Institut
Mines-Telecom

Safety and Security Checking of Real-Time Systems Modeled in SysML

Ludovic Apvrille
ludovic.apvrille@telecom-paristech.fr

Tutorial - ModelsWard 2015

Learning Objectives

- To share an experience of real-time systems modeling
- To present a language, a tool, and a method that can be applied to the development of a broad variety of systems
- Focus on both safety and security models and proofs
- To practice modeling using a UML/SysML framework (TTool)
- To answer your questions

Content

1. Avatar

- Methodology
- Main concepts

2. Demonstration

- Microwave oven models
- Safety and security-oriented proofs
- Code generation

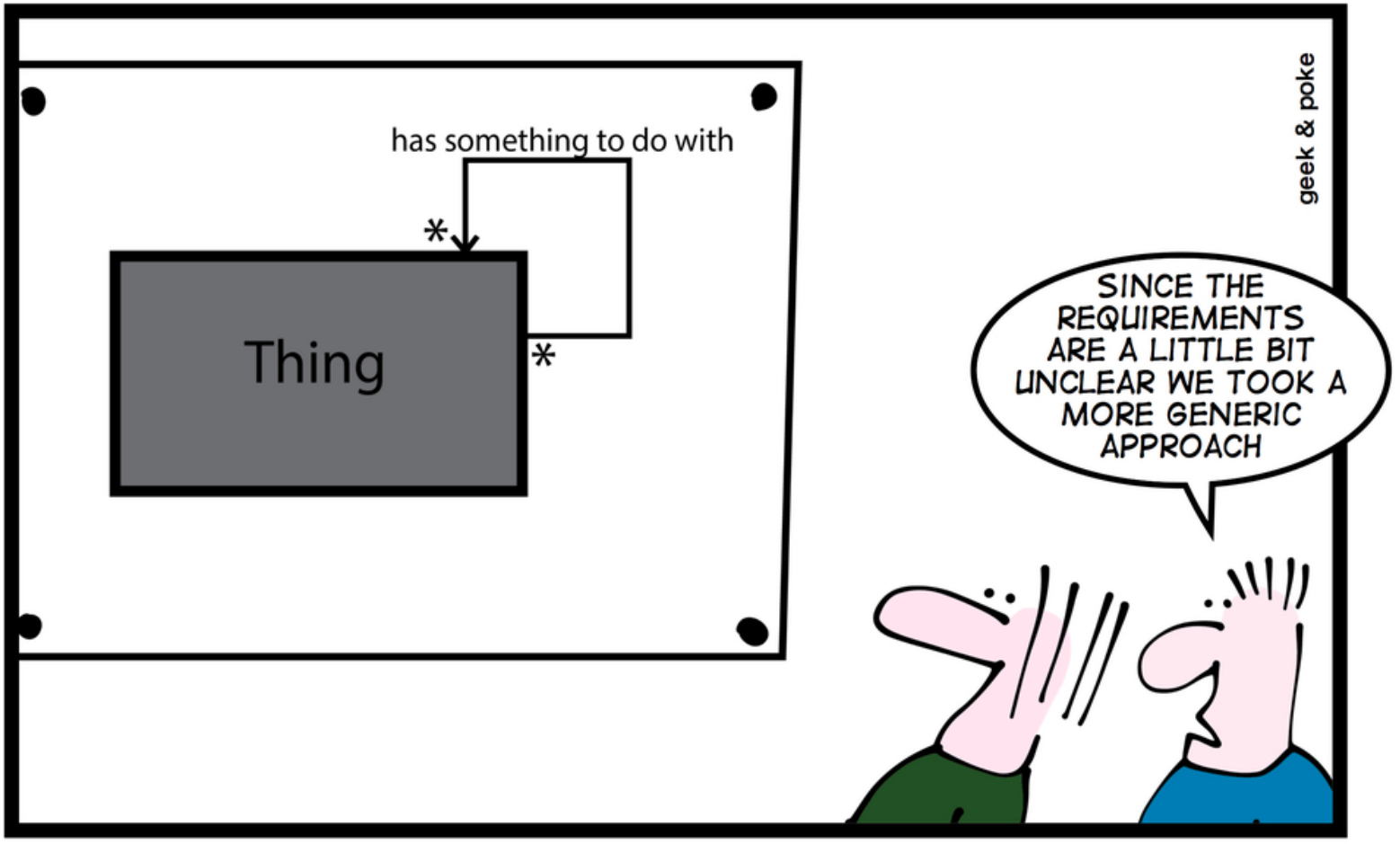
3. Practise

Your turn to work ;-)

Modeling is not Really a New Technique...

...and it is not limited to Software!

Abstraction Level



(*source: peek and Poke, July, 2013*)

What is UML?

