



Institut
Mines-Telecom

Requirements Analysis in SysML Models

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

NEPTUNE 2013, Paris, France



Outline

Introduction

SysML-based requirement engineering process

Our approach: AVATAR

Requirements analysis of the pacemaker with AVATAR/TTool

Requirement capture, design

Simulation

Formal verification

Conclusion



Outline

Introduction

SysML-based requirement engineering process
Our approach: AVATAR

Requirements analysis of the pacemaker with AVATAR/TTool

Conclusion

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 simplifying 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

Introduction

Requirements analysis of the pacemaker with AVATAR/TTool

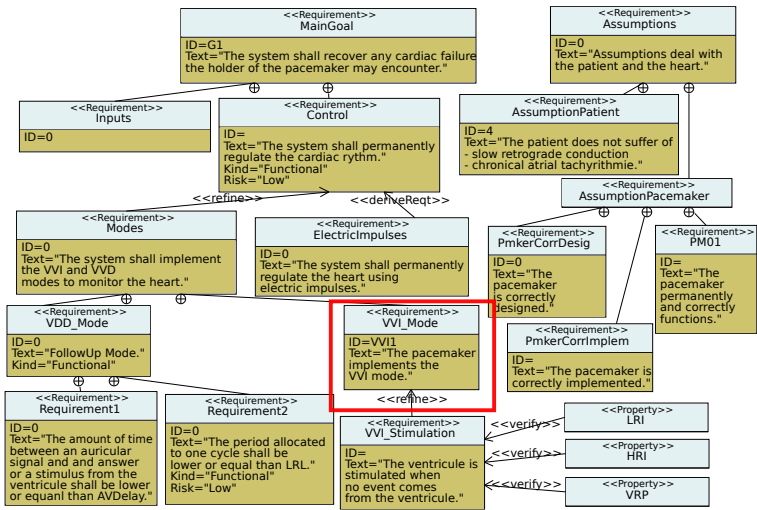
Requirement capture, design

Simulation

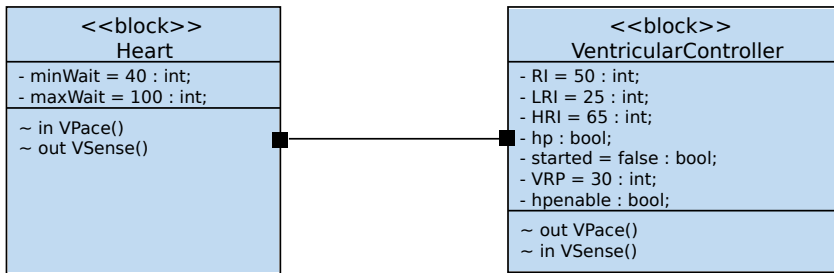
Formal verification

Conclusion

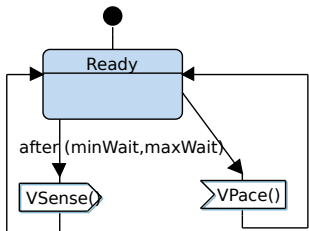
Goals, Assumptions, Refined requirements and Properties



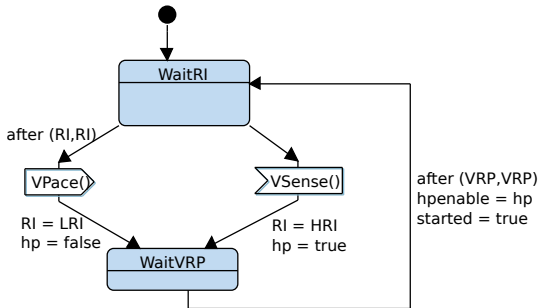
Architectural Design: VVI Mode of the Pacemaker



Behavioral Design: VVI Mode of the Pacemaker



State machine of *Heart*



State machine of *VentricularController*

Design Simulation

The screenshot displays the AVATAR simulation interface. On the left, a state transition diagram for the 'VentricularController' block is shown. It includes states like 'WaitRI', 'VPace()', 'VSense()', and 'WaitVRP', with transitions labeled with conditions such as 'RI = LRI' and 'hp = false'. On the right, the simulation control panel shows the simulation is 'Stopped' at 'Time: 80'. Below this, a simulation trace shows the execution of 'VPace()' from the 'VentricularController' block to the 'Heart' block, resulting in a 'Ready' state with parameters 'RI = 25' and 'hp = false'.

Connector from Send signal to state0: List of all parameters of an Avatar SMD transition

Run simulation for x commands. Works only if the simulator is "ready"

- ▶ 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*

Verify with UPPAAL: options

- Search for absence of deadock situations
- Reachability of selected states
- Liveness of selected states
- Custom verification

Custom formulae =

- Generate simulation trace
- Show verification details

Select options and then, click on 'start' to start generation
Session id on launcher=1
Sending UPPAAL specification data

Searching for absence of deadlock situations
-> property is satisfied

Reachability of: VentricularController.Send signal: VPace()
-> property is satisfied

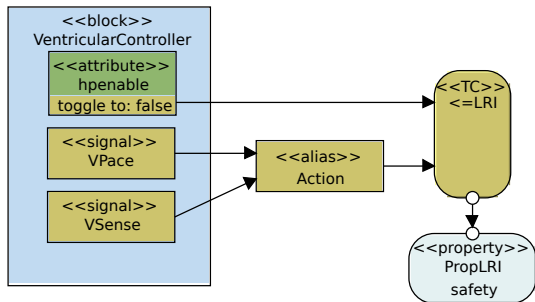
Liveness of: VentricularController.Send signal: VPace()
-> property is NOT satisfied

All Done

TEPE: Expressing Complex Properties

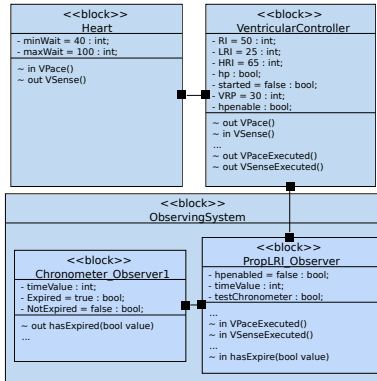
- ▶ Use of SysML parametric diagrams
- ▶ Logical and temporal relations between events and attributes

- ▶ **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



Verify with UPPAAL: options

- Search for absence of deadlock situations
- Reachability of selected states
- Liveness of selected states
- Custom verification

Custom formulae =

- Generate simulation trace
- Show verification details

Select options and then, click on 'start' to start generating the verification results.
 Session id on launcher=1
 Sending UPPAAL specification data

Reachability of: PropLRI_Observer.state0: ErrorPropLRI
 -> property is NOT satisfied

All Done



Outline

Introduction

Requirements analysis of the pacemaker with AVATAR/TTool

Conclusion

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