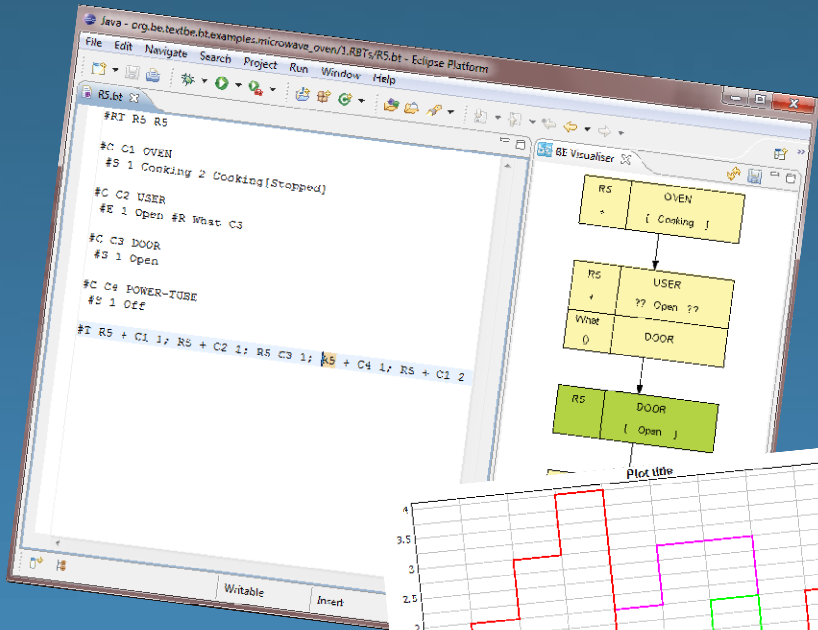
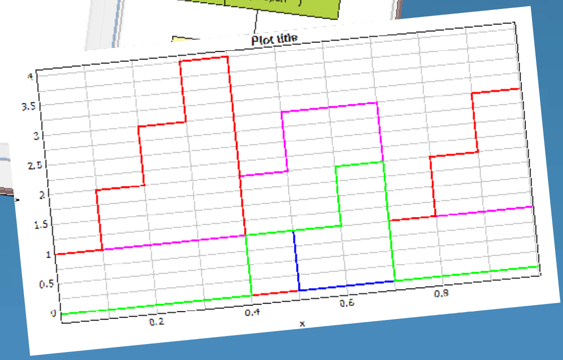
Comodeling Revisited:
Execution of Behavior Trees in Modelica

Toby Myers
Wladimir Schamai
Peter Fritzson

BEHAVIOR ENGINEERING





MODELICA

Outline

- Introduction to Behavior Engineering
- Comodeling Revisited
- Behavior Trees in Modelica
- Complementing Comodeling with vVDR

Outline

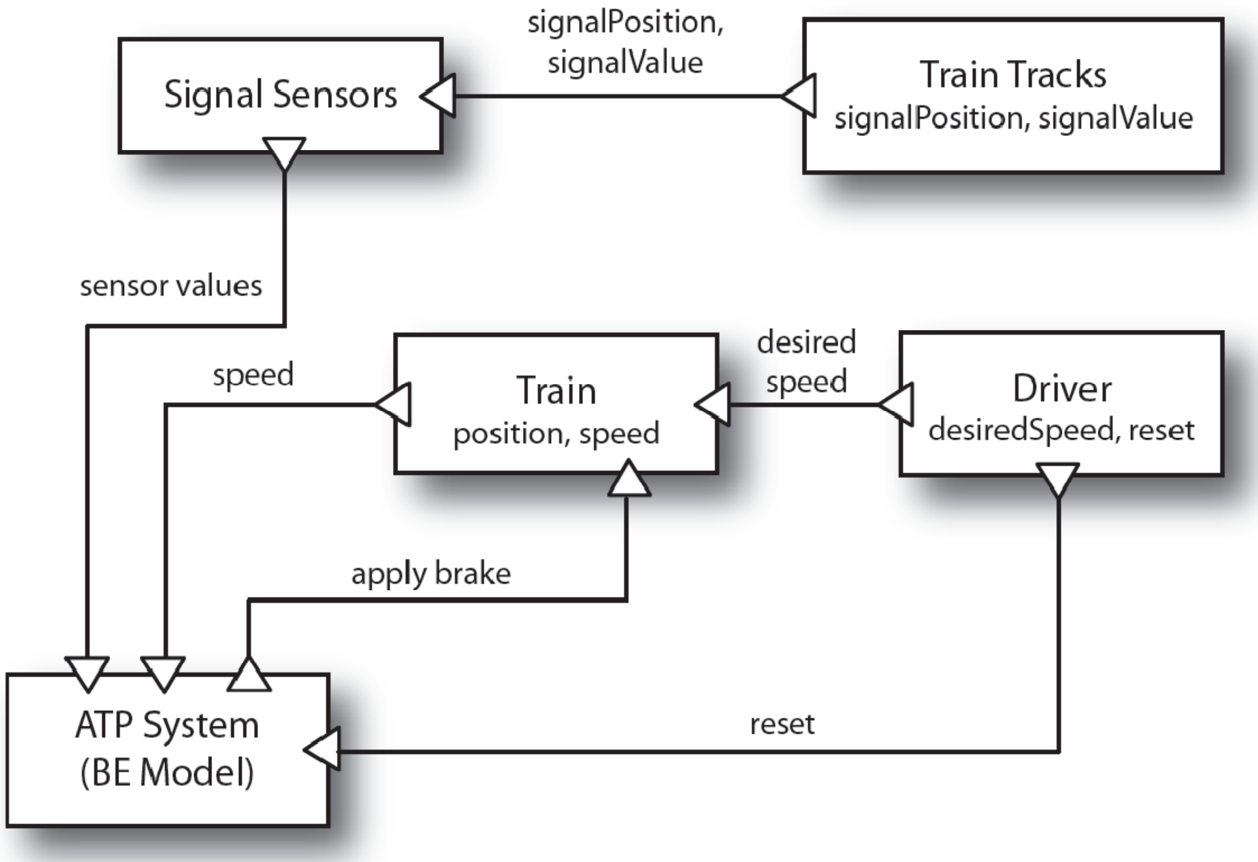
- **Introduction to Behavior Engineering**
- Comodeling Revisited
- Behavior Trees in Modelica
- Complementing Comodeling with vVDR