UML = Unified Modeling Language

Main characteristics of UML

- Standard graphical modeling language for complex systems
  - Defined by OMG
- Specification, design, automatic code generation, documentation
- Independent of any programming language
- Object-oriented design
- Supported by many CASE Tools
  - CASE = Computer-Aided Software Engineering
- **But**: no standard UML methodology

From UML to SysML

What's wrong with UML? (as far as system modeling is concerned)

- Objects are for computer-literates, not for systems engineers
- Requirements are described outside the model using, e.g., IBM DOORS
- Allocation relations are not explicitly supported

Nevertheless SysML is a UML 2 profile

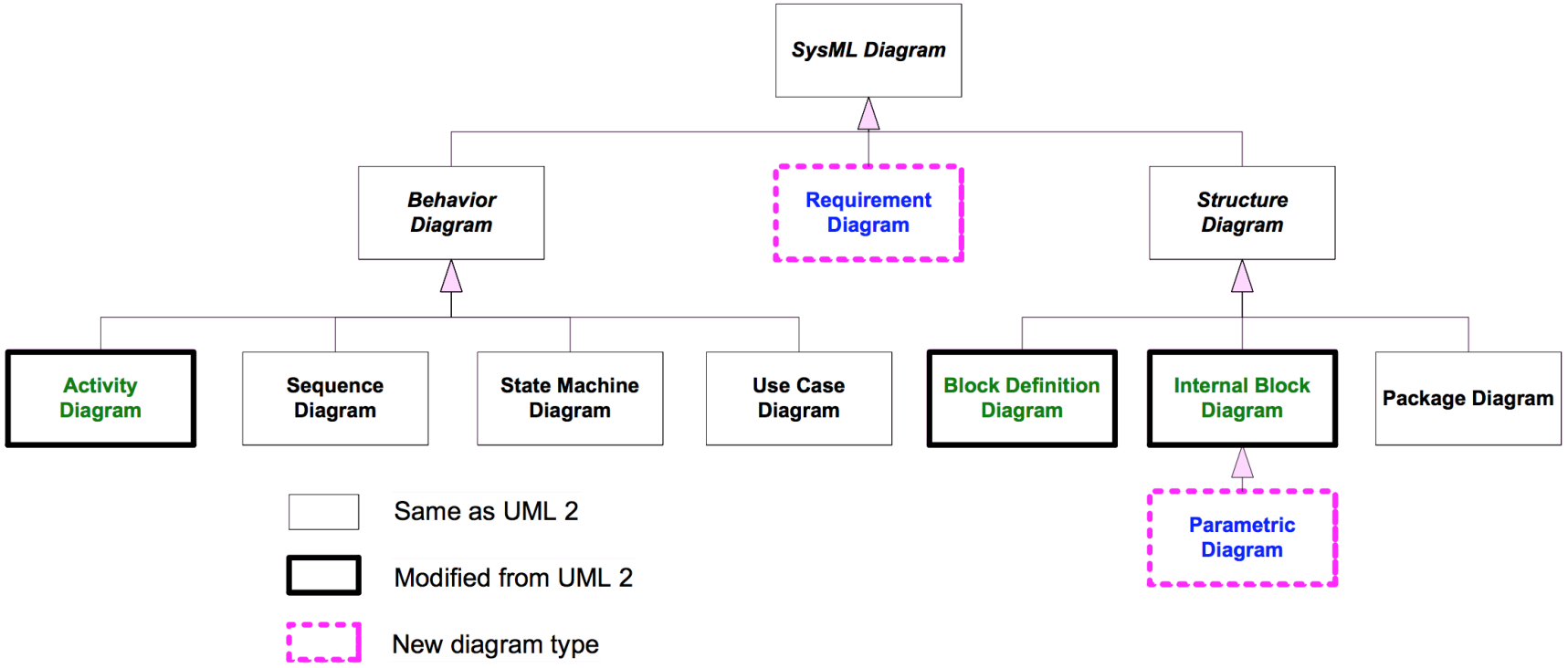
- Developed by the Object Management Group (OMG) and the International Council on Systems Engineering (INCOSE)

SysML standard:
www.omgsysml.org

SysML

- **An international standard** at OMG
  - UML profile
- **A graphical modelling language** that supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, staff, procedures, and facilities
- **A notation**, not a method
- **Proprietary tools**
  - Enterprise Architect, Rhapsody, Modelio, ...
- **Free software tools**
  - TOPCASED, Papyrus, **TTool**, ...
- **User communities**
  - http://sysmlfrance.blogspot.com/
  - http://sysmlbrasil.blogspot.fr/p/sysml-brasil.html

