Institut
Mines-Telecom

Requirements Analysis in SysML Models

Ludovic Apvrille,
ludovic.apvrille@telecom-paristech.fr

NEPTUNE 2013, Paris, France

Outline

Outline

Rationale

Goal

Validate a SysML design against requirements

Issues

- Unformal SysML diagrams
- Designers have limited knowledge of formal languages and tools

Contributions

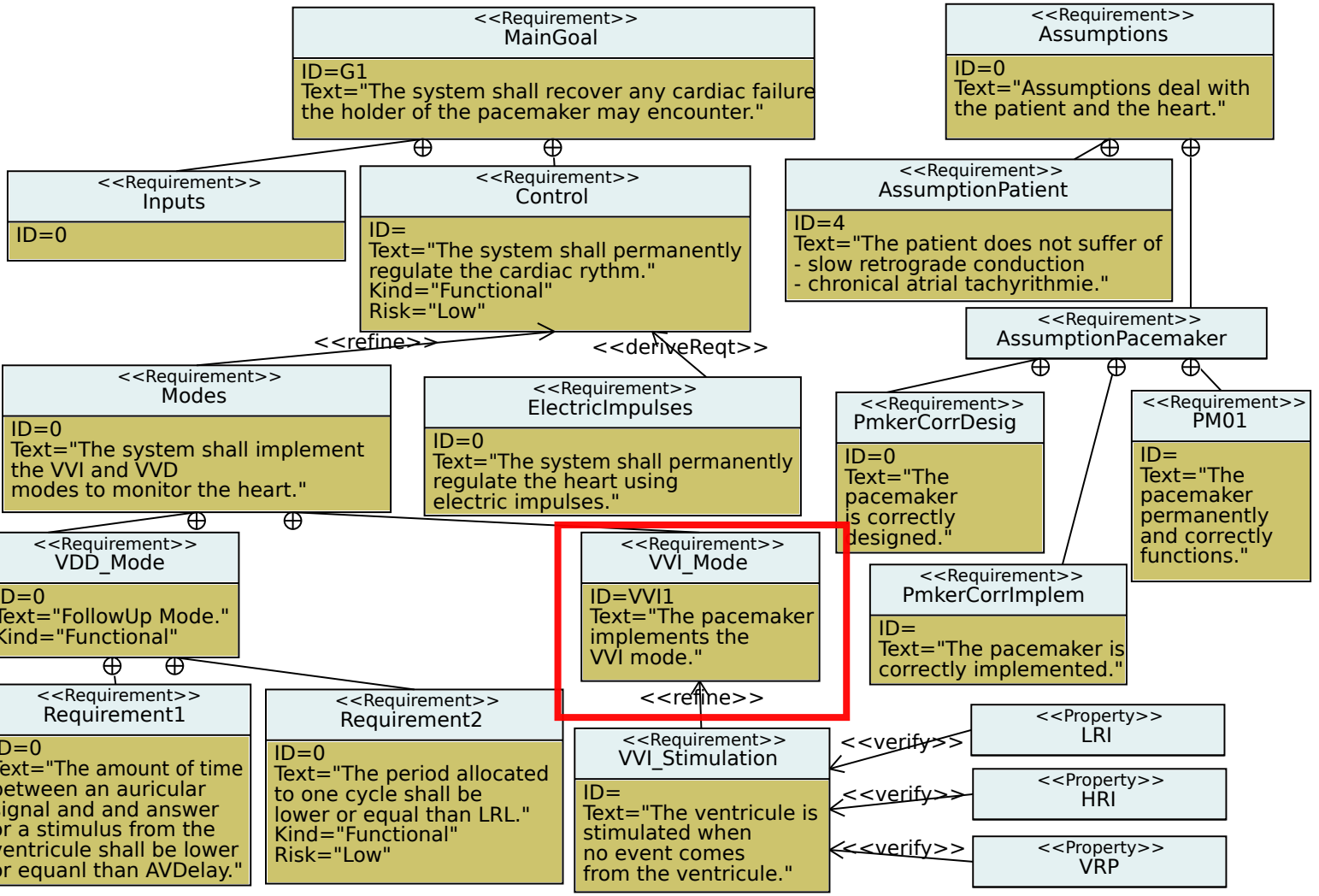
- Formal SysML environment (AVATAR)
- Easy-to-use toolkit (TTool) interfaced with UPPAAL
- Method

