AVATAR-T Tool
A SysML Environment for the Proof of Safety and Security Properties

Ludovic Apvrille
Telecom ParisTech
ludovic.apvrille@telecom-paristech.fr

April 2011
Outline

Introduction
  Model-Driven Engineering
  TTool
  DIPLODOCUS

AVATAR: From Requirements to Prototyping
  Introduction
  Requirements
  Analysis
  Design

Conclusions, References
  Conclusions
  References
Outline

Introduction

Model-Driven Engineering
TTool
DIPLODOCUS

AVATAR: From Requirements to Prototyping

Conclusions, References
Model Driven Engineering

**Definition**
- Process based on abstract representations for a given domain (domain model)
- Notion of patterns
- Should enhance team working and exchanges between clients/system-level teams and development teams

**UML and SysML**
- MDE is commonly based on UML profiles
  - Profiles defined at OMG’s (e.g., SPT, MARTE, SysML)
  - Profiles defined by tool vendors (e.g., in Rhapsody, Artisan)
  - User-defined and company-defined models
A Multi Profile Platform: TTool

**TTool**
- Open-source toolkit mainly developed by Telecom ParisTech
- 8 UML profiles
  - DIPLODOCUS, AVATAR
- Support from academic (e.g. INRIA) 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
The DIPLODOCUS UML Profile

- Partitioning
  - Finding the best SW / HW function repartition
- Follows the Y-Chart approach
- Ultra-fast simulation and verification
  - Up to 100 times the real-time execution
  - Variable simulation coverage
Outline

Introduction

AVATAR: From Requirements to Prototyping
  Introduction
  Requirements
  Analysis
  Design

Conclusions, References
AVATAR in a Nutshell

Former contribution: TURTLE (1999)
- Formally defined UML profile (RT-LOTOS, UPPAAL)

AVATAR (2010)
- SysML environment supporting all methodological phases
- Graphical capture of properties
- Integrated simulation
- Safety and **security** proofs at the push of a button
- C-POSIX code generation
- TURTLE is now deprecated!
Methodology

- Requirement capture
  - Requirement Diagrams
  - Attack trees
    - Parametric Diagrams

- Use cases and interfaces
  - Use Case Diagrams
  - Scenarios
    - Activity Diagrams
    - Sequence Diagrams

- Analysis

- Design
  - Design
    - Block Definition Diagram
    - Internal Block Diagram
  - Detailed design
    - State Machine Diagrams
  - Property modeling
    - Extended Parametric Diagrams
      - TEPE

- Simulation
- Formal verification
- Code generation
Requirement Capture

- **SysML Requirement Diagrams**
- **Specialization for security-related requirements (e.g., confidentiality, privacy, etc.)**
- **Modeling assumptions inside notes**

![Diagram](image)
Attack Trees

- Represent all possible attacks on the system
  - And relations between those attacks: OR, AND, SEQUENCE, BEFORE, AFTER, etc.
- SysML Parametric Diagrams
Use Cases

- System boundary, actors, and main functions (use cases) provided by the system
  - And high-level links between use cases
- SysML Use Case Diagrams
Main Functioning Modes

- Identify various system functioning modes
- Represent relations between functioning modes
  - Sequence, choice, preemption, parallel
- (Slightly extended) SysML Activity Diagrams
Scenarios

- Identify a specific trace of the system
- SysML Sequence Diagrams
Design: Architecture

- SysML Block Definition and Internal Block Diagrams
- Block = attributes, methods, in/out signals, behaviour
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)

Security properties

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

#Confidentiality RemoteControl.duration
#Authenticity RemoteControl.SendingRemoteOrder.msg1 WirelessInterface.gotWirelessOrder.msg2
#InitialCommonKnowledge RemoteControl.PSK WirelessInterface.PSK

Back to methodology
Simulation

- Integrated in TTool
- Model animation
- Breakpoints, step, backstep, reset, introspection of block variables, etc.
- Simulation traces are displayed as SysML Sequence Diagrams
Formal Verification

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

Safety properties
- UPPAAL based

Security properties
- ProVerif based
Prototyping

- C-POSIX code generation from design
  - Compiled and executed on localhost (e.g. Windows, MacOS, Linux)
  - Prototyped with the SocLib + MutekH platform

SoClib/MutekH

- SoClib = virtual prototyping platform (LIP6)
  - Many microprocessors supported (MIPS, ARM, PowerPC, etc.)
  - Embedded Operating System = MutekH
  - Transaction Level Modeling or Cycle Accurate Bus Accurate

- Code is first cross-compiled for the selected microprocessor
- Then, execution of: SocLib, MutekH, application
  - Performance metrics (traces)
  - Easy debugging (gdb: step by step execution, etc.)
Prototyping with SoCLib
Introduction

AVATAR: From Requirements to Prototyping

Conclusions, References

Conclusions

References
Conclusions

**TTool = Open-source solution for MDE**
- Academic and industrial involvement on TTool
- Many success stories (e.g., Freescale, EVITA)
- Used for teaching activities

**AVATAR**
- Integrated simulation
- Formal proof at the push of a button
  - Safety proof (UPPAAL)
  - Security proof (ProVerif)
- Easy-to-use virtual prototyping
Website, Publications

**TTool website:** http://labsoc.comelec.enst.fr/ttool/

- Under google: "TTool"
- How to install TTool, tutorials, these slides, etc.

**Papers**