SysML Diagrams vs. UML Diagrams



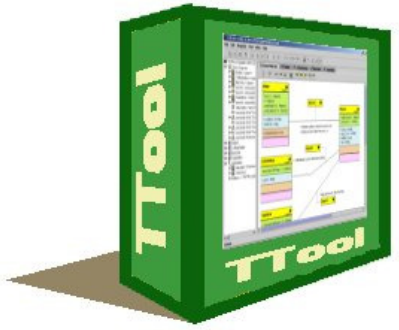
From SysML to AVATAR

- **AVATAR reuses most SysML diagrams**
  - Requirement capture: requirement diagrams
  - Analysis: use case, sequence and activity diagrams
  - Design: block instances and state machines diagrams
- **AVATAR does not entirely comply with the OMG-based SysML**
  - In AVATAR, block instances diagrams merge block and internal block diagrams
  - AVATAR tunes SysML parametric diagrams to express properties (TEPE)
  - AVATAR does not support continuous flows
- **AVATAR gives a formal semantics to several diagrams, including**:
  - Block instance and state machine diagrams
    - Starting point for simulation, verification and code generation

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)



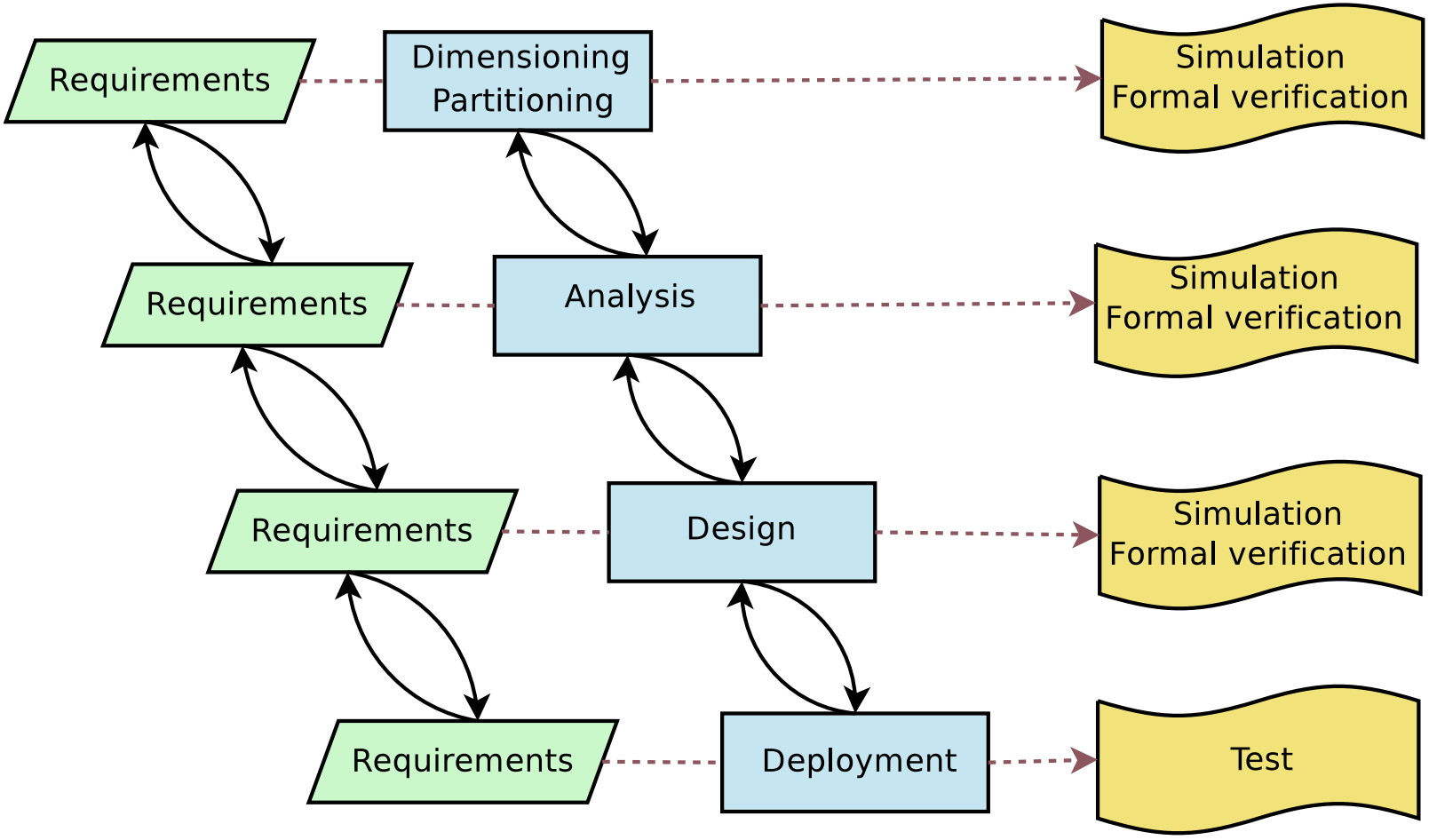
Main ideas

Lightweight, easy-to-use toolkit

Simulation with model animation

Formal proof at the push of a button

Overview of the Extended V-Cycle



Simulation vs. Formal Verification

Simulation explores execution paths in the model relying on

- The experience of the Human who guides the simulation
- Random selection in case of non deterministic choice (several transitions fireable at the same time)