The Software-Hardware Integration Problem Cyber-Physical System Modeling

- At the **early stages** of system development, many **decisions** must be made about how the **system will be realised** as a combination of Software and Hardware
- **Requirements** of the system at these early stages **lack quantified and temporal information** so it is hard to make an informed decision
- **Changing** the partioning of software / hardware or how they interact later in development can be **time-consuming and costly**
- There is a potential for **errors** and **incompatibility** to be introduced as **software/hardware** specifications are created **independently**

Example: Model of Automated Train Protection System

An ATP System monitors train position and speed, and may apply brakes if the driver does not react in time



Introduction to Behavior Engineering

Behavior Engineering for Requirements Analysis

- 5 large-scale industry projects
  - In Defence, Transportation, Banking and Finance
  - Between 800-1250 requirements
- All previously reviewed with respective organisations' internal review processes
- Defect detection rate approximately 2 to 3 times that of traditional ad-hoc, checklist-based, and scenario-based reading techniques reported in Porter, 1998.

Requirements Evaluation Using Behavior Trees

Findings from Industry

Daniel Powell

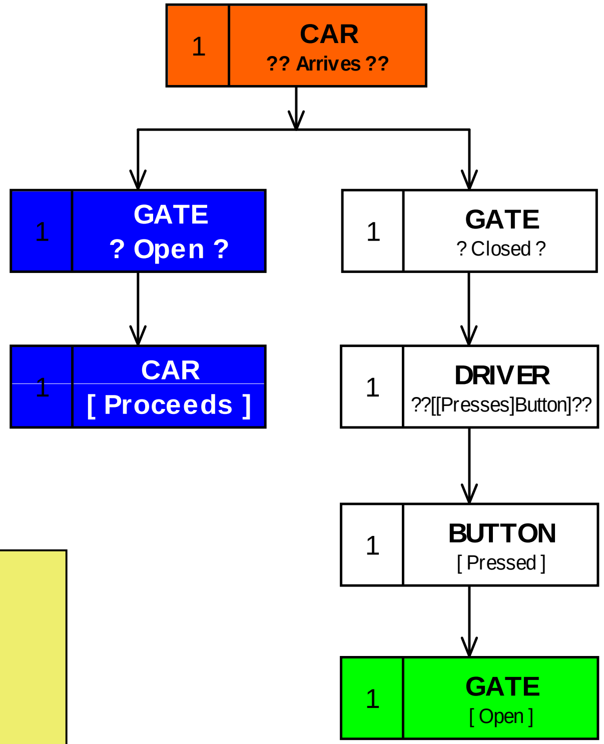
http://aswec07.cs.latrobe.edu.au/5.zip

Formalization - Requirements Translation

Functional Requirement

When a car is arrives,
if the gate is open the car proceeds,
otherwise if the gate is closed, when the driver presses the button
it causes the gate to open

Behavior Tree



Formalization
- clarification and preservation of intent
- strict use of original vocabulary
- removes ambiguity, aliases, etc
- aids stakeholder validation, understanding
- approaches repeatability

A Brief Introduction to Behavior Engineering

- Behavior Engineering is a methodology with a tightly interlinked language and process

Behavior Modeling Process (BMP)	Behavior Modeling Language (BML)	
	Behavior Trees (BT)	**Composition Trees (CT)**
Requirements Translation	Requirement Behavior Trees (RBTs)	Requirement Composition Tree (RCT)
Requirements Integration	Integrated Behavior Tree (IBT)	Integrated Composition Tree (ICT)
System Specification	Model Behavior Tree (MBT)	Model Composition Tree (MCT)
System Design	Design Behavior Tree (DBT)	Design Composition Tree (DCT)

