

MOVEP 2012 Tutorial

Safety, Dependability and Performance Analysis of Extended AADL Models

Part 2: System Modeling Using AADL



European Space Agency
European Space Research and Technology Centre



RWTH Aachen University
Software Modeling and Verification Group
Thomas Noll



Fondazione Bruno Kessler
Centre for Scientific and Technological Research
Alessandro Cimatti

MOVEP 2012 School; December 7, 2012; Marseille, France

- 1 Representing Nominal System Behavior
- 2 Formal Semantics
- 3 Modeling Error Behavior
- 4 Tool Support
- 5 References & Ongoing Activities

- 1 Representing Nominal System Behavior
- 2 Formal Semantics
- 3 Modeling Error Behavior
- 4 Tool Support
- 5 References & Ongoing Activities

The Industry Standard AADL

- **1989** MetaH

Paradigm

- Architecture-based and model-driven top-down and bottom-up engineering
- Real-time and performance critical distributed systems
- Complements component-based product-line development

- **1998** SAE AS-2C

- **2004** AADL 1.0

- **2006** Error Annex 1.0

- **2009** AADL 2.0

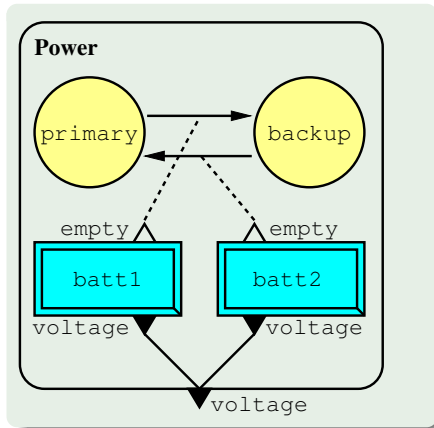
- **2010** Error Annex 2.0



AADL Example: Redundant Power System

Redundant power system:

- contains two batteries **batt1/batt2**
- used in **primary/backup** mode
- power switches from **primary** to **backup** (and back) when **batt1** (**batt2**) empty
- additionally provides **voltage** information



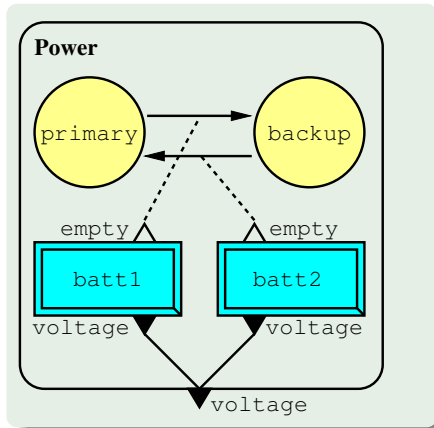
AADL Example: Redundant Power System

Redundant power system:

- contains two batteries **batt1/batt2**
- used in **primary/backup** mode
- power switches from **primary** to **backup** (and back) when **batt1** (**batt2**) empty
- additionally provides **voltage** information

We shall show:

- hybrid behavior of the **batteries**
- composition of the **power system**
- **semantics** as transition systems
- interweaving of **errors**



Modeling a Battery

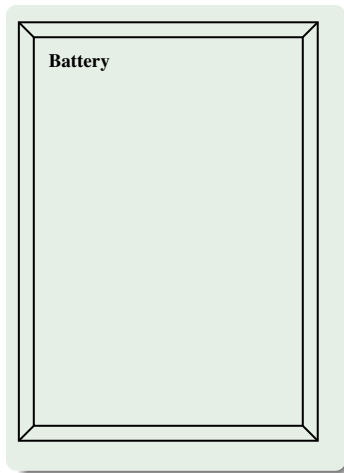
Component **type** and **implementation**:

```
device Battery
```

```
end Battery;
```

```
device implementation Battery.Imp
```

```
end Battery.Imp;
```



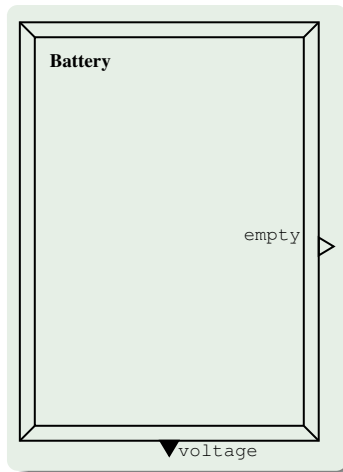
Modeling a Battery

Type defines the **interface**:

```
device Battery
  features
    empty: out event port;
    voltage: out data port real default 6.0;
end Battery;

device implementation Battery.Imp
```

```
end Battery.Imp;
```



