
|---|---|
| 1 | 4 |
| 2 | 6 |
| 3 | 8 |
| 4 | 20 |

# Outline

1. Introduction
2. Model-Based Debugging Approach- Concept
3. Model-Based Debugging – Prototype
4. Illustrative Example
5. Performance metrics
6. Summary and Conclusion

# Summary and Conclusion

- Model-based, design level debugging approach
  - Target Monitor with static (constant) overhead
    - Memory size (approx. 1 Kbyte) accomodative for small targets
    - Time (inside monitor) known before hand - can be included in system design
    - Bundle with production code (end user‘s decision)
    - Opportunity to debug small targets at design level (UML)
  - Future Work: Support for additional target platforms
  - Further Application(s): Deploying/executing Model-Based Testing (MBT) in small (resource-constrained) embedded targets.

# Thank You!

## Contact :

Padma Iyenghar[1,2],
Elke Pulvermueller[2], Clemens Westerkamp[1],
Michael Uelschen[2] & Juergen Wuebbelmann[1]

1- University of Applied Sciences, Osnabrueck, Germany
2- University of Osnabrueck, Germany

{piyengha, elke.pulvermueller}@uos.de
{c.westerkamp,j.wuebbelmann, m.uelschen}@hs-osnabrueck.de

**Willert Software Tools GmbH**
http://www.willert.de

# Additional Information: Debug Interfaces/Target Systems

- Debug Interfaces
  - **Serial / UART**
  - **µVision – JTAG based (Keil)**
  - **Trace32 – JTAG based (Lauterbach)**

- Target System
  - **Evaluation boards**
    - **MCB2140 → ARM7(LPC2148)**
    - **MCB1700 → Cortex-M3(LPC1768)**