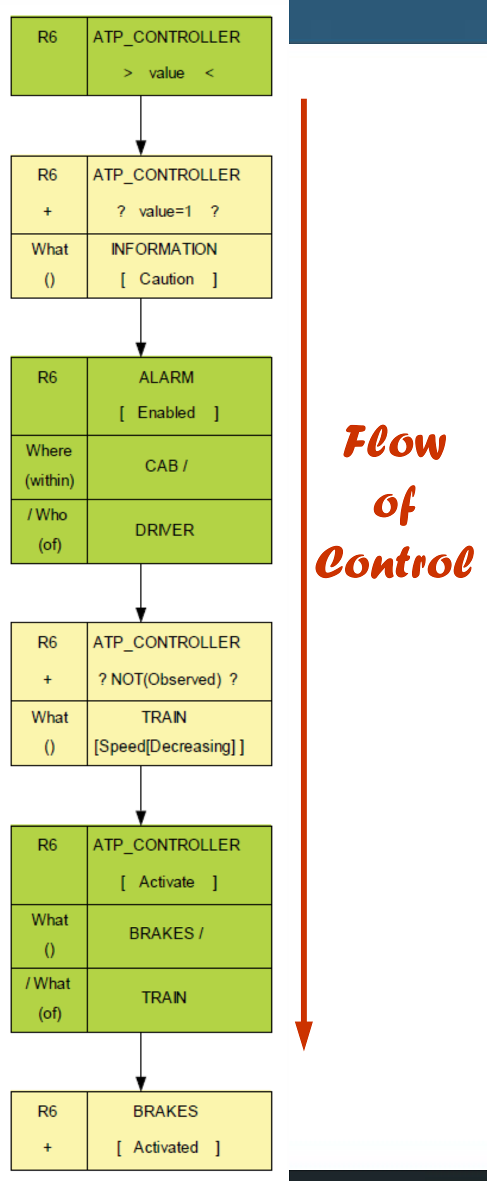
Introduction to Behavior Engineering

How to translate from a Requirement in Natural Language to an RBT

R6. If a caution signal is returned to the ATP controller then the alarm is enabled within the driver's cab. Furthermore, once the alarm has been enabled, if the speed of the train is not observed to be decreasing then the ATP controller activates the train's braking system.

The Tag traces these Behavior Tree nodes back to Requirement 6.

A '+' and a yellow color denote the behavior is implied by the requirements



A Brief Introduction to Behavior Engineering

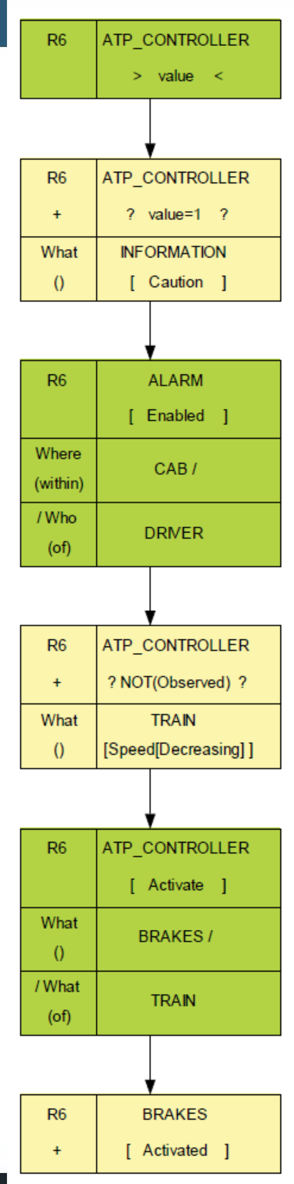
How to translate from a Requirement in Natural Language to an RBT

R6. If a caution signal is returned to the ATP controller then the alarm is enabled within the driver's cab. Furthermore, once the alarm has been enabled, if the speed of the train is not observed to be decreasing then the ATP controller activates the train's braking system.

ATP Controller receives a value from another component

Check if the value is a caution signal

If it is, enable the Alarm. To maintain the intent of the original requirement, use a relation to show the Alarm is enabled in the Driver's Cab.



A Brief Introduction to Behavior Engineering

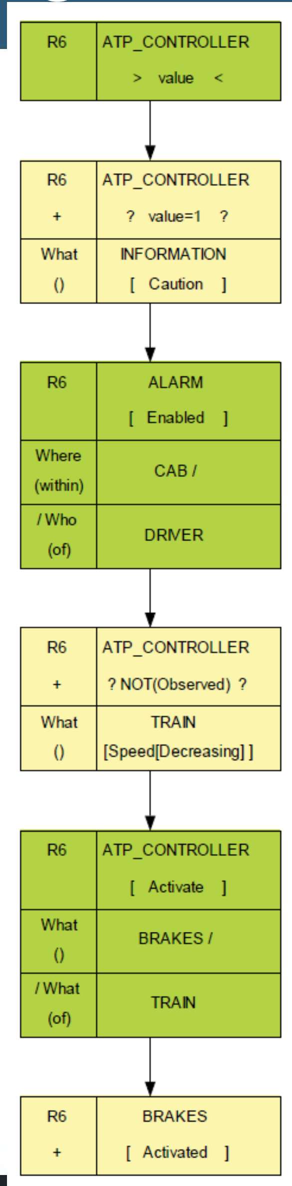
How to translate from a Requirement in Natural Language to an RBT

R6. If a caution signal is returned to the ATP controller then the alarm is enabled within the driver's cab. Furthermore, once the alarm has been enabled, if the speed of the train is not observed to be decreasing then the ATP controller activates the train's braking system.

