TEPE: A SysML language for Time-Constrained Property Modeling and Formal Verification

Daniel Knorreck, Ludovic Apvrille, Pierre de Saqui-Sannes

Telecom ParisTech, ISAE
{daniel.knorreck, ludovic.apvrille}@telecom-paristech.fr, p.de-saqui-sannes@isae.fr

November, 16, 2010

Outline

Outline

Need for a homogeneous Design/Verification Environment

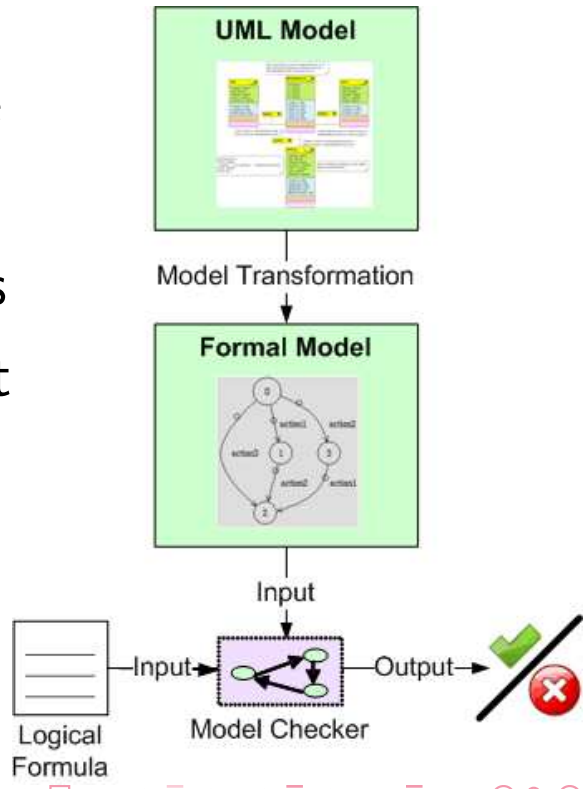
Analysis and Design Stages are quite well supported

- UML/SysML widely used for system analysis and design
- Profiles often provide a formal semantics
- Simulation and formal verification applicable

Despite comprehensive and intuitive models...

- Properties often expressed in temporal logics
- Transparency of formal techniques not maint
- Formal verification remains cumbersome

Objective: Describe a system and its properties in the same language



Excerpt of Related Work on Constraint Languages

- UML 2.0 poorly equipped with means to express behavioral/temporal constraints
- CCSL in MARTE [Mallet et al.]: very useful approach to formalize/abstract (logical) time relations, inspired our work
- Live Sequence Charts [Harel et al.]: add formalism to UML Sequence Diagrams like discriminating mandatory from provisional behavior
- Timeline Editor [Holzmann et al.]: Protocol centric graphical verification language
- Hardware Specification Languages (PSL, e, System-Verilog, etc.): low level, heavily tied to clock cycles/sampling events

Outline

Strengths of TEPE

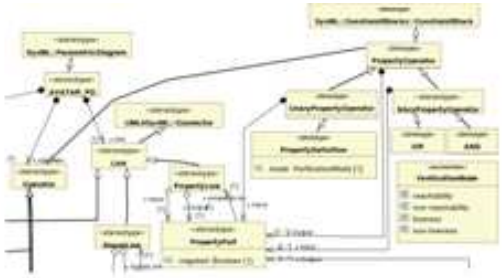
Ease of use

- Seamlessly integrates verification into a SysML environment
- High level description, not explicitly tied to physical clocks
- Far more intuitive than other formalisms (e.g. CTL)

Expressiveness

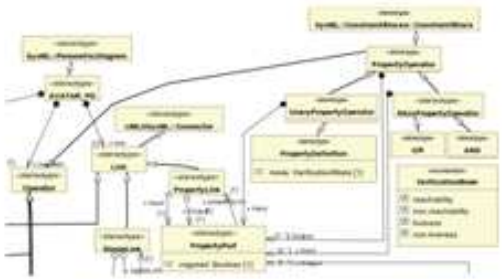
- Relates block attributes and signals
- Naturally captures sets of execution traces
- Supports logical and physical time
- Formally defined (in terms of observer automata, timers and CTL formulas)