AVATAR Method

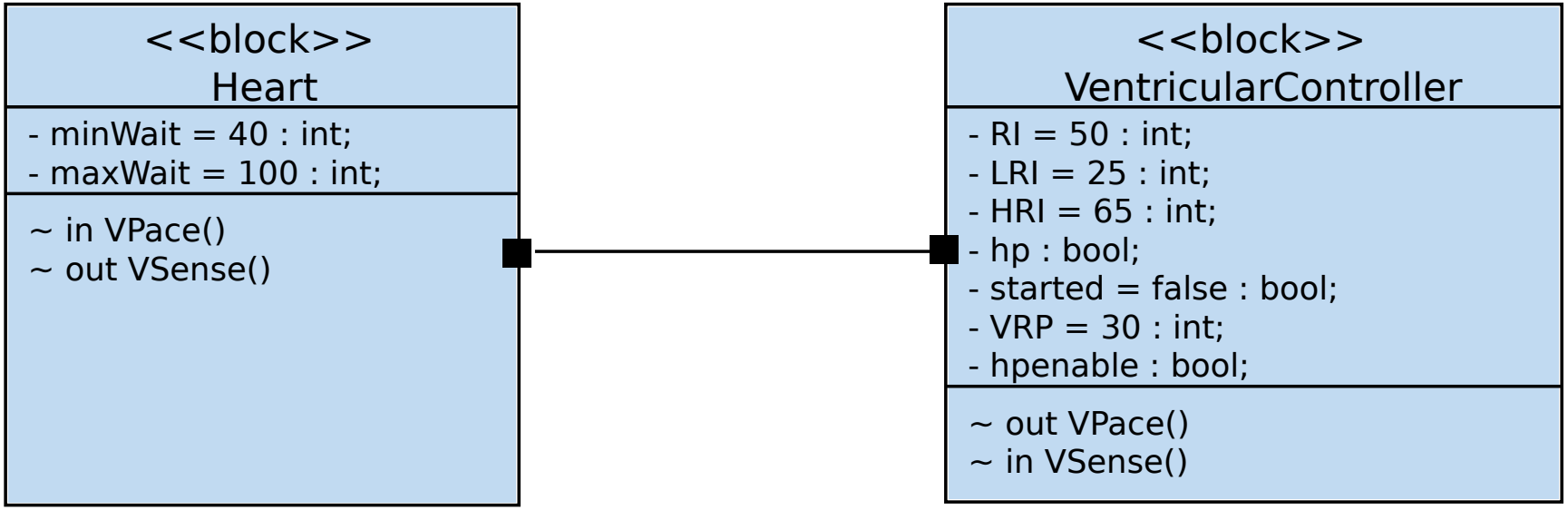
1. Gather the requirements
2. Express symplifying hypotheses
3. Analyze the specifications with use-cases, and document the uses cases with sequence diagrams
4. Design with SysML blocks and state machine diagrams
5. Express the properties to be verified
6. Use of simulation and formal verification techniques in order to check design diagrams against properties
7. Build the traceability matrix

Outline

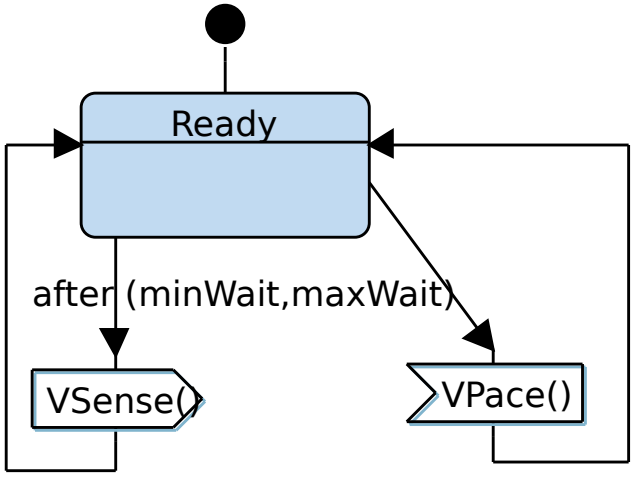
Goals, Assumptions, Refined requirements and Properties