It is implied the ATP Controller must observe whether the Train's speed is decreasing.

If the Train isn't decreasing in speed, the ATP Controller activates the Braking System of the Train.

..Which results in the Braking System being Activated



Outline

- Introduction to Behavior Engineering
- **Comodeling Revisited**
- Behavior Trees in Modelica
- Complementing Comodeling with vVDR

Comodeling of Cyber-Physical Systems Revisited

- At the **early stages** of system development, many **decisions** must be made about how the **system will be realised** as a combination of Software and Hardware
- **Requirements** of the system at these early stages **lack quantified and temporal information** so it is hard to make an informed decision
- **Changing** the partioning of software / hardware or how they interact later in development can be **time-consuming and costly**
- There is a potential for **errors** and **incompatibility** to be introduced as **software/hardware** specifications are created **independently**

Comodeling Revisited

Requirement	Description
R1	The ATP system is located on board the train. It involves a central controller and five boundary subsystems that manage the sensors, speedometer, brakes, alarm and a reset mechanism.
R2	The sensors are attached to the side of the train and detect information on the approach to track-side signals, i.e. they detect what the signal is displaying to the train driver.
R3	In order to reduce the effects of component failure three sensors are used. Each sensor generates a value in the range 0 to 3, where 0, 1 and 2 denote the danger, caution, and proceed signals respectively. The fourth sensor value, i.e. 3, is generated if an undefined signal is detected, e.g. may correspond to noise between the signal and the sensor.
R4	The sensor value returned to the ATP controller is calculated as the majority of the three sensor readings. If there does not exist a majority then an undefined value is returned to the ATP controller.
R5	If a proceed signal is returned to the ATP controller then no action is taken with respect to the train's brakes.
R6	If a caution signal is returned to the ATP controller then the alarm is enabled within the driver's cab. Furthermore, once the alarm has been enabled, if the speed of the train is not observed to be decreasing then the ATP controller activates the train's braking system.
R7	In the case of a danger signal being returned to the ATP controller, the braking system is immediately activated and the alarm is enabled. Once enabled, the alarm is disabled if a proceed signal is subsequently returned to the ATP controller.
R8	Note that if the braking system is activated then the ATP controller ignores all sensor input until the system has been reset.
R9	If enabled, the reset mechanism deactivates the train's brakes and disables the alarm.

Table 1. Requirements of the ATP system

Comodeling Revisited

Interaction with Sensors ...

Requirement	Description
R1	The ATP system is located on board the train. It involves a central controller and five boundary subsystems that manage the sensors, speedometer, brakes, alarm and a reset mechanism.
R2	The sensors are attached to the side of the train and detect information on the approach to track-side signals, i.e. they detect what the signal is displaying to the train driver.
R3	In order to reduce the effects of component failure three sensors are used. Each sensor generates a value in the range 0 to 3, where 0, 1 and 2 denote the danger, caution, and proceed signals respectively. The fourth sensor value, i.e. 3, is generated if an undefined signal is detected, e.g. may correspond to noise between the signal and the sensor.
R4	The sensor value returned to the ATP controller is calculated as the majority of the three sensor readings. If there does not exist a majority then an undefined value is returned to the ATP controller.
R5	If a proceed signal is returned to the ATP controller then no action is taken with respect to the train's brakes.
R6	If a caution signal is returned to the ATP controller then the alarm is enabled within the driver's cab. Furthermore, once the alarm has been enabled, if the speed of the train is not observed to be decreasing then the ATP controller activates the train's braking system.
R7	In the case of a danger signal being returned to the ATP controller, the braking system is immediately activated and the alarm is enabled. Once enabled, the alarm is disabled if a proceed signal is subsequently returned to the ATP controller.
R8	Note that if the braking system is activated then the ATP controller ignores all sensor input until the system has been reset.
R9	If enabled, the reset mechanism deactivates the train's brakes and disables the alarm.

Table 1. Requirements of the ATP system

How often does this need to be checked?

Decreasing by how much?