Modeling a Battery

Adding **modes** behavior:

```
device Battery
  features
    empty: out event port;
    voltage: out data port real default 6.0;
end Battery;
```

```
device implementation Battery.Imp
```

modes

charged: activation mode

depleted: mode

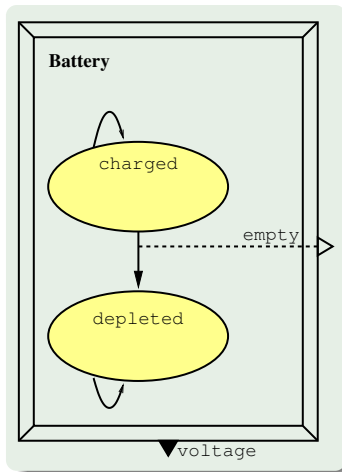
transitions

charged -[]-> charged;

charged -[empty]-> depleted;

depleted -[]-> depleted;

```
end Battery.Imp;
```

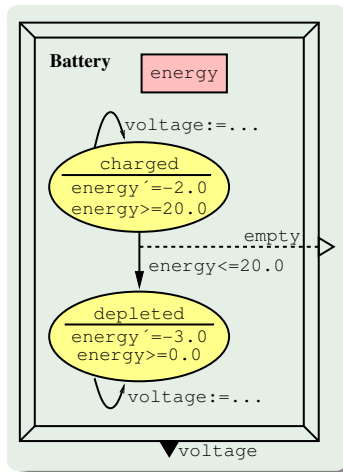


Modeling a Battery

Adding **hybrid** behavior:

```
device Battery
  features
    empty: out event port;
    voltage: out data port real default 6.0;
  end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous default 100.0;
  modes
    charged: activation mode
      while energy'=-2.0 and energy>=20.0;
    depleted: mode
      while energy'=-3.0 and energy>=0.0;
  transitions
    charged -[then voltage:=energy/50.0+4.0]-> charged;
    charged -[empty when energy<=20.0]-> depleted;
    depleted -[then voltage:=energy/50.0+4.0]-> depleted;
  end Battery.Imp;
```



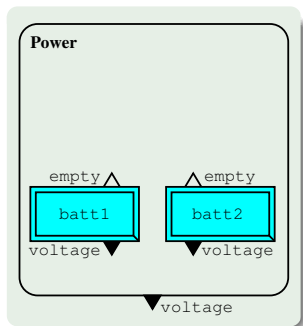
Modeling the Redundant Power System

Power system with **battery subcomponents**:

```
system Power
  features
    voltage: out data port real;
  end Power;
```

```
system implementation Power.Imp
  subcomponents
    batt1: device Battery.Imp
    batt2: device Battery.Imp
```

```
end Power.Imp;
```



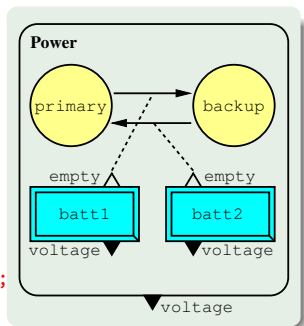
Modeling the Redundant Power System

Adding dynamic reconfiguration:

```
system Power
  features
    voltage: out data port real;
  end Power;

system implementation Power.Imp
  subcomponents
    batt1: device Battery.Imp in modes (primary);
    batt2: device Battery.Imp in modes (backup);

  modes
    primary: initial mode;
    backup: mode;
  transitions
    primary -[batt1.empty]-> backup;
    backup -[batt2.empty]-> primary;
  end Power.Imp;
```

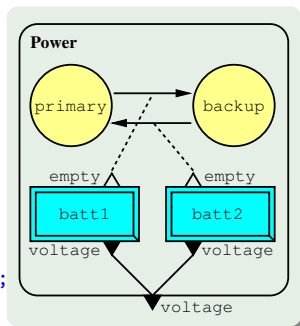


Modeling the Redundant Power System

Adding port connections:

```
system Power
  features
    voltage: out data port real;
  end Power;

system implementation Power.Imp
  subcomponents
    batt1: device Battery.Imp in modes (primary);
    batt2: device Battery.Imp in modes (backup);
  connections
    data port batt1.voltage -> voltage in modes (primary);
    data port batt2.voltage -> voltage in modes (backup);
  modes
    primary: initial mode;
    backup: mode;
  transitions
    primary -[batt1.empty]-> backup;
    backup -[batt2.empty]-> primary;
end Power.Imp;
```