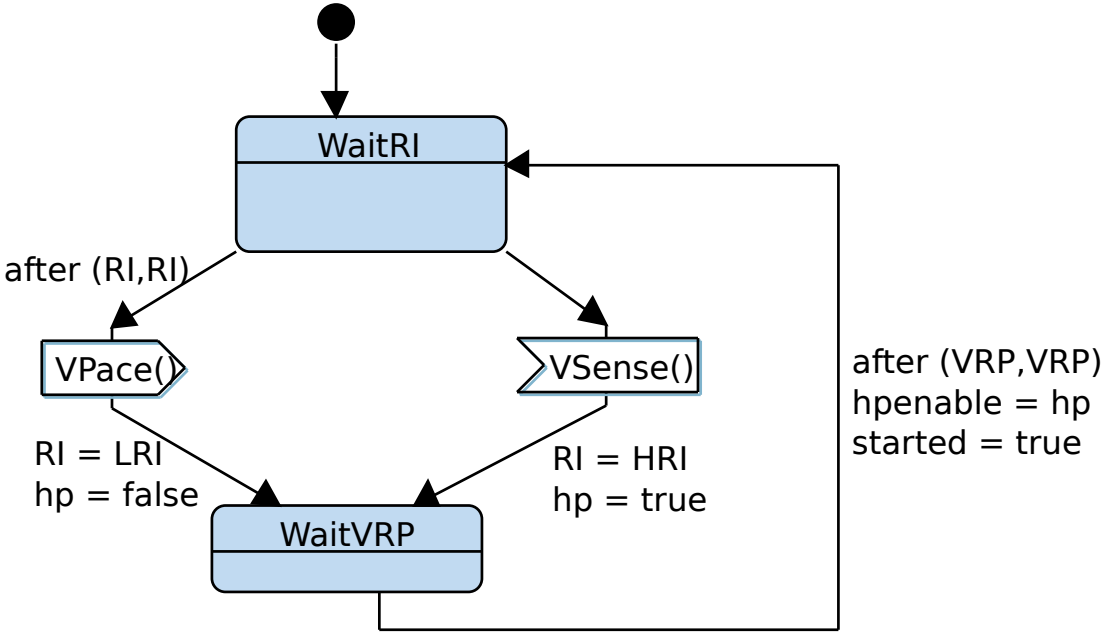
Architectural Design: VVI Mode of the Pacemaker