Comodeling Revisited

Interaction with Actuators …

Requirement	Description
R1	The ATP system is located on board the train. It involves a central controller and five boundary subsystems that manage the sensors, speedometer, brakes, alarm and a reset mechanism.
R2	The sensors are attached to the side of the train and detect information on the approach to track-side signals, i.e. they detect what the signal is displaying to the train driver.
R3	In order to reduce the effects of component failure three sensors are used. Each sensor generates a value in the range 0 to 3, where 0, 1 and 2 denote the danger, caution, and proceed signals respectively. The fourth sensor value, i.e. 3, is generated if an undefined signal is detected, e.g. may correspond to noise between the signal and the sensor.
R4	The sensor value returned to the ATP controller is calculated as the majority of the three sensor readings. If there does not exist a majority then an undefined value is returned to the ATP controller.
R5	If a proceed signal is returned to the ATP controller then no action is taken with respect to the train's brakes.
R6	If a caution signal is returned to the ATP controller then the alarm is enabled within the driver's cab. Furthermore, once the alarm has been enabled, if the speed of the train is not observed to be decreasing then the ATP controller activates the train's braking system.
R7	In the case of a danger signal being returned to the ATP controller, the braking system is immediately activated and the alarm is enabled. Once enabled, the alarm is disabled if a proceed signal is subsequently returned to the ATP controller.
R8	Note that if the braking system is activated then the ATP controller ignores all sensor input until the system has been reset.
R9	If enabled, the reset mechanism deactivates the train's brakes and disables the alarm.

Table 1. Requirements of the ATP system

What response time is realistically acceptable?

Comodeling Revisited

Software / Hardware Partitioning …

Requirement	Description
R1	The ATP system is located on board the train. It involves a central controller and five boundary subsystems that manage the sensors, speedometer, brakes, alarm and a reset mechanism.
R2	The sensors are attached to the side of the train and detect information on the approach to track-side signals, i.e. they detect what the signal is displaying to the train driver.
R3	In order to reduce the effects of component failure three sensors are used. Each sensor generates a value in the range 0 to 3, where 0, 1 and 2 denote the danger, caution, and proceed signals respectively. The fourth sensor value, i.e. 3, is generated if an undefined signal is detected, e.g. may correspond to noise between the signal and the sensor.
R4	The sensor value returned to the ATP controller is calculated as the majority of the three sensor readings. If there does not exist a majority then an undefined value is returned to the ATP controller.
R5	If a proceed signal is returned to the ATP controller then no action is taken with respect to the train's brakes.
R6	If a caution signal is returned to the ATP controller then the alarm is enabled within the driver's cab. Furthermore, once the alarm has been enabled, if the speed of the train is not observed to be decreasing then the ATP controller activates the train's braking system.
R7	In the case of a danger signal being returned to the ATP controller, the braking system is immediately activated and the alarm is enabled. Once enabled, the alarm is disabled if a proceed signal is subsequently returned to the ATP controller.
R8	Note that if the braking system is activated then the ATP controller ignores all sensor input until the system has been reset.
R9	If enabled, the reset mechanism deactivates the train's brakes and disables the alarm.

Table 1. Requirements of the ATP system

Perform in Software or Hardware?

Comodeling Revisited

The Environment in which the system will exist …

Requirement	Description
R1	The ATP system is located on board the train. It involves a central controller and five boundary subsystems that manage the sensors, speedometer, brakes, alarm and a reset mechanism.
R2	The sensors are attached to the side of the train and detect information on the approach to track-side signals, i.e. they detect what the signal is displaying to the train driver.
R3	In order to reduce the effects of component failure three sensors are used. Each sensor generates a value in the range 0 to 3, where 0, 1 and 2 denote the danger, caution, and proceed signals respectively. The fourth sensor value, i.e. 3, is generated if an undefined signal is detected, e.g. may correspond to noise between the signal and the sensor.
R4	The sensor value returned to the ATP controller is calculated as the majority of the three sensor readings. If there does not exist a majority then an undefined value is returned to the ATP controller.
R5	If a proceed signal is returned to the ATP controller then no action is taken with respect to the train's brakes.
R6	If a caution signal is returned to the ATP controller then the alarm is enabled within the driver's cab. Furthermore, once the alarm has been enabled, if the speed of the train is not observed to be decreasing then the ATP controller activates the train's braking system.
R7	In the case of a danger signal being returned to the ATP controller, the braking system is immediately activated and the alarm is enabled. Once enabled, the alarm is disabled if a proceed signal is subsequently returned to the ATP controller.
R8	Note that if the braking system is activated then the ATP controller ignores all sensor input until the system has been reset.
R9	If enabled, the reset mechanism deactivates the train's brakes and disables the alarm.