- 1 Representing Nominal System Behavior
- 2 Formal Semantics**
- 3 Modeling Error Behavior
- 4 Tool Support
- 5 References & Ongoing Activities

Operational Semantics of Single Components

- **States:** (mode, data values)
- **Transitions:** timed or internal or synchronized

Operational Semantics of Single Components

- States: (mode, data values)
- Transitions: timed or internal or synchronized

Example (Battery)

`⟨mode = charged, energy = 100.0, voltage = 6.0⟩`

Operational Semantics of Single Components

- States: (mode, data values)
- Transitions: **timed** or internal or synchronized

Example (Battery)

$\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle$

↓ 30.0

$\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle$

Operational Semantics of Single Components

- States: (mode, data values)
- Transitions: timed or **internal** or synchronized

Example (Battery)

```
⟨mode = charged, energy = 100.0, voltage = 6.0⟩  
    ↓ 30.0  
⟨mode = charged, energy = 40.0, voltage = 6.0⟩  
    ↓ τ⟨voltage:=...⟩  
⟨mode = charged, energy = 40.0, voltage = 4.8⟩
```

Operational Semantics of Single Components

- States: (mode, data values)
- Transitions: **timed** or internal or synchronized

Example (Battery)

```
⟨mode = charged, energy = 100.0, voltage = 6.0⟩  
    ↓ 30.0  
⟨mode = charged, energy = 40.0, voltage = 6.0⟩  
    ↓ τ⟨voltage:=...⟩  
⟨mode = charged, energy = 40.0, voltage = 4.8⟩  
    ↓ 10.0  
⟨mode = charged, energy = 20.0, voltage = 4.8⟩
```

Operational Semantics of Single Components

- States: (mode, data values)
- Transitions: timed or **internal** or synchronized

Example (Battery)

```
⟨mode = charged, energy = 100.0, voltage = 6.0⟩  
    ↓ 30.0  
⟨mode = charged, energy = 40.0, voltage = 6.0⟩  
    ↓ τ⟨voltage:=...⟩  
⟨mode = charged, energy = 40.0, voltage = 4.8⟩  
    ↓ 10.0  
⟨mode = charged, energy = 20.0, voltage = 4.8⟩  
    ↓ τ⟨voltage:=...⟩  
⟨mode = charged, energy = 20.0, voltage = 4.4⟩
```

Operational Semantics of Single Components

- States: (mode, data values)
- Transitions: timed or internal or **synchronized**

Example (Battery)

```
⟨mode = charged, energy = 100.0, voltage = 6.0⟩  
    ↓ 30.0  
⟨mode = charged, energy = 40.0, voltage = 6.0⟩  
    ↓ τ⟨voltage:=...⟩  
⟨mode = charged, energy = 40.0, voltage = 4.8⟩  
    ↓ 10.0  
⟨mode = charged, energy = 20.0, voltage = 4.8⟩  
    ↓ τ⟨voltage:=...⟩  
⟨mode = charged, energy = 20.0, voltage = 4.4⟩  
    ↓ empty  
⟨mode = depleted, energy = 20.0, voltage = 4.4⟩
```

Operational Semantics of Single Components

- States: (mode, data values)
- Transitions: timed or internal or synchronized

Example (Battery)

$\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle$
 $\downarrow 30.0$
 $\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle$
 $\downarrow \tau \langle \text{voltage} := \dots \rangle$
 $\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle$
 $\downarrow 10.0$
 $\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.8 \rangle$
 $\downarrow \tau \langle \text{voltage} := \dots \rangle$
 $\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle$
 $\downarrow \text{empty}$
 $\langle \text{mode} = \text{depleted}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle$
 $\downarrow \dots$

The Power System Revisited

```
system Power
  features
    voltage: out data port real;
  end Power;

system implementation Power.Imp
  subcomponents
    batt1: device Battery.Imp in modes (primary);
    batt2: device Battery.Imp in modes (backup);
  connections
    data port batt1.voltage -> voltage
      in modes (primary);
    data port batt2.voltage -> voltage
      in modes (backup);
  modes
    primary: initial mode;
    backup: mode;
  transitions
    primary -[batt1.empty]-> backup;
    backup -[batt2.empty]-> primary;
end Power.Imp;
```