Building Blocks I



SysML Metamodel

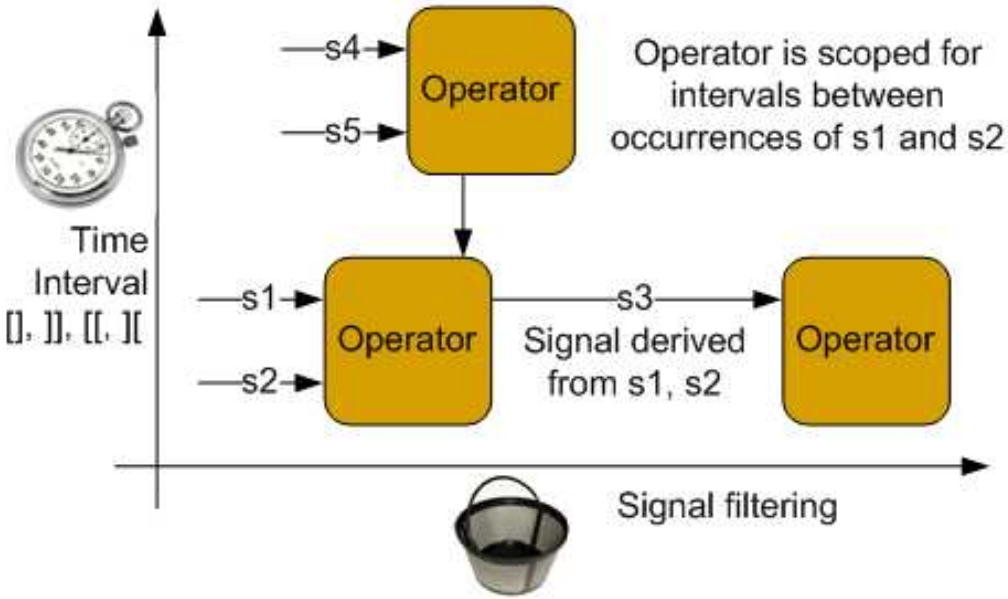
Building Blocks II



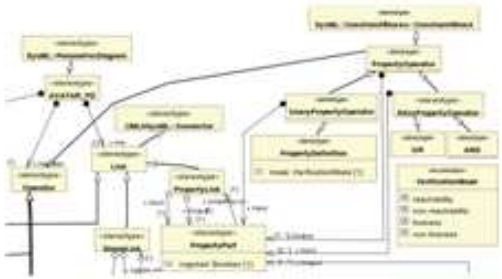
SysML Metamodel

+

Verification Concept



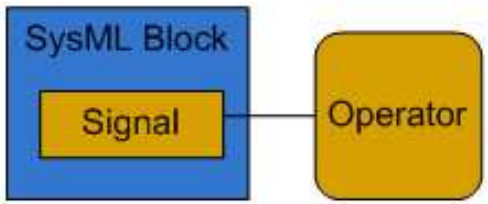
Building Blocks III



SysML Metamodel

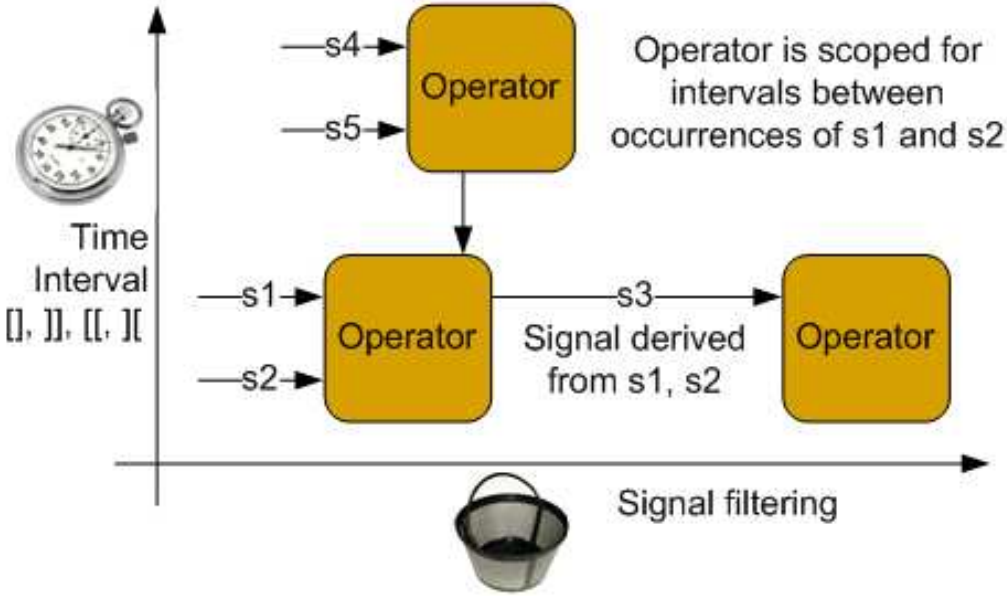
+

Direct reference to the model



+

Verification Concept



= TEPE

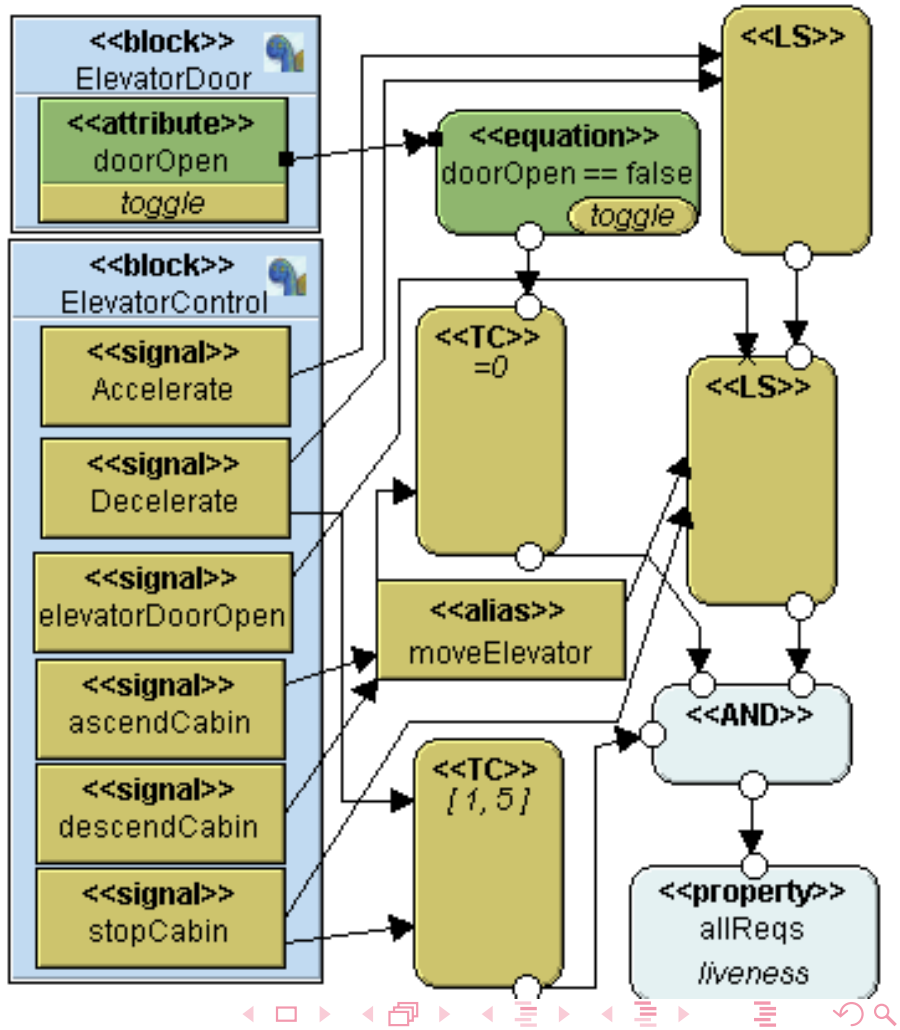
Building a TEPE Diagram

1. Blocks are represented with their *Attributes* and *Signals*
2. Derived entities (*Equations*, *Signals*) are defined
3. Reasoning about sequential and temporal behavior in terms of cascaded operators
4. Properties are related to each other with logic operators (AND, OR, etc)
5. Properties are labeled and their verification scope (Liveness, Reachability) is determined

Elevator Example

Properties

1. Door does not open when elevator is moving
2. Elevator does not depart with an open door
3. Operational profile: start, accelerate, decelerate, stop
4. Deceleration to be accomplished between 1 and 5 time units before the selected floor is reached