Formal verification formally checks a model of the system against (a subset of) its expected properties

- **Safety analysis** with UPPAAL
  - Search through the state space of the system
- **Security analysis** with ProVerif
  - Confidentiality, authenticity
- **Structural analysis** without state space exploration
  - Invariants

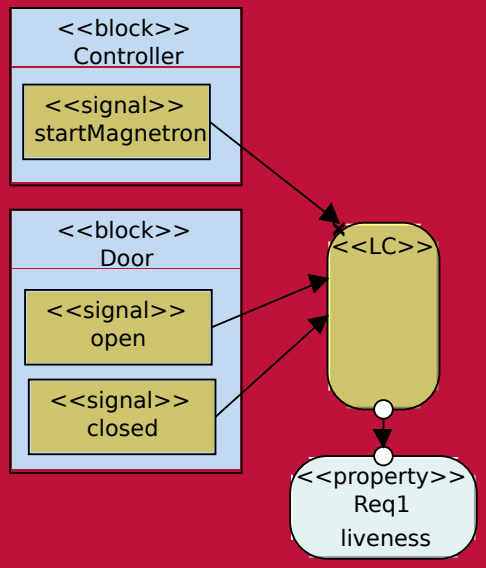
Formal verification relies on mathematics rather than chance

Property Modeling

Safety properties

Customized Parametric Diagrams (TEPE)

Reachability, liveness



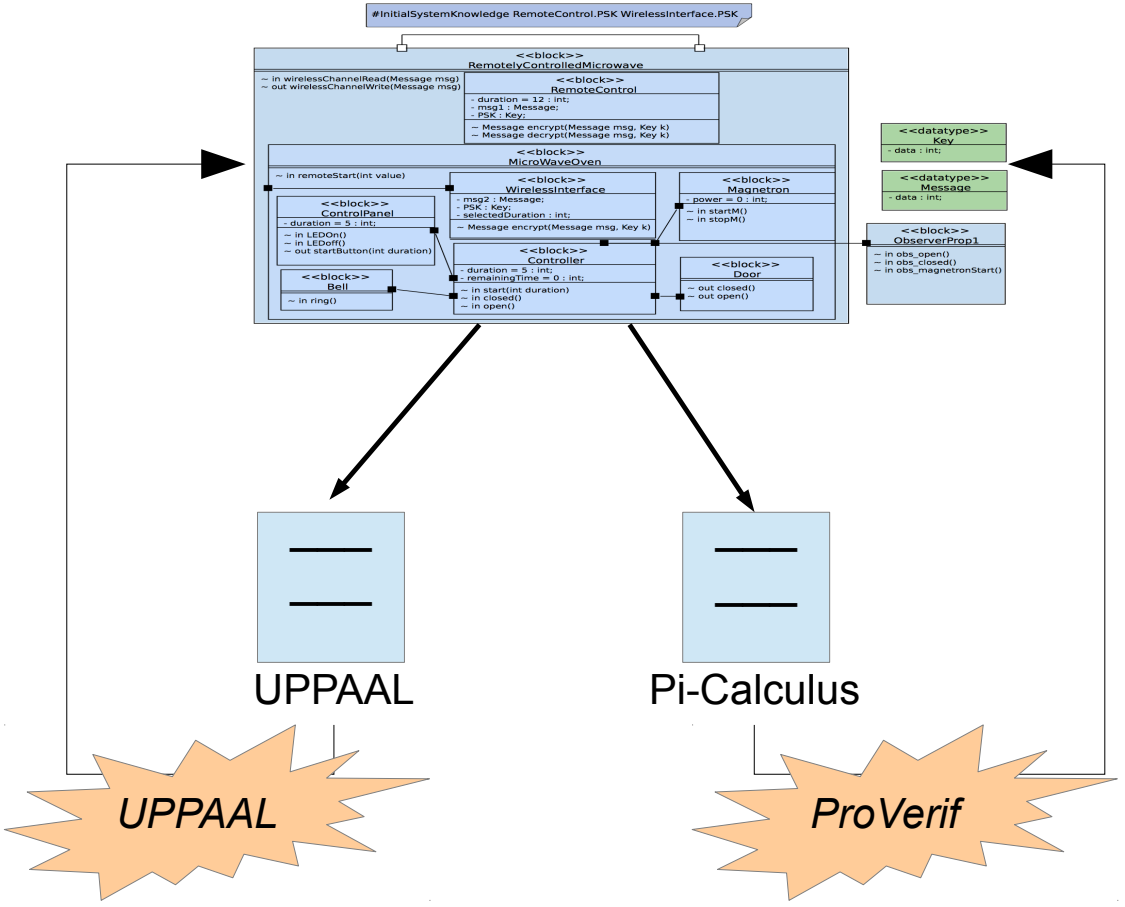
Security properties

Based on basic pragmas

- Confidentiality of a block attribute
- Authenticity of interconnected block signals

```
#Confidentiality RemoteControl.duration
#Authenticity RemoteControl.SendingRemoteOrder.msg1
              WirelessInterface.gotWirelessOrder.msg2
```