Operational Semantics of Systems

- **States:** (mode, data values)⁺
- **Transitions** determined by active components:
 - ① Perform local transitions:
 - timed local transition in all components or
 - internal transition in component or
 - multi-way event communication from component to ≥ 1 connected components
 - ② Initialize (re-)activated subcomponents
 - ③ Evaluate data connections (copy source \rightarrow target data port)

Operational Semantics of Systems

- States: (mode, data values)⁺
- Transitions determined by active components:
 - 1 Perform local transitions:
 - timed local transition in all components or
 - internal transition in component or
 - multi-way event communication from component to ≥ 1 connected components
 - 2 Initialize (re-)activated subcomponents
 - 3 Evaluate data connections (copy source \rightarrow target data port)

Example (Power system)

$\langle m = \text{primary}, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle$

Operational Semantics of Systems

- States: (mode, data values)⁺
- Transitions determined by active components:
 - 1 Perform local transitions:
 - **timed local transition in all components** or
 - internal transition in component or
 - multi-way event communication from component to ≥ 1 connected components
 - 2 Initialize (re-)activated subcomponents
 - 3 Evaluate data connections (copy source \rightarrow target data port)

Example (Power system)

$\langle m = \underline{\text{primary}}, v = 6.0 \rangle \mid \langle m = \underline{\text{charged}}, e = 100.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle$
 $\quad \quad \quad \downarrow 40.0$
 $\langle m = \underline{\text{primary}}, v = 6.0 \rangle \mid \langle m = \underline{\text{charged}}, e = 20.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle$

Operational Semantics of Systems

- States: (mode, data values)⁺
- Transitions determined by active components:
 - 1 Perform local transitions:
 - timed local transition in all components or
 - **internal transition in component** or
 - multi-way event communication from component to ≥ 1 connected components
 - 2 Initialize (re-)activated subcomponents
 - 3 **Evaluate data connections** (copy source \rightarrow target data port)

Example (Power system)

$$\begin{aligned} &\langle m = \underline{\text{primary}}, v = 6.0 \rangle \mid \langle m = \underline{\text{charged}}, e = 100.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\ &\quad \downarrow 40.0 \\ &\langle m = \underline{\text{primary}}, v = 6.0 \rangle \mid \langle m = \underline{\text{charged}}, e = 20.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\ &\quad \downarrow \tau \langle \text{voltage} := \dots \rangle \\ &\langle m = \underline{\text{primary}}, v = 4.4 \rangle \mid \langle m = \underline{\text{charged}}, e = 20.0, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \end{aligned}$$

Operational Semantics of Systems

- States: (mode, data values)⁺
- Transitions determined by active components:
 - 1 Perform local transitions:
 - timed local transition in all components or
 - internal transition in component or
 - multi-way event communication from component to ≥ 1 connected components
 - 2 Initialize (re-)activated subcomponents
 - 3 Evaluate data connections (copy source \rightarrow target data port)

Example (Power system)

$$\begin{aligned} &\langle m = \underline{\text{primary}}, v = 6.0 \rangle \mid \langle m = \underline{\text{charged}}, e = 100.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\ &\quad \Downarrow 40.0 \\ &\langle m = \underline{\text{primary}}, v = 6.0 \rangle \mid \langle m = \underline{\text{charged}}, e = 20.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\ &\quad \Downarrow \tau \langle \text{voltage} := \dots \rangle \\ &\langle m = \underline{\text{primary}}, v = 4.4 \rangle \mid \langle m = \underline{\text{charged}}, e = 20.0, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\ &\quad \Downarrow \tau \langle \text{empty} \rangle \\ &\langle m = \underline{\text{backup}}, v = 6.0 \rangle \mid \langle m = \underline{\text{depleted}}, e = 20.0, v = 4.4 \rangle \mid \langle m = \underline{\text{charged}}, e = 100.0, v = 6.0 \rangle \end{aligned}$$

Operational Semantics of Systems

- States: (mode, data values)⁺
- Transitions determined by active components:
 - 1 Perform local transitions:
 - **timed local transition in all components** or
 - internal transition in component or
 - multi-way event communication from component to ≥ 1 connected components
 - 2 Initialize (re-)activated subcomponents
 - 3 Evaluate data connections (copy source \rightarrow target data port)