Behavioral Design: VVI Mode of the Pacemaker

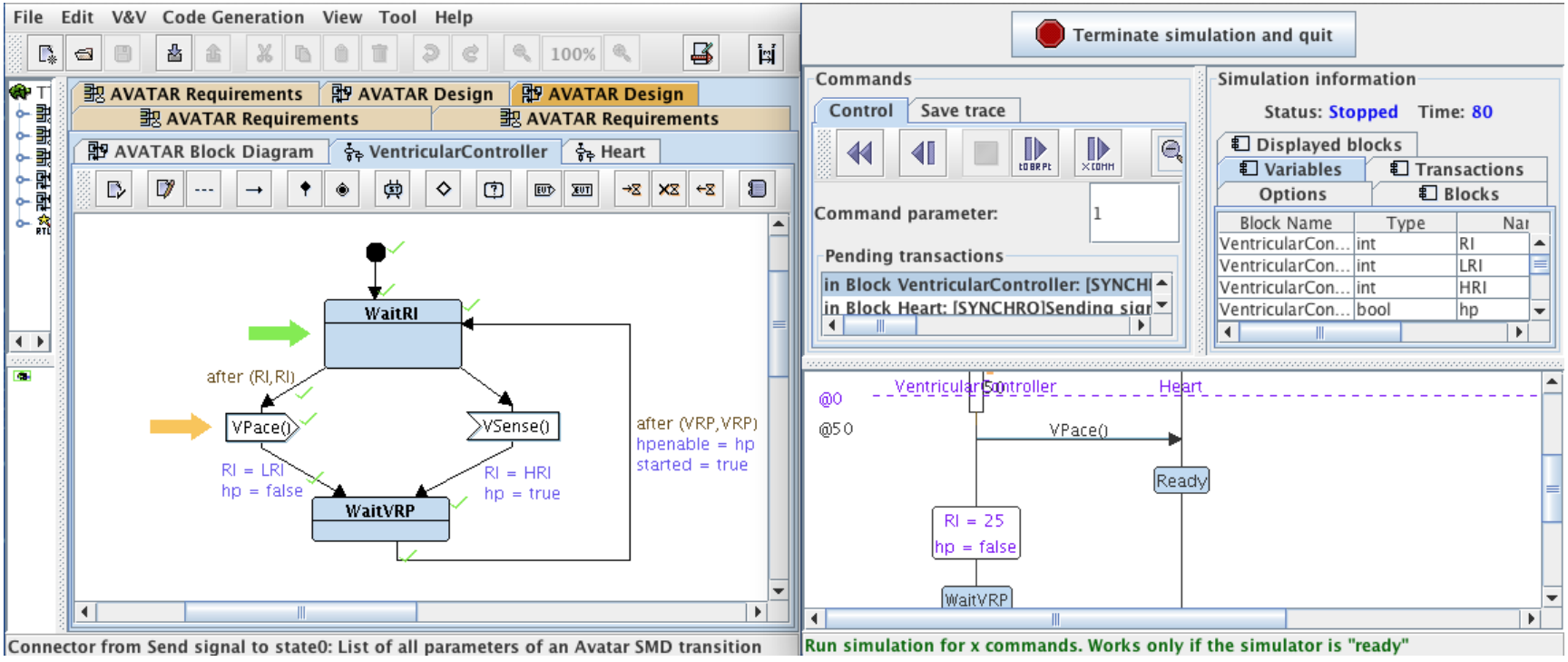


State machine of *Heart*



State machine of *VentricullarController*

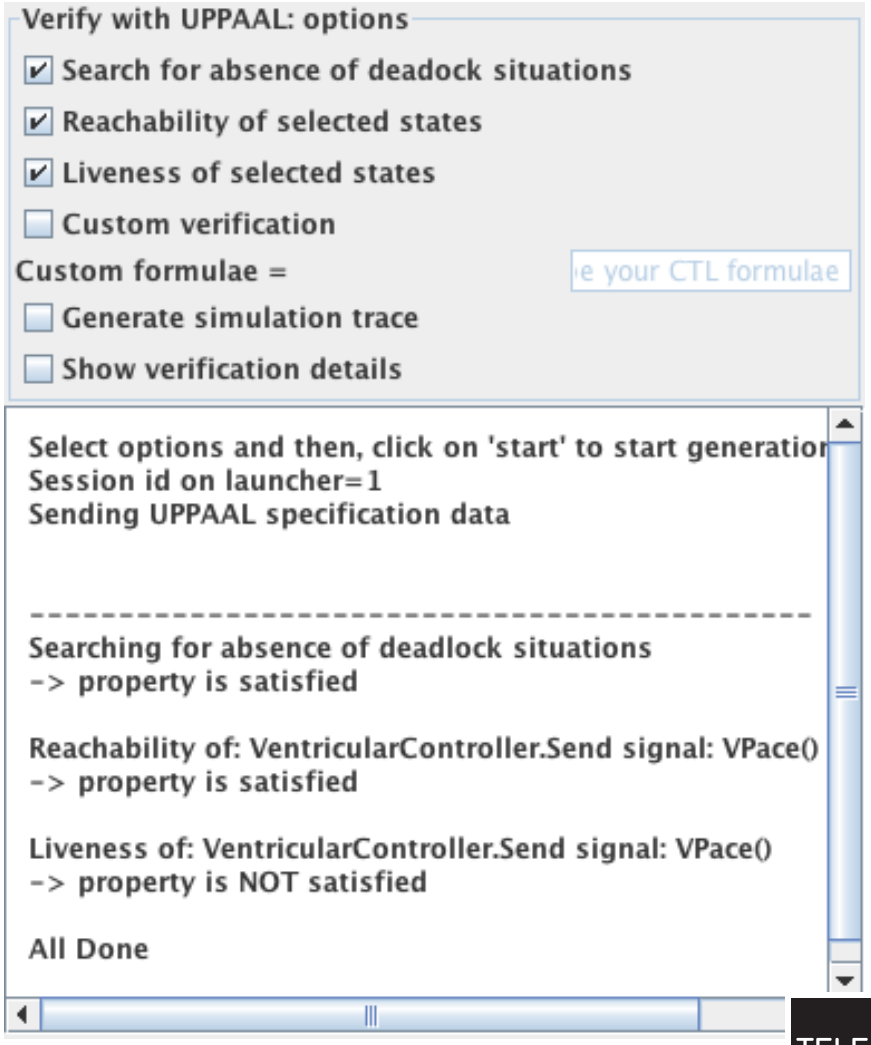
Design Simulation



- Model debugging
- Interactive simulation integrated in TTool
- Back-tracing to model
  - e.g., taken paths, current states, next transitions