Model Transformation for Formal Verification

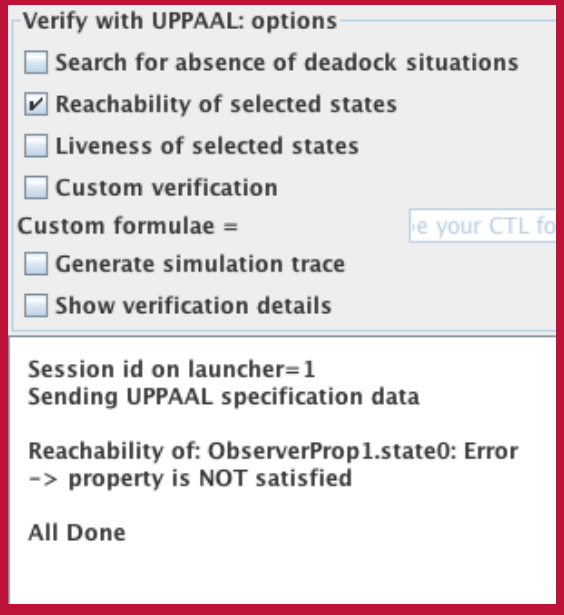


Formal Verification

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

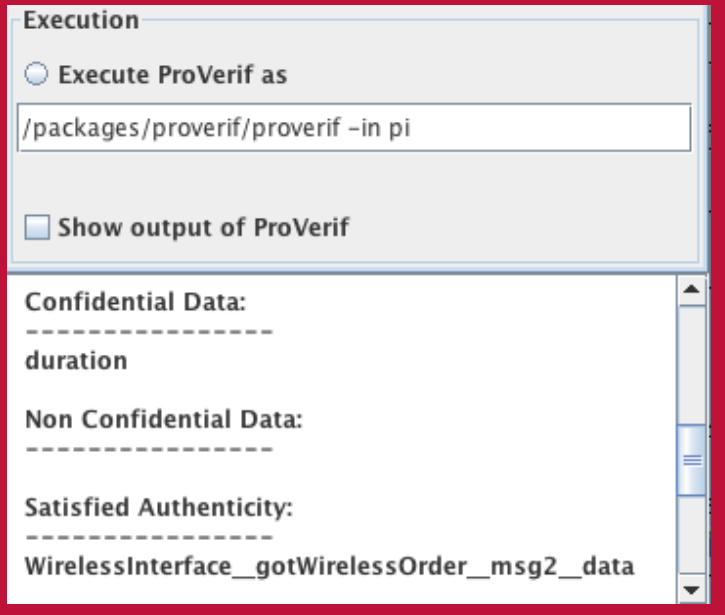
Safety properties

UPPAAL based

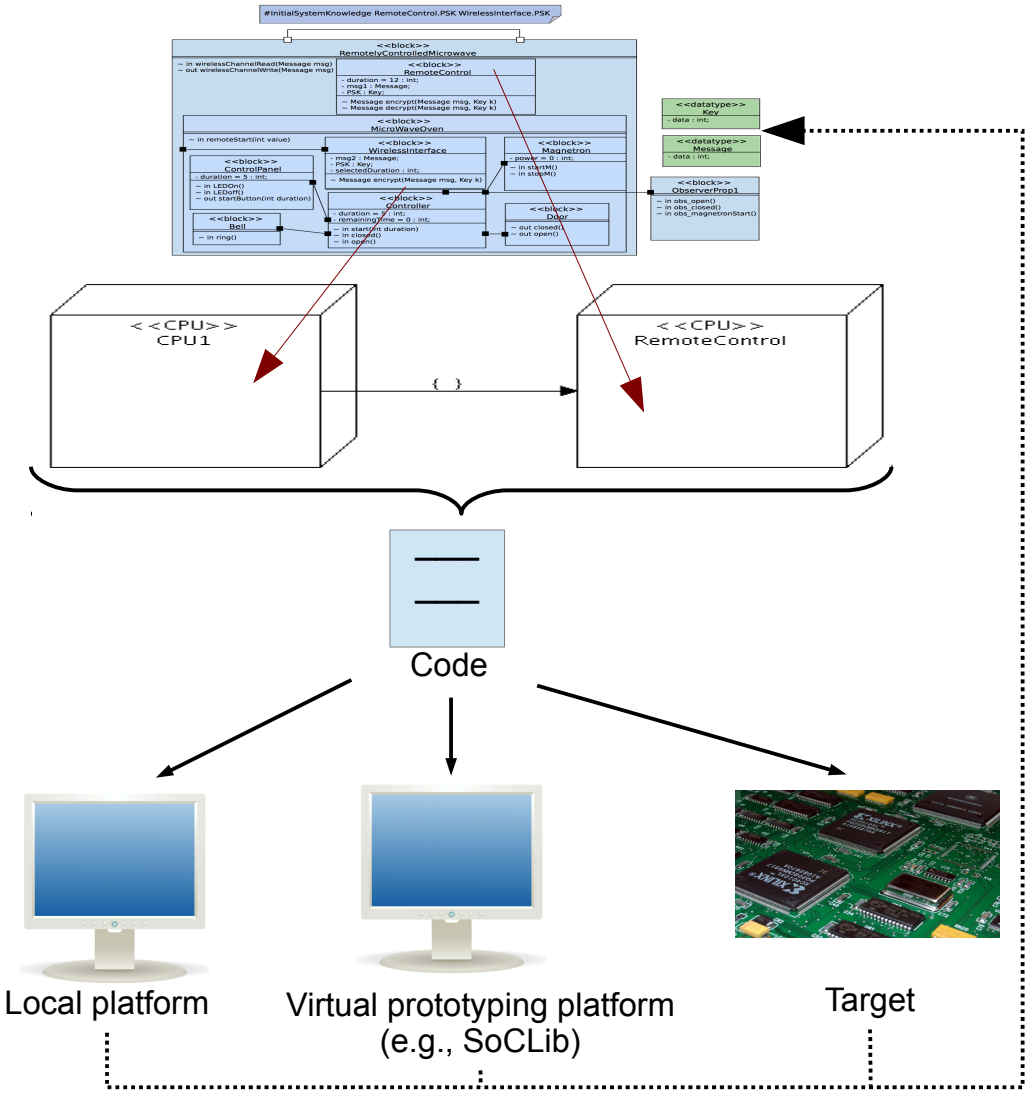


Security properties

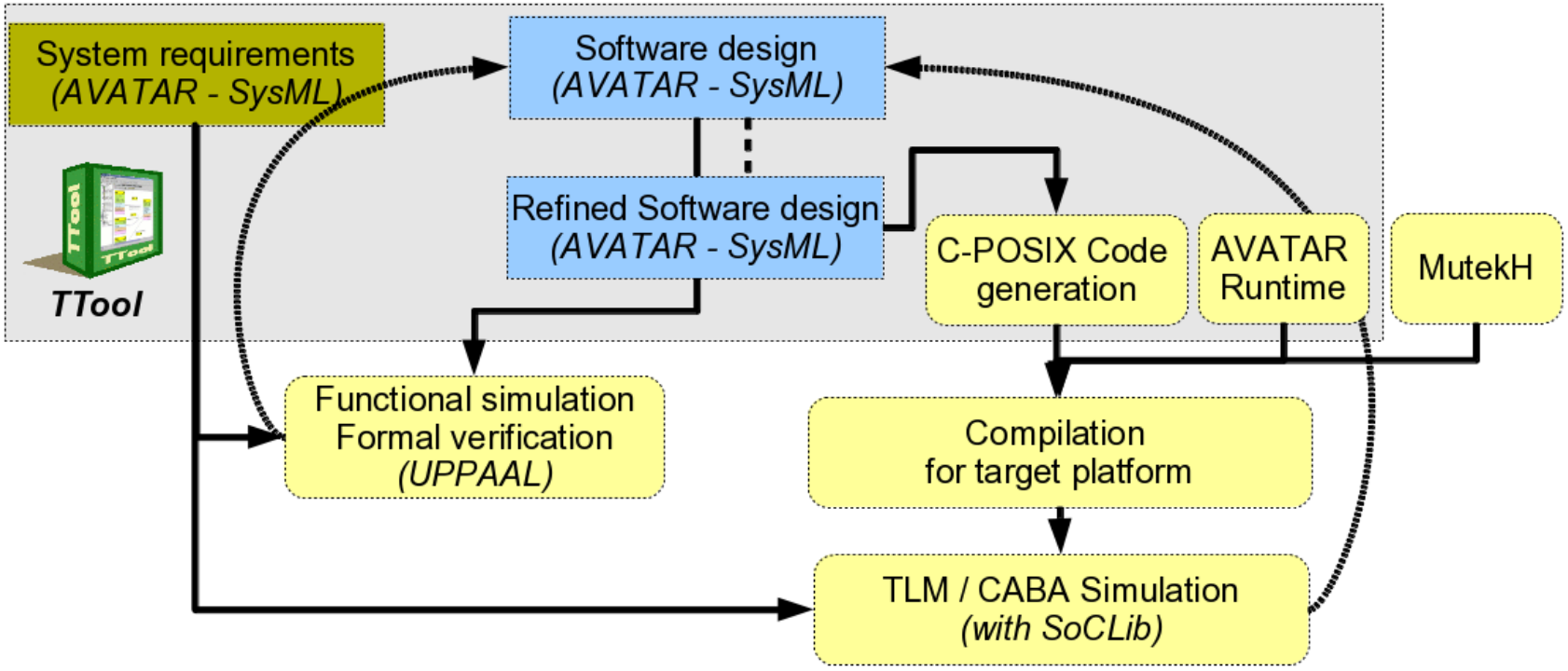
ProVerif based



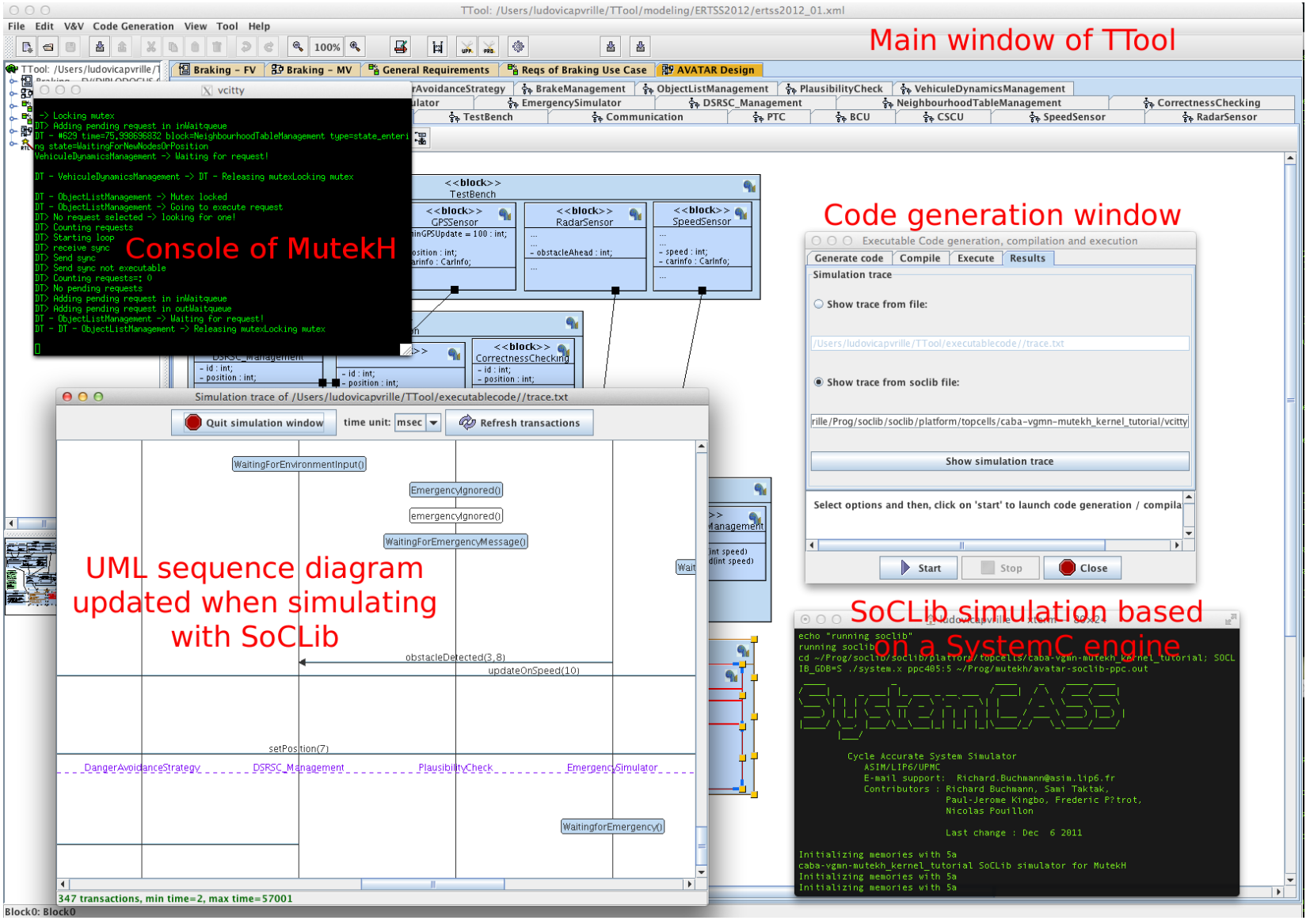
