Firmware Synthesis for Ultra-Thin IoT Devices Based on Model Integration

Arthur Kühlwein\textsuperscript{1}, Anton Paule\textsuperscript{1}, Leon Hielscher\textsuperscript{1}, Wolfgang Rosenstiel\textsuperscript{2}, and Oliver Bringmann\textsuperscript{2}

\textsuperscript{1} FZI Research Center for Information Technology
\textsuperscript{2} University of Tübingen

International Workshop on Modeling Language Engineering and Execution 2019
Introduction & Motivation

- FW for ultra-thin IoT devices challenging to develop
  - Resource constraints (Power, memory...)
  - Extensive FW functionalities (RT computing, security, safety, ...)
  - Market pressure (Short time-to-market)
- MD approaches can tackle some of these issues, variety of different DSLs are used

Model-based activities (Transformations, Analyses, ...)

MM A (DSL 1)  
X

MM B (DSL 2)  
X

MM C (DSL 3)  

DSLs
Introduction & Motivation

- FW for ultra-thin IoT devices challenging to develop
  - Resource constraints (Power, memory...)
  - Extensive FW functionalities (RT computing, security, safety, ...)
  - Market pressure (Short time-to-market)
- MD approaches can tackle some of these issues, variety of different DSLs are used
Introduction & Motivation

⇒ No common interface between MMs
  • Difficult automation of FW development
  • Capabilities of MD activities limited by MM

⇒ Co-design & coordination challenging
  • HW/SW codesign common practice
  • Prolongs FW development cycle
  • Can lead to late detection of design errors
Introduction & Motivation

⇒ No common interface between MMs
  • Difficult automation of FW development
  • Capabilities of MD activities limited by MM

⇒ Co-design & coordination challenging
  • HW/SW codesign common practice
  • Prolongs FW development cycle
  • Can lead to late detection of design errors

Model-based activities (Transformations, Analyses, ...)

Unifying, Interoperable Modeling Language

DSLs

MM A (DSL 1) | MM B (DSL 2) | MM C (DSL 3)
**Introduction & Motivation**

- Exploit data synergies via common interface
  - Expand capabilities of MD activities

- Easier co-design & coordination
  - Shorter FW development cycle
  - Earlier detection of design errors

- Holistic approach to the automated synthesis of FW

---

Model-based activities (Transformations, Analyses, ...)

Unifying, Interoperable Modeling Language

MM A (DSL 1)  
MM B (DSL 2)  
MM C (DSL 3)

**DSLs**
Introduction & Motivation

⇒ Exploit data synergies via common interface
  • Expand capabilities of MD activities

⇒ Easier co-design & coordination
  • Shorter FW development cycle
  • Earlier detection of design errors

• Holistic approach to the automated synthesis of FW

Model-based activities (Transformations, Analyses, ...)

IoT Platform Modeling Language (IoT-PML)

MM A (DSL 1)  MM B (DSL 2)  MM C (DSL 3)

DSLs
The IoT-PML

- **Basic idea:** Capture essential concepts of related MMs
  - FRs, NFRs/NFPs of SW/HW platform
  - Device configurability
  - Usage scenarios

- Common abstractions of these concepts to enable effective integration and cooperation
  - Careful analysis necessary, as we do **not** want to create a gargantuan metamodel
  - Provide data exchange at model runtime via a model abstraction layer

- Support for top-down and bottom-up workflows
- MOF-conformant metamodel
  - Currently implemented as a UML profile
The IoT-PML – Architecture & Features

![Diagram of IoT-PML architecture and features]

- **Model Abstraction Layer (MAL)**
  - **LO**
    - Core
  - **L1**
    - NFP
    - GSM
  - **LN**
    - Registers
    - Impl.
    - ...

- **Metamodels**
  - MM A
  - MM B
  - MM C

- **Drivers**
  - Driver A
  - Driver B
  - Driver C
The IoT-PML – Architecture & Features

Modular
- **Module**: contains concepts common to a number of MMs
- **Concepts** have to specialize concepts of the Core module

Layered
- **Layer**: contains concepts at a particular abstraction level
- **L0** highest, **LN** lowest abstraction level
The IoT-PML – Architecture & Features

Core Module Metamodel

The diagram illustrates the core module of the IoT-PML metamodel, which includes entities such as `AnnotatableElement`, `Element`, `AnnotatableElement` with attributes like `linkedMRef [0..1]`, `annotatedElement 1..*`, `target 1..*`, `containedBlock *`, `container 0..1`, `source 1`, and `source 0..1`. The diagram also shows the relationship between these entities and the Model Abstraction Layer (MAL) of the IoT-PML, which includes layers `LO`, `L1`, `LN`, Core, NFP, GSM, Registers, Impl., and ... with connections to drivers such as Driver A, Driver B, Driver C, and metamodels MM A, MM B, MM C.
Model Linkage
• Each IoT-PML element can link to an external model element $e$ using a **model reference** $M : URI_e$
  - MM Identifier
• Linkage at model runtime facilitated by **Model Abstraction Layer (MAL)**
  - Module-specific interfaces, which are implemented by metamodel-specific drivers
The IoT-PML – Architecture & Features

- **Extensibility**
  - **User modules** can be added to the IoT-PML
    - New concepts, refinement of existing concepts
    - Extend MAL with corresponding interfaces
  - IoT-PML and MAL are constructed at model runtime by assembling built-in and user modules

![Diagram](image_url)
The IoT-PML – Implementation

- Currently implemented as a UML profile
  - Layer ↔ package, module ↔ (sub)profile, concept ↔ stereotype
  - Exploit UML for modeling SW aspects
  - Leverage large ecosystem of model-based technologies (M2M, M2T, ...) that evolved around OMG standards
  - Mature tooling support

- Realized using Eclipse-based frameworks and tools
  - EMF
  - Papyrus Modeling Environment
Use Case

- Code generation (and verification) of a driver for an IoT sensor device peripheral
Use Case

- Basic top-down workflow
  - Generate SW-centric IoT-PML model of peripheral driver

```
<sensor.xml>
<CommandRegister>
  <CommandReg descr="Command register" bitW 16 bitOffs 8>
    <value>0x0</value>
    <value>0x1</value>
    <writeOnly/>
  </CommandReg>
  <CommandReg descr="Sleep command" bitW 16 bitOffs 0>
    <value>0x0</value>
    <value>0x1</value>
    <writeOnly/>
  </CommandReg>
  <CommandReg descr="Wakeup command" bitW 16 bitOffs 0>
    <value>0x2</value>
    <value>0x3</value>
    <writeOnly/>
  </CommandReg>
<primitiveType descr="Temperature" fixedWidthType="Int16">
  <name>temp_raw</name>
</primitiveType>
<primitiveType descr="Humidity" fixedWidthType="Int16">
  <name>humid_raw</name>
</primitiveType>
```

17.09.2019 © FZI Forschungszentrum Informatik
Use Case

- Basic top-down workflow
  - Use templates to generate driver skeleton

Templates

```
tearDown()
```
Conclusions

- First concept of novel unifying modeling language for ultra-thin IoT device FW
  - Linking mechanism enables data exchange with external metamodels
  - MOF-conformant, currently implemented as a UML profile
- Language development and analysis of metamodels still ongoing
  - New external metamodel: device trees
- Rudimentary tool support
  - Currently working on Papyrus integration of MAL features
  - Need to keep data consistent between IoT-PML ↔ external model
- Generic mechanism to map IoT-PML concepts to arbitrary XSD-based metamodels
  - Automatic generation of MAL drivers
  - Could be extended to arbitrary metamodels (e.g. text-based)
Thank you for your attention!

Any questions?