Example (Power system)

$$\begin{array}{l}
\langle m = \text{primary}, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\quad \Downarrow 40.0 \\
\langle m = \text{primary}, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\quad \Downarrow \tau \langle \text{voltage} := \dots \rangle \\
\langle m = \text{primary}, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 20.0, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\quad \Downarrow \tau \langle \text{empty} \rangle \\
\langle m = \text{backup}, v = 6.0 \rangle \mid \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\quad \Downarrow 40.0 \\
\langle m = \text{backup}, v = 6.0 \rangle \mid \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle
\end{array}$$

Operational Semantics of Systems

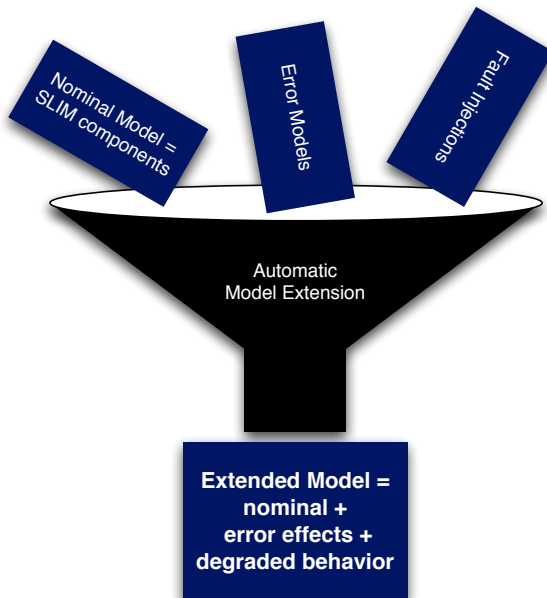
- States: (mode, data values)⁺
- Transitions determined by active components:
 - 1 Perform local transitions:
 - timed local transition in all components or
 - internal transition in component or
 - multi-way event communication from component to ≥ 1 connected components
 - 2 Initialize (re-)activated subcomponents
 - 3 Evaluate data connections (copy source \rightarrow target data port)

Example (Power system)

$$\begin{array}{c}
\langle m = \underline{\text{primary}}, v = 6.0 \rangle \mid \langle m = \underline{\text{charged}}, e = 100.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow 40.0 \\
\langle m = \underline{\text{primary}}, v = 6.0 \rangle \mid \langle m = \underline{\text{charged}}, e = 20.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow \tau \langle \text{voltage} := \dots \rangle \\
\langle m = \underline{\text{primary}}, v = 4.4 \rangle \mid \langle m = \underline{\text{charged}}, e = 20.0, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow \tau \langle \text{empty} \rangle \\
\langle m = \underline{\text{backup}}, v = 6.0 \rangle \mid \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \mid \langle m = \underline{\text{charged}}, e = 100.0, v = 6.0 \rangle \\
\downarrow 40.0 \\
\langle m = \underline{\text{backup}}, v = 6.0 \rangle \mid \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \mid \langle m = \underline{\text{charged}}, e = 20.0, v = 6.0 \rangle \\
\downarrow \dots
\end{array}$$

- 1 Representing Nominal System Behavior
- 2 Formal Semantics
- 3 Modeling Error Behavior**
- 4 Tool Support
- 5 References & Ongoing Activities

Integrating Erroneous and Nominal Behavior



Error Modeling

```
error model BatteryFailure
  features
    ok: initial state;
    dead: error state;
    batteryDied: out error propagation;
end BatteryFailure;

error model implementation BatteryFailure.Imp
  events
    fault: error event occurrence poisson 0.01;
  transitions
    ok -[fault]-> dead;
    dead -[batteryDied]-> dead;
end BatteryFailure.Imp;
```

Error Modeling

```
error model BatteryFailure
  features
    ok: initial state;
    dead: error state;
    batteryDied: out error propagation;
  end BatteryFailure;

error model implementation BatteryFailure.Imp
  events
    fault: error event occurrence poisson 0.01;
  transitions
    ok -[fault]-> dead;
    dead -[batteryDied]-> dead;
  end BatteryFailure.Imp;
```

Fault injection

An error model does not influence the nominal behavior unless they are linked through **fault injection**: (s, d, a) means that on entering error state s , the assignment $d := a$ is performed, where d is a data element and a the fault effect.