Deadlock Free, Reachability and Liveness

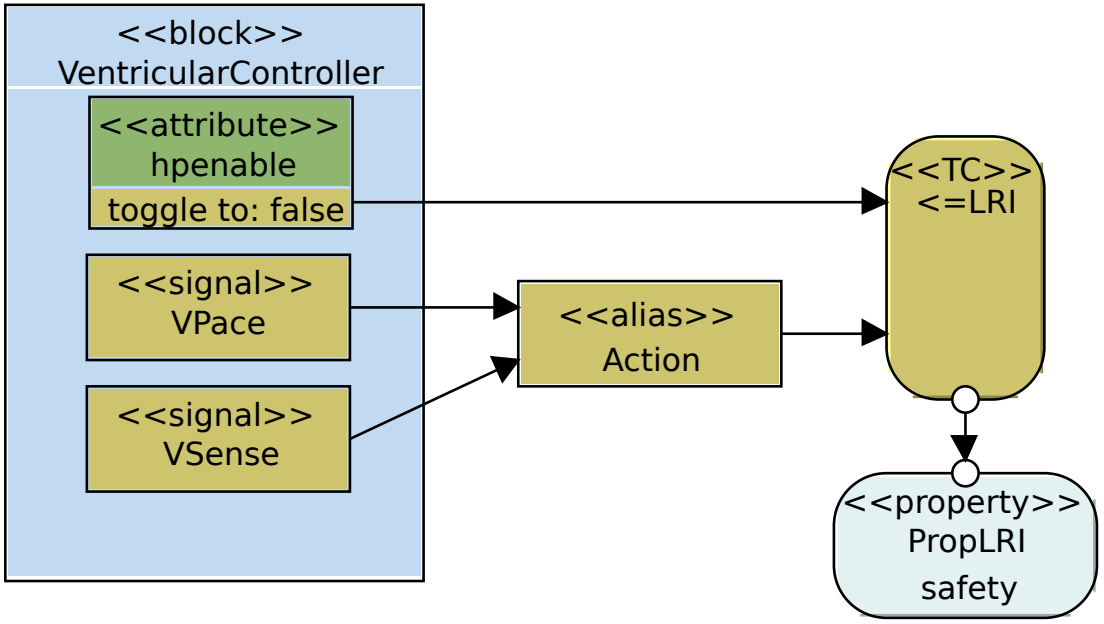
- Press-button approach to search for the reachability/liveness of actions of state machines
  - Proofs via UPPAAL
- No deadlock
- Reachability and liveness of the sending of *VPace*



TEPE: Expressing Complex Properties

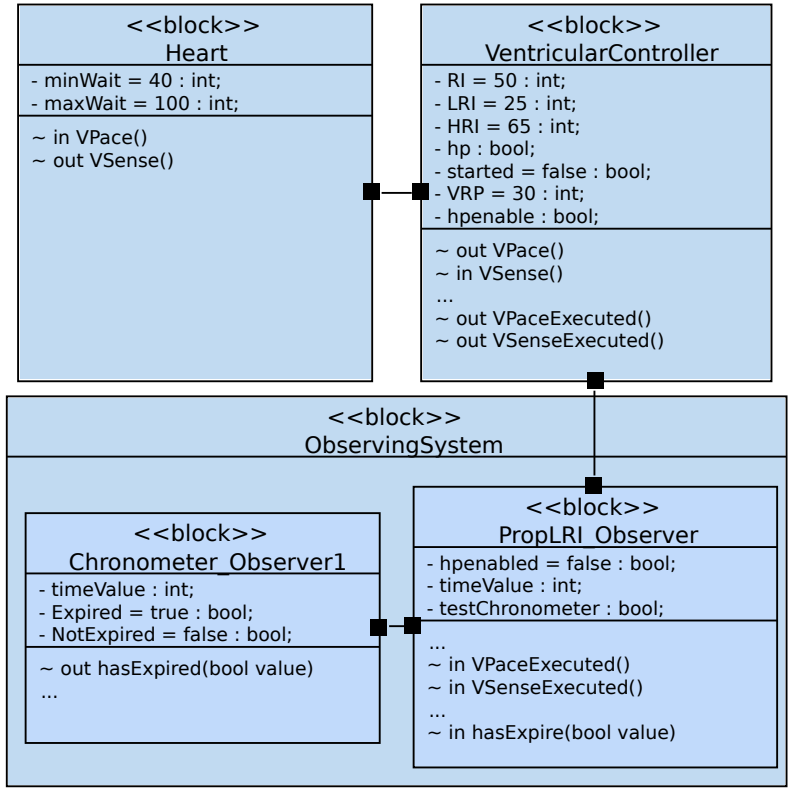
- Use of SysML parametric diagrams
- Logical and temporal relations between events and attributes