Table 1. Requirements of the ATP system

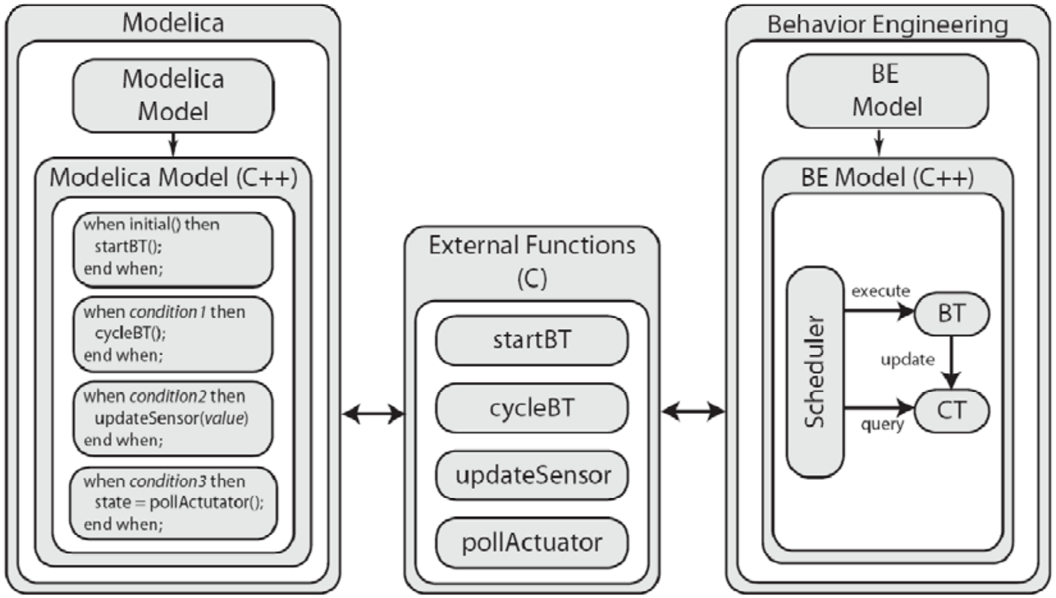
How far apart are the signals?

What are the characteristics of the train?

Will it be deployed on many different types of trains?

Comodeling Revisited

- Previous implementation of Comodeling
  – BE and Modelica models executed separately
  – Integrated using Modelica external functions



Outline

- Introduction to Behavior Engineering
- Comodeling Revisited
- **Behavior Trees in Modelica**
- Complementing Comodeling with vVDR

Behavior Trees in Modelica

- Why execute Behavior Trees in Modelica?
  1. Comodeling is easier to apply as the comodel is captured solely in Modelica
     - A Behavior Tree can directly **affect the acausal equations** used to model the hardware and environmental components
     - Modelica is used as an **action language** for the whole integrated Co-model
  2. Portable models – the ability to execute Behavior Trees is available to a wide audience, several tools
  3. Comodeling can be used with other complementary approaches such as the **virtual verification of system designs** against system requirements

Behavior Trees in Modelica

- Representing Behavior Trees in Modelica
  - Basic Nodes
    - State Realisation, Selection, Guard, Input, Output
  - Branching & Composition
    - Sequential Composition, Atomic Composition, Parallel Branching, Alternative Branching
  - Operators
    - Reference, Reversion, Branch-Kill, Synchronisation

Behavior Trees in Modelica

- State Realisation
  - A state realisation updates the *state* variable of the associated component to the enumerated value of the behavior of the BT node.

```
c.state := Integer(c.states.s);
```

Behavior Trees in Modelica

- Selection
  - A selection performs an equality check on the *state* variable of the associated component, comparing it to the enumerated value of the behavior of the BT node. Depending on the result of this equality check, the flow of control either continues or is terminated.

```
if c.state == c.state_s then
  ... // Continue flow of control
else
  ... // Terminate branch
end if;
```

Behavior Trees in Modelica

- Guard
  - A guard is similar to a selection, with the exception that the else branch is not included to ensure that the guard is continually re-evaluated until true.

```
if c.state == c.state_s then
  ... // Continue flow of control
end if;
```

Behavior Trees in Modelica

- Input
  - Inputs and outputs are implemented as boolean variables. When the variable is true, the input is active. Events last for one cycle, to ensure that if an internal output is active in one branch, it can be received by an internal output in another branch. Inputs are represented with an equality check that is true if the associated event becomes active.

```
if e2 then
  ... // Continue flow of control
end if;
```

Behavior Trees in Modelica

- Sequential Composition
  - Updates the value of the branch variable

```
if branch1 == 1 then
  ... // Node behavior
  branch1 := 2;
elseif ...
```

Behavior Trees in Modelica

- Parallel Branching
  - Clears the current branch value and sets the branch value of the child branches to their first node.

```
if branch1 == 1 then
  ... // Node behavior
  branch1 := 0;
  branch2 := 1;
  branch3 := 1;
elseif ...
```

Behavior Trees in Modelica

- Alternative Branching
  - As per parallel branch, but when the first node of any of the child branches is activated all the sibling branches are terminated.

```
if branch2 == 1 then
  ... // Node behavior
  branch2 := 2;
  branch3 := 0;
end if;
```

Behavior Trees in Modelica

- Atomic Composition
  - Adds further constraint to all sibling branches of atomic composed nodes that flow of control cannot continue if the branch values of the atomic composed nodes are active.

```
if branch3 == 1 and not(branch2==1 or
   branch2==2) then
  ... // Node behavior
  branch3 := 2;
```

Behavior Trees in Modelica

- Reference
  - Clears the current branch value and sets the branch value of the destination node.

```
if branch3 == 1 then
  branch3 := 0;
  branch2 := 2;
```

Behavior Trees in Modelica

- Reversion
  - Clears the current branch value and all sibling parallel branches and sets the branch value of the destination node.

```
if branch1 == 3 then
  branch2 := 0;
  branch3 := 0;
  branch1 := 1;
```

