Model-Based Design of Complex Embedded Systems

Ludovic Apvrille

ludovic.apvrille@telecom-paristech.fr

HDR Defense, Nov., the 29th, 2012
Outline

Introduction
  Context
  Model-Driven Engineering

Synthetic overview of contributions
  UML Profiles
  Overview
  TTool

Focus on a few contributions
  Partitioning
  Handling security
  Deployment

Conclusions and perspectives
  Conclusions
  Perspectives
Outline

Introduction
  Context
  Model-Driven Engineering

Synthetic overview of contributions

Focus on a few contributions

Conclusions and perspectives
Designing Embedded Systems

**How to Handle Complexity?**
Modeling and verification!
(But there are other options)
Modeling is not Really a New Technique... 

...and it is not limited to Software!
Software Development Techniques for Embedded Systems

**Code-based approaches**
- Extreme Programming
  - Strongly tested step-by-step code increments
- Agile Software Development
  - Focus on change in specification

**Model-based approaches**
- V-Cycle
  - KAOS, AADL, MDE, ...
  - Making message
  - Sending message
  - Chout(m)
  - m.data = secretData
  - m1 = sencrypt(m, sk)
- Formal models
  - B, LOTOS, Petri nets, ...

Ludovic Apvrille
Habilitation à Diriger les Recherches
Model Driven Engineering

**Definition**

- Process based on abstract graphical representations for a given domain
- Intends to improve software engineering quality criteria
  - Reliability, extensibility, maintainability, ...
- Should enhance team communication and documentation

**Abstraction levels**

- Platform Independent Model, Platform Specific Model
- Model transformations
UML Profiles

**Definition**
- UML defined extension mechanisms to e.g.,
  - Define new operators
  - Provide a semantics
  - Give a methodology

**Example of profiles**
- Profiles defined by OMG (e.g., SPT, MARTE, SysML)
- Profiles defined by tool vendors (e.g., in Rhapsody, Artisan)
- User-defined and company-defined models
UML Profiles and MDE

UML profiles are a way to define domain-specific languages for MDE

Our contribution in MDE

Definition of UML profiles for modeling and verifying complex embedded systems

Definition of methodologies based on the V-cycle

Definition of model transformations for simulation, formal verification and code generation purpose

Implementation in a toolkit (TTool)
Outline

Introduction

Synthetic overview of contributions
  UML Profiles
  Overview
  TTool

Focus on a few contributions

Conclusions and perspectives

TURTLE: A Formally Defined UML Profile (Cont.)

- Developed in the scope of my Ph.D.
- Partners: Thalès Alenia Space, LAAS-CNRS, ISAE
- Software design: class and activity diagram
  - Communication based on synchronous exchanges
  - Non-deterministic choices
  - Non-deterministic time intervals
- Model transformation to RT-LOTOS
Extending TURTLE (2003-2010)

Extending TURTLE (Cont.)

- Developed in the scope of my post-doctorate and in LabSoC
- Collaboration with ISAE on Dimensioning stage
  - Network calculus toolkit for computing the Worst Case
  - Use case provided by Airbus
- Collaboration with ISAE and Concordia University for analysis and deployment stages
- Other partners / projects: LAAS-CNRS, UDCast, European project Maestro, ANR project Safecast, DoceaPower
- 1 Ph.D. completed (Benjamin Fontan, ISAE)
DIPLODOCUS: HW/SW Partitioning (2006-...)

DIPLODOCUS (Cont.)

- High abstraction level ("system-level")
- Can be used for Design Space Exploration
- Developed at LabSoC
- Partners: Texas Instruments, Freescale, European project EVITA, European project SACRA, LIP6
- 2 Ph.D. completed (Chafic Jaber, Daniel Knorreck), 3 on-going Ph.D. (Jair Gonzalez-Pina, Ferial Ben Abdallah, Andrea Enrici)
- Applied to:
  - Multimedia system, e.g., partitioning of smartphone platforms
  - Telecommunication systems, e.g., partitioning of LTE
Requirement modeling within SysML requirement diagrams
Graphical modeling of safety properties using SysML
Parametric Diagrams
  - Reachability, liveness
Handle logical and temporal constraints
2 Ph.D. completed (Benjamin Fontan, Daniel Knorreck)
Collaboration with ISAE
AVATAR: Taking Into Account Security (2010-...)

- Requirements
  - Safety, security
- Analysis
  - Sequence and interaction overview diagrams
- Design
  - Block and state machine diagrams
- Deployment
  - Performance, security
  - Deployment diagrams
- Simulation
  - Formal verification
  - TTool, UPPAAL
  - TTool, UPPAAL, ProVerif
- Test

UML Profiles
Overview
TTool
AVATAR

- Main idea: safety and security property proofs at the push of a button
- 1 Ph.D. completed (Muhammad Sabir Idrees, With Eurecom), 1 on-going Ph.D. (Gabriel Pedroza)
- Defined in the scope of the European project EVITA, Collaboration with ISAE and Eurecom
  - Used to model and prove security properties of cryptographic protocols for automotive systems
- Most TURTLE users have switched to AVATAR
- Widely used for educational purpose (Universities, training in companies)
Overview of Contributions

- Requirements
  - Dimensioning Partitioning
    - Simulation
      - Formal verification
  - Analysis
    - Simulation
      - Formal verification
  - Design
    - Simulation
      - Formal verification
  - Deployment
    - Test
TTool: A Multi Profile Platform

TTool

- Open-source toolkit mainly developed by Telecom ParisTech
- Multi-profile toolkit
  - DIPLODOCUS, AVATAR, ...
- Support from academic (e.g. INRIA, ISAE) and industrial partners (e.g., Freescale)