Error Modeling

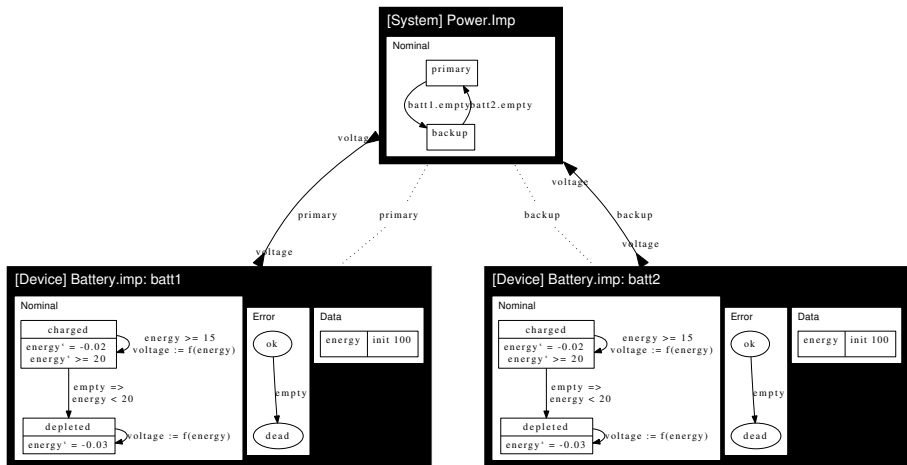
```
error model BatteryFailure
  features
    ok: initial state;
    dead: error state;
    batteryDied: out error propagation;
end BatteryFailure;

error model implementation BatteryFailure.Imp
  events
    fault: error event occurrence poisson 0.01;
  transitions
    ok -[fault]-> dead;
    dead -[batteryDied]-> dead;
end BatteryFailure.Imp;
```

Fault injection

In error state `dead`, `voltage := 0`

The Complete Power System Model



Nominal model + error model + fault injections = **extended model**

- **Modes** are pairs of nominal modes and error model states
 - **Starting mode** = (original starting mode, starting error state)
- **Event ports** $+=$ error propagations
- **Event port connections** $+=$ propagation port connections
- **Transition relation** $:=$ all possible **interleavings** and **interactions** between nominal and error model, taking failure effects into account
- Other elements (e.g., mode invariants) are unaffected

Nominal model + error model + fault injections = **extended model**

- **Modes** are pairs of nominal modes and error model states
 - **Starting mode** = (original starting mode, starting error state)
- **Event ports** $+=$ error propagations
- **Event port connections** $+=$ propagation port connections
- **Transition relation** $:=$ all possible **interleavings** and **interactions** between nominal and error model, taking failure effects into account
- Other elements (e.g., mode invariants) are unaffected

Probabilistic error transitions

As an error model has probabilistic transitions, our semantical model has to be equipped with such transitions.

This yields **interactive Markov chains** $:=$ LTS + Markov chains.

Battery Component

Nominal specification

```
device Battery
  features
    empty: out event port;
    voltage: out data port real default 6.0;

  end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous default 100.0;
  modes
    charged: activation mode while ...;
    depleted: mode while ...;
  transitions
    charged -[then voltage:=...] -> charged;
    charged -[empty when energy<=20.0] -> depleted;
    depleted -[then voltage:=...] -> depleted;

  end Battery.Imp;
```

Battery Component After Model Extension

Product construction for modes

```
device Battery
  features
    empty: out event port;
    voltage: out data port real default 6.0;

end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous default 100.0;
  modes
    charged#ok: activation mode while ...;
    depleted#ok, charged#dead, depleted#dead: mode while ...;
  transitions
    charged -[then voltage:=...]-> charged;
    charged -[empty when energy<=20.0]-> depleted;
    depleted -[then voltage:=...]-> depleted;

end Battery.Imp;
```


Battery Component After Model Extension

Integration of nominal transitions

```
device Battery
  features
    empty: out event port;
    voltage: out data port real default 6.0;

  end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous default 100.0;
  modes
    charged#ok: activation mode while ...;
    depleted#ok, charged#dead, depleted#dead: mode while ...;
  transitions
    charged#ok -[then voltage:=...]-> charged#ok;
    charged#ok -[empty when energy<=20.0]-> depleted#ok;
    depleted#ok -[then voltage:=...]-> depleted#ok;

  end Battery.Imp;
```

Battery Component After Model Extension

Fault injection

```
device Battery
  features
    empty: out event port;
    voltage: out data port real default 6.0;

end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous default 100.0;
  modes
    charged#ok: activation mode while ...;
    depleted#ok, charged#dead, depleted#dead: mode while ...;
  transitions
    charged#ok -[then voltage:=...]-> charged#ok;
    charged#ok -[empty when energy<=20.0]-> depleted#ok;
    depleted#ok -[then voltage:=...]-> depleted#ok;
    charged#ok -[prob 0.01 then voltage:=0]-> charged#dead;
    depleted#ok -[prob 0.01 then voltage:=0]-> depleted#dead;

end Battery.Imp;
```