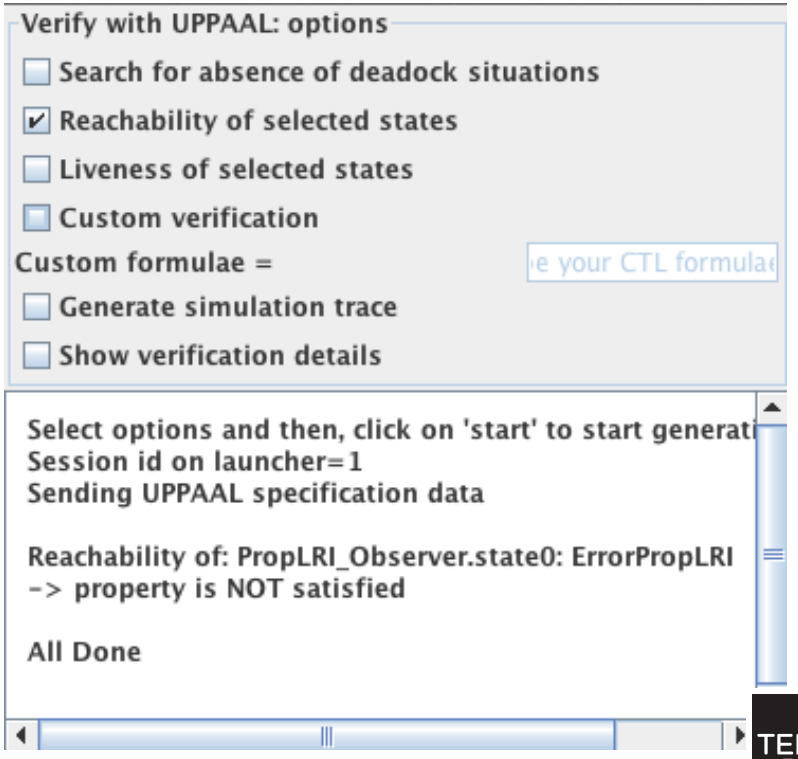
- **LRI**: when the pace on the hysteresis is deactivated, a pace on the ventricle or a sense action of the heart must be carried out before LRI time units



Observer-Guided Verification

- Expression of properties within the design
- Observers should have an error state whose reachability can be searched for in TTool/UPPAAL





Outline

Conclusion

- Press-button approach for simulation and formal proof
- Simulation and verification results are back-traced to design models
- Easy-to-learn and easy-to-use toolkit
- Full support for handling complex embedded systems, e.g.:
  - System partitioning (DIPLODOCUS)
  - Proof of safety and security properties
  - Prototyping (code generation from models, integration with SoCLib)
- Many academic and industrial systems modeled and verified
  - e.g., Critical automotive systems, mobile platforms, base stations

To Go Further ...

TTool

- http://ttool.telecom-paristech.fr
- Can be executed on usual operating systems (Windows, Linux, MacOS)
- Supports several profiles (e.g., DIPLODOCUS, AVATAR)
- Open-source, contributions are welcome
- Support from industrial and academic partners

AVATAR

- http://ttool.telecom-paristech.fr/avatar.html
- Tutorials, examples