Behavior Trees in Modelica

- Branch-Kill
  - Clears the branch value of the destination node and the branch value of any of its descendants.

```
if branch1 == 2 then
  branch3 := 0;
  … // Continue flow of control
```

Behavior Trees in Modelica

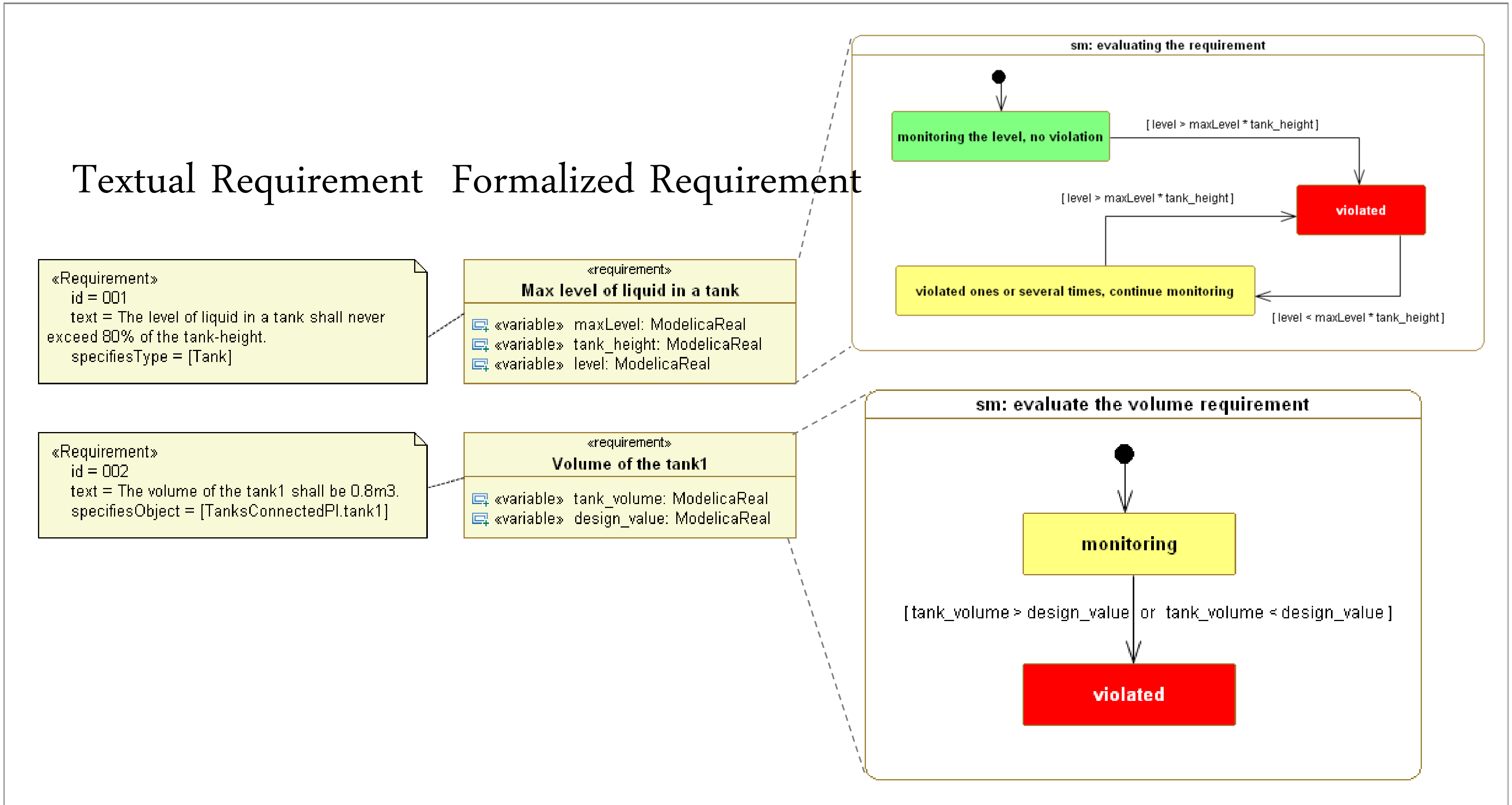
- Synchronisation
  - One synchronisation node checks when the branch value of all nodes is set correctly and sets a boolean variable to true. All other synchronisation nodes wait until this Boolean variable is true.

```
if branch3 == 2 and branch2 == 3 then
  sync1 := true;
  ... // Node behavior
  ... // Continue flow of control
if branch2 == 3 and sync1 then
  ... // Continue flow of control
```