Battery Component After Model Extension

Original transitions with fault effects

```
device Battery
  features
    empty: out event port;
    voltage: out data port real default 6.0;

end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous default 100.0;
  modes
    charged#ok: activation mode while ...;
    depleted#ok, charged#dead, depleted#dead: mode while ...;
  transitions
    charged#ok -[then voltage:=...]-> charged#ok;
    charged#ok -[empty when energy<=20.0]-> depleted#ok;
    depleted#ok -[then voltage:=...]-> depleted#ok;
    charged#ok -[prob 0.01 then voltage:=0]-> charged#dead;
    depleted#ok -[prob 0.01 then voltage:=0]-> depleted#dead;
    charged#dead -[then voltage:=0]-> charged#dead;
    charged#dead -[empty when energy<=20.0]-> depleted#dead;
    depleted#dead -[then voltage:=0]-> depleted#dead;

end Battery.Imp;
```

Battery Component After Model Extension

Addition of error propagations

```
device Battery
  features
    empty: out event port;
    voltage: out data port real default 6.0;
    batteryDied: out event port;
end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous default 100.0;
  modes
    charged#ok: activation mode while ...;
    depleted#ok, charged#dead, depleted#dead: mode while ...;
  transitions
    charged#ok -[then voltage:=...]-> charged#ok;
    charged#ok -[empty when energy<=20.0]-> depleted#ok;
    depleted#ok -[then voltage:=...]-> depleted#ok;
    charged#ok -[prob 0.01 then voltage:=0]-> charged#dead;
    depleted#ok -[prob 0.01 then voltage:=0]-> depleted#dead;
    charged#dead -[then voltage:=0]-> charged#dead;
    charged#dead -[empty when energy<=20.0]-> depleted#dead;
    depleted#dead -[then voltage:=0]-> depleted#dead;
    depleted#dead -[batteryDied]-> depleted#dead;
end Battery.Imp;
```

Specifying Observability

- Specification of **observables** for diagnosability analysis (later)
 - for outgoing data ports of type **bool**
- Example:

```
system PowerSystem
  features
    voltage: out data port real;
    alarm: out data port bool initially false observable;
  end PowerSystem;

system implementation PowerSystem.Imp
  subcomponents
    pow: system Power.Imp;
  connections
    data port pow.voltage -> voltage;
  modes
    normal: initial mode;
    critical: mode;
  transitions
    normal -[when voltage<4.5 then alarm:=true]-> critical;
    critical -[when voltage>5.5 then alarm:=false]-> normal;
  end PowerSystem.Imp;
```

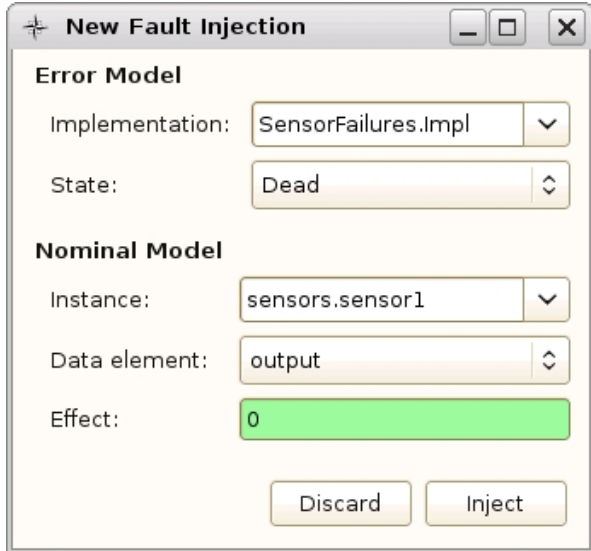
- 1 Representing Nominal System Behavior
- 2 Formal Semantics
- 3 Modeling Error Behavior
- 4 Tool Support**
- 5 References & Ongoing Activities

Loading Models

The screenshot displays the COMPASS Toolset interface, which is used for loading and managing models. The interface is divided into several sections:

- COMPASS Toolset** (Title Bar)
- File Edit View Activities Help** (Menu Bar)
- Model Properties Validation Correctness Performability Safety FDIR** (Tab Bar)
- Loaded Files**: A list of files loaded into the toolset, including `Desktop/COMPASS-toolset/tools/examples/demo_sae/sensorfilter.slim` and `Desktop/COMPASS-toolset/tools/examples/demo_sae/sensorfilterErr.slim`. Buttons for **Reload All**, **Remove**, and **Add** are present.
- FDIR Components**: A table showing the implementation and filename for FDIR components. The table has two columns: **Implementation** and **Filename**. The first row shows `__default__::Monitor.Impl` and `Desktop/COMPASS-toolset/tools/examples/demo_sae/se`.
- Root**: A table showing the implementation and filename for the root component. The table has two columns: **Implementation** and **Filename**. The first row shows `__default__::Acquisition.Impl` and `Desktop/COMPASS-toolset/tools/examples/demo_sae`.
- Fault Injections**: A table showing the use, error implementation, error state, and effect for fault injections. The table has four columns: **Use**, **Error Implementation**, **Error State**, and **Effect**. The first three rows are highlighted in orange. The first row shows `__default__::SensorFailures.Impl` with a **Dead** error state and the effect `sensors.sensor1.output := 15`. The second row shows `__default__::SensorFailures.Impl` with a **Dead** error state and the effect `sensors.sensor2.output := 15`. The third row shows `__default__::FilterFailures.Impl` with a **Dead** error state and the effect `filters.filter1.output := 0`. The fourth row shows `__default__::FilterFailures.Impl` with a **Dead** error state and the effect `filters.filter2.output := 0`. Buttons for **Remove**, **Clone**, and **Add** are present.
- Output Console**: A text area showing the output of the toolset, including the compilation of `sensorfilter.slim` and `sensorfilterErr.slim`, and the loading of fault injections. The output shows that 4 of 4 fault injections were loaded successfully.
- Compiler Logging Extended Model Metrics** (Bottom Bar)

Defining Fault Injections



A screenshot of a software dialog box titled "New Fault Injection". The dialog has a standard window header with a maximize button, a minimize button, and a close button. The content is organized into two sections: "Error Model" and "Nominal Model".

Error Model

- Implementation:** A text field containing "SensorFailures.Impl" with a dropdown arrow on the right.
- State:** A text field containing "Dead" with a double-headed arrow on the right.

Nominal Model

- Instance:** A text field containing "sensors.sensor1" with a dropdown arrow on the right.
- Data element:** A text field containing "output" with a double-headed arrow on the right.
- Effect:** A green rectangular field containing the number "0".

At the bottom of the dialog are two buttons: "Discard" and "Inject".

Simulating System Behavior

The screenshot displays the COMPASS Toolset interface for simulating system behavior. The main window is titled "COMPASS Toolset" and includes a menu bar (File, Edit, View, Activities, Help) and a tabbed interface with "Model", "Properties", "Validation", "Correctness", "Performability", "Safety", and "FDIR".

The "Properties" panel on the left shows a tree view of the model components, including "sensors" and "sensor1". The "Correctness" tab is active, showing a "Model extended by fault injections" checkbox and a "Guided by Transitions" dropdown. The "Simulation" section includes a "Run" button, a "Length" field set to 10, and "Restart" and "Jump" buttons.

The "Simulation" results are displayed in a table with columns for "Step1", "Step2", "Step3", and "Name". The table shows the progression of the simulation, including states like "OK", "Primary", "Drifted", and "Dead".

Step1	Step2	Step3	Name
			mode
			monitor.alarmF
			monitor.alarmS
			monitor.mode
			monitor.switchF
			monitor.switchS
2	2	30	monitor.valueF
1	1	15	monitor.valueS
			mybus.mode
Primary	Primary	Primary	sensors.mode
1	1	15	sensors.output
			sensors.sensor1.#die
			sensors.sensor1.#dieByDrift
			sensors.sensor1.#drift
			sensors.sensor1.error

The "Transitions" panel shows a list of events and their corresponding actions, such as "OK [resetEvent] > OK;" and "Drifted [resetEvent] > Drifted;".

- 1 Representing Nominal System Behavior
- 2 Formal Semantics
- 3 Modeling Error Behavior
- 4 Tool Support
- 5 References & Ongoing Activities

References

- Our AADL variant (Bozzano et. al, [MEMOCODE 2009](#))
- AADL formal semantics (Bozzano et. al, [Computer J. 2010](#))
- Relation to attribute grammars (Noll, [FACS 2011](#))

References

- Our AADL variant (Bozzano et. al, [MEMOCODE 2009](#))
- AADL formal semantics (Bozzano et. al, [Computer J. 2010](#))
- Relation to attribute grammars (Noll