Main ideas

- Lightweight, easy-to-use toolkit
- Simulation with model animation
- Formal proof at the push of a button
Outline

Introduction

Synthetic overview of contributions

Focus on a few contributions
  Partitioning
  Handling security
  Deployment

Conclusions and perspectives
Partitioning with the Y-Methodology

Diagram:
- Partitioning
  - Simulation
    - Formal verification
  - Application model
  - Architecture model
  - Mapping model
    - Simulation
      - Formal verification
Tasks are mapped on execution nodes (e.g., CPUs, HWAs)
Channels are mapped on communication and storage nodes
After-Mapping Simulation

- TTool Built-in simulator
- Extremely fast
- Diagram animation
- Step-by-step execution, breakpoints, etc.
After-Mapping Simulation (Cont.)
After-Mapping Coverage-Enhanced Simulation

Possibility to select a given part of the model to be explored

- Minimum percentage of operators coverage
- Minimum percentage of branch coverage

Implementation: TTool built-in model-checking techniques
After-Mapping Formal Verification

- TTool built-in model checker can compute all possible execution paths
- Graph analysis and visualization
Secured Design with AVATAR
Design Features for Security

**Initial knowledge:** Introduced as a common knowledge of the system, or of a specific session of the system:

```
#InitialSystemKnowledge Alice.sk Bob.sk
```

**Cryptographic functions:** Predefined in each AVATAR block: MAC(), encrypt(), decrypt(), sign(), verifyMAC(), verifySign()...

**Communication Architecture:** Common public channels can be defined in blocks. Attackers can eavesdrop public channels

**Attacker model:** Taken from the underlying security framework ProVerif
Detailed Design

- Block’s behaviour is described in terms of SysML State Machine Diagrams
- Non deterministic choices, non deterministic temporal operators
Property Modeling

Safety properties

- Customized Parametric Diagrams (TEPE)
- Reachability, liveness

Security properties

- Based on basic pragmas
  - Confidentiality of a block attribute
  - Authenticity of interconnected block signals

```plaintext
#Confidentiality RemoteControl.duration
#Authenticity RemoteControl.SendingRemoteOrder.msg1
WirelessInterface.gotWirelessOrder.msg2
```
Model Transformation for Formal Verification
Formal Verification

- Push button approach, both for safety and security properties!

Safety properties
  - UPPAAL based

Security properties
  - ProVerif based

- Verify with UPPAAL: options
  - Search for absence of deadlock situations
  - Reachability of selected states
  - Liveness of selected states
  - Custom verification
  - Custom formulae = [input your CTL formula]

- Session id on launcher= 1
- Sending UPPAAL specification data

- Reachability of: ObserverProp1.state0: Error -> property is NOT satisfied
- All Done

- Execute ProVerif as
  - /packages/proverif/proverif -in pi

- Show output of ProVerif

- Confidential Data:
  - [output]

- Non Confidential Data:
  - [output]

- Satisfied Authenticity:
  - [output]

- WirelessInterface__gotWirelessOrder__msg2__data
Deployment

- Requirements
  - safety, security
- Analysis
  - Sequence and interaction overview diagrams
- Design
  - Block and state machine diagrams
- Simulation
  - Formal verification
    - TTool, UPPAAL
  - Formal verification
    - TTool, UPPAAL, ProVerif
- Requirements
  - performance, security
- Deployment
  - Deployment diagrams
- Test
Deployment (Cont.)

- Local platform
- Virtual prototyping platform (e.g., SoCLib)
- Target
Prototyping with SoCLib
Outline

Introduction

Synthetic overview of contributions

Focus on a few contributions

Conclusions and perspectives

Conclusions

Perspectives
Contributions

- Global approach for complex embedded systems based on Model Driven Engineering techniques
  - Safety-oriented Dimensioning, Analysis, Design, Deployment: TURTLE
  - Partitioning: DIPLODOCUS
  - Requirements and properties: TEPE
  - Safety and security: AVATAR

- Toolkit (TTool)

- Collaboration with academic and industrial partners
And So?

Embedded System

\[ \text{Tightly coupled hardware and software integration} \]

MDE is still a software-centric approach!

- Definitely not tightly coupled hardware and software methodology!
- Hardware is seen like a resource, i.e. it has no behaviour
- Same remark applies to intermediate layers (e.g., OS, middleware)
Hardware in MDE

A few techniques to handle hardware

- UML Deployment diagrams
- SysML / MARTE allocation mechanisms
- DIPLODOCUS mapping
- Platform Independent Model, Platform Specific Model
Methodological steps

1. System partitioning
2. Software-centric model-based methodology
   - Including "low level" layers, e.g. drivers, OS, middleware
3. Hardware-centric model-based methodology
4. Model-based hardware and software integration

MDHSE: Model Driven Hardware and Software Engineering
Meta-modeling for describing languages for all stages
MDE for Hardware Design: A Few Issues

- Abstraction levels?
  - Software: PIM, PSM
  - Hardware: Transaction level (TLM, CABA), RTL
- Time handling?
  - Time model of MARTE?
- Is UML adapted as a modeling language? If yes, is it adapted to all abstraction levels?
- Underlying simulation technologies, i.e., model transformation to SystemC, System Verilog, SocLib?
What About Security?

- AVATAR handles security from Analysis to Deployment
- Dimensioning, partitioning are not (yet) handled
Questions?

▶ http://perso.telecom-paristech.fr/~apvrille/
▶ http://ttool.telecom-paristech.fr/