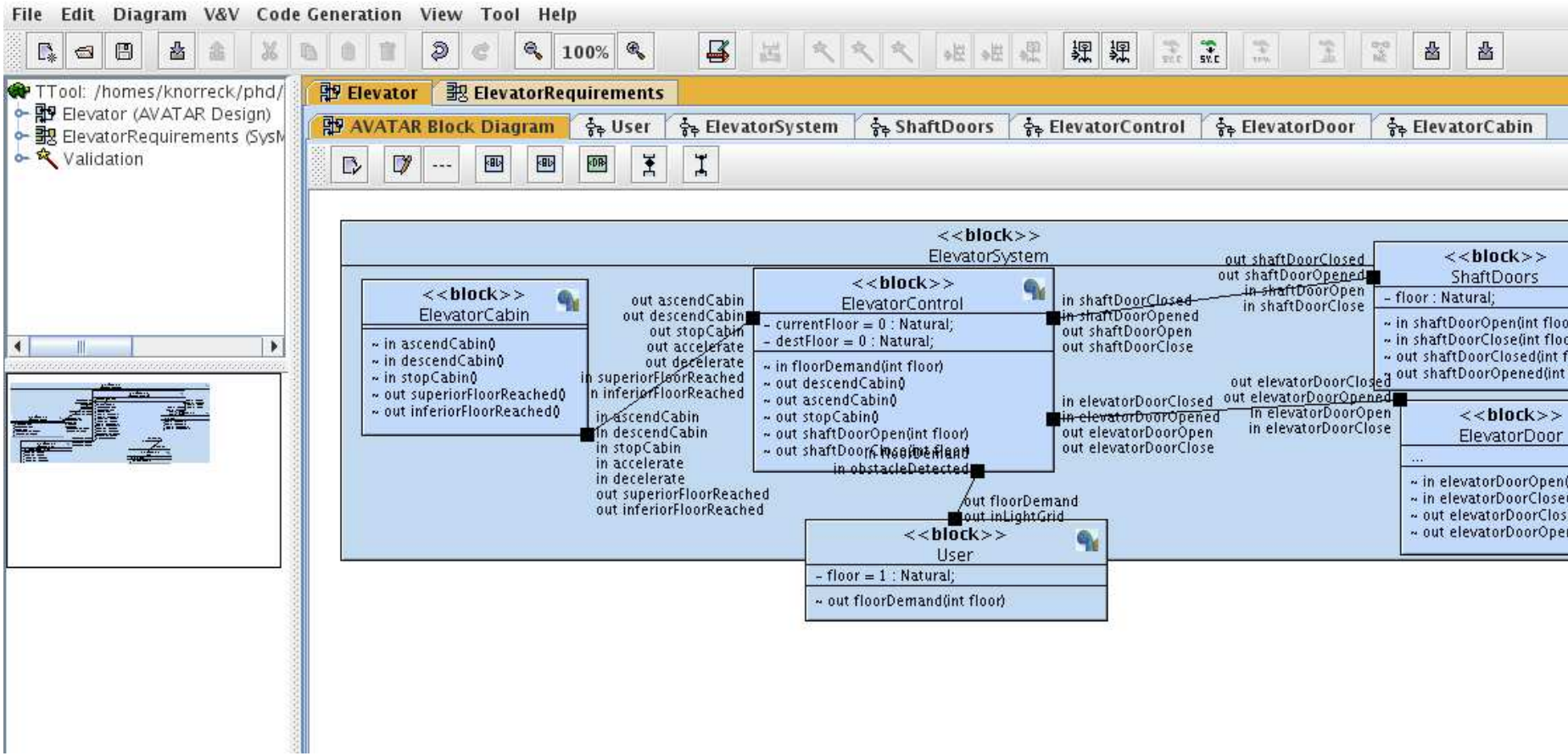


Outline

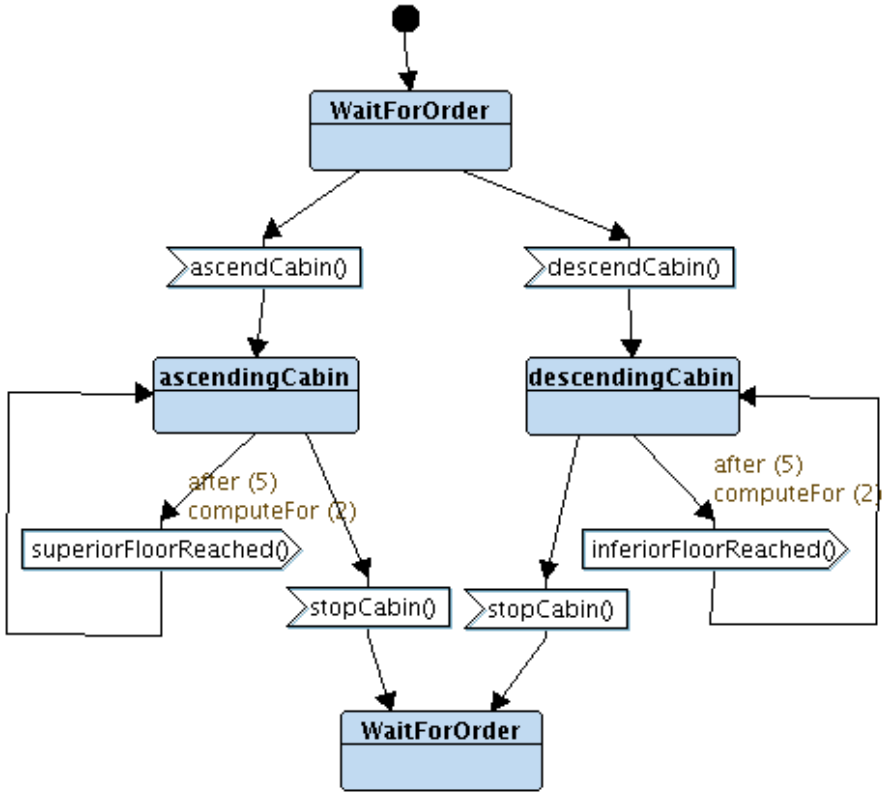
The AVATAR SysML Profile

- Reuses 8 SysML diagrams (Package diagrams not supported)
- Comes with a methodology
- Supported Design stages:
  1. Requirement Capture (Requirement D.)
  2. System Analysis (Use Case D., Interaction Overview D., Sequence D.)
  3. System Design (Block D., State Machine D.)
  4. Property Modeling (Parametric D. $\rightarrow$ TEPE)
  5. Formal verification (UPPAAL)
- To refine models the stages are iterated

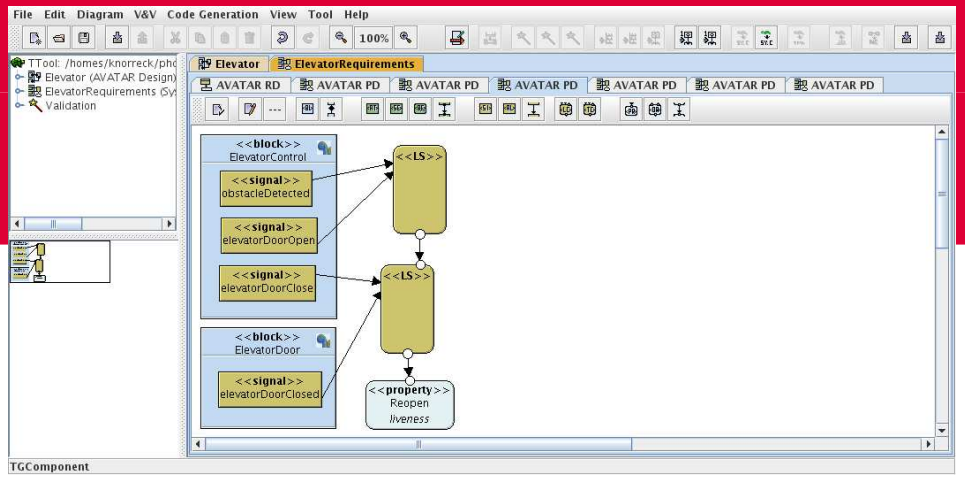
AVATAR Profile Example: Block Diagram



AVATAR Profile Example: State Machine



TTool: Main Features



- Open-source UML toolkit
- Meant to support UML2/SysML profiles
  - 8 UML2/SysML profiles are currently supported
- Mostly implemented in Java
  - Editor, interfaces with external tools
  - Simulators implemented in C++ or SystemC
- Formal verification and simulation features
  - Hides formal verification and simulation complexity to modelers
  - Relies on external tools
  - Press-button approach

Outline

Conclusions on TEPE

- Captures behavioral and static properties in a natural fashion
- Supports a notion of logical and physical time
- Integrated into the AVATAR SysML profile for Embedded Systems
- Extends transparency of formal techniques to the verification stage
- Directly refers to system elements introduced at previous stages
- Graphical representation recommended but not mandatory

Questions

Thank you for your attention!