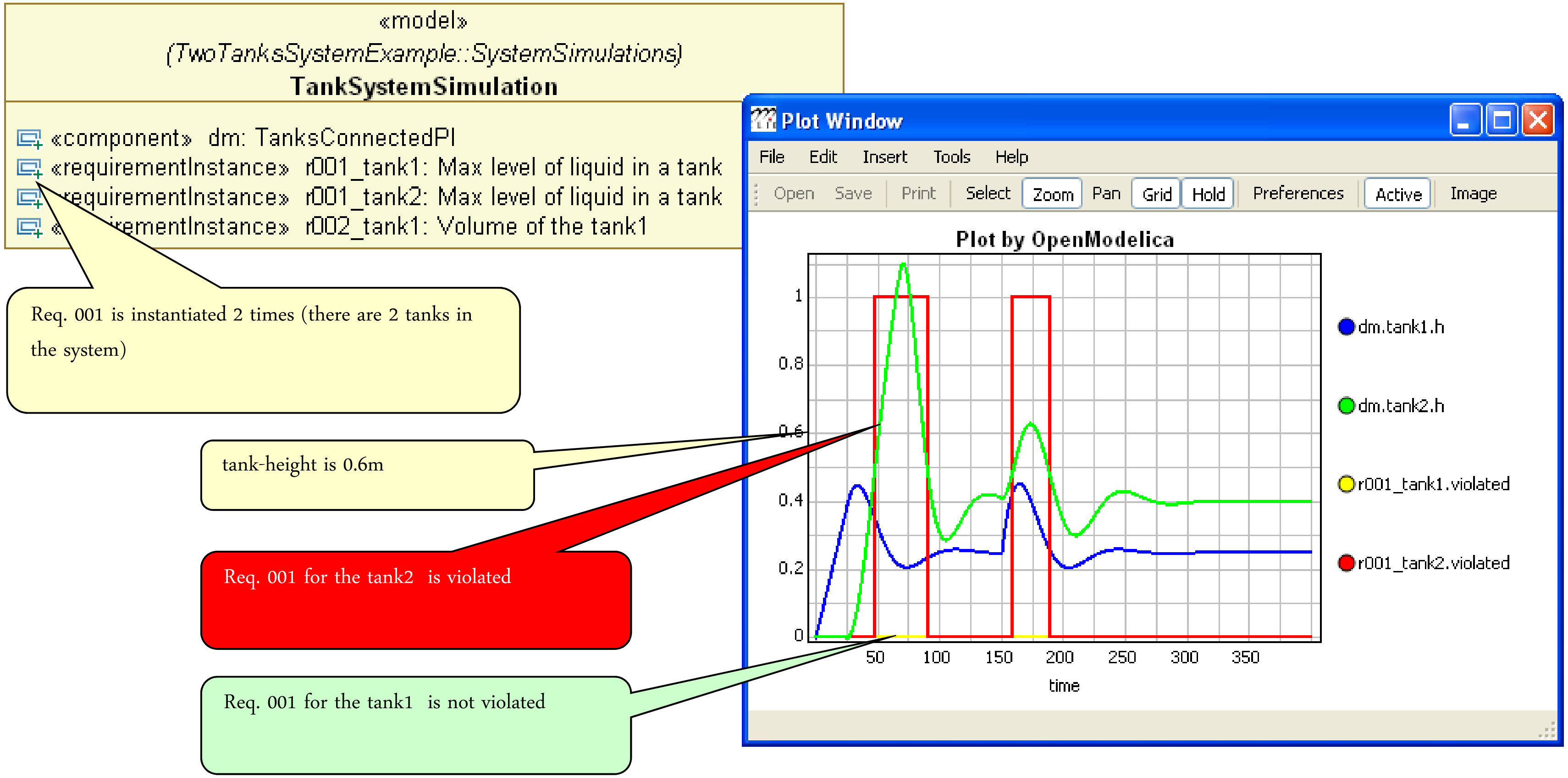
Outline

- Introduction to Behavior Engineering
- Comodeling Revisited
- Behavior Trees in Modelica
- **Complementing Comodeling with vVDR**

vVDR – Virtual Verification of Design Requirements ModelicaML Example: System Requirements



vVDR – Virtual Verification of Design Requirements in ModelicaML Simulation and Requirements Check



Complementing Comodeling with vVDR

- vVDR and BE provide differing benefits
  - vVDR can verify different system design alternatives against the same set of requirements or drive verifications driven by different test scenarios.
  - BE provides a means to ensure the consistency of a specification.
- Their combination could leverage the benefits of both approaches

Complementing Comodeling with vVDR

- vVDR is a method for virtual Verification of system Design alternatives against system Requirements
- In vVDR each requirement is formalised as a model to evaluate violation and fulfilment criteria
- These requirement models and a system design are instantiated and bound into a test model which is translated into Modelica

Complementing Comodeling with vVDR

- Two possible applications
  1. Integrate BE with vVDR by using a model behavior tree as the source for generation of both vVDR requirements violation monitors and test cases
  2. Augment the existing comodeling approach with vVDR. vVDR provides monitors that evaluate the performance of different comodels to find the best candidate that fulfils a set of criteria

Implementation Aspects

- Use declarative (equations) or imperative (algorithms) constructs?
- Create a Modelica library or create a code generator?

- In both cases, Behavior Trees in Modelica, and ModelicaML state charts, a code generator to algorithmic Modelica was implemented

Thank-you

- Any Questions?

- vVDR support in ModelicaML available from www.openmodelica.org
- A little plug ...
  - www.tjmyersconsulting.com
  - Specialise in practicing Behavior Engineering