Code Generation: Overview



Virtual Prototying: Method



Virtual Prototyping: Graphical Environment



Use of Customized Generated Code

Console debug

- Using e.g. *printf()* function

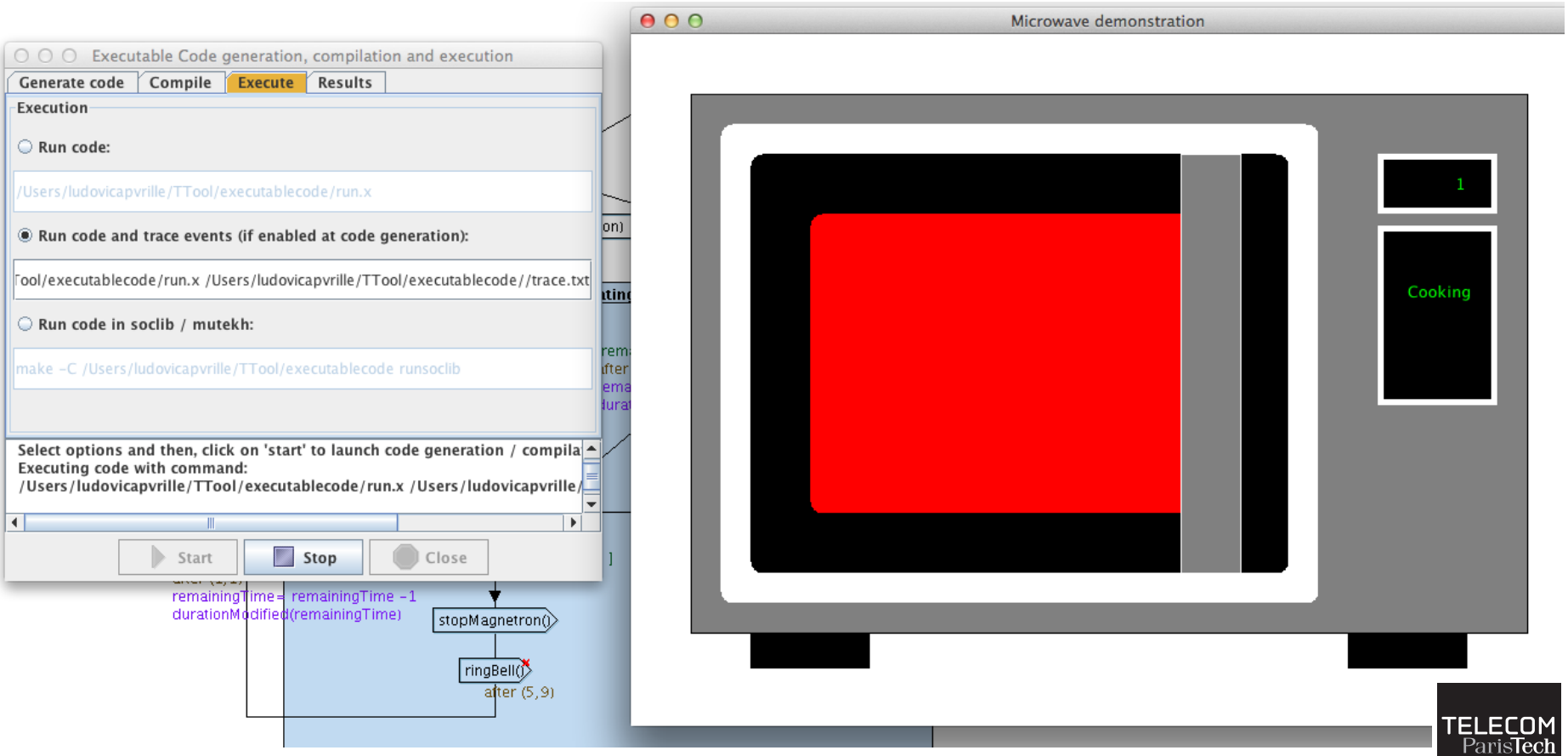
Connection to a graphical interface

- Piloting the code with a graphical interface
- Visualizing what's happening in the executed code
- Connection to graphical interface via, e.g., *sockets*

Use of Customized Generated Code (Cont)

Graphical interface for the microwave oven

- Socket connection to a graphical interface programmed in Java



Demonstration

System Modeling

- Very quick overview of requirement and analysis
- Design

Property Modeling

- Safety, Security

Code generation

- Execution on localhost, prototyping, connection to a graphical interface

Your Turn: Incremental Modeling of a Landing Gear

Version 1

- Basic landing gear: can go up and down. The procedure takes 15 seconds and cannot be aborted.

Version 2

- Procedure can be aborted by starting the opposite function at whatever moment

Version 3

- Warning if altitude is close to the ground, and the gear is in
- Add confidentiality and authenticity mechanisms/properties to the input and output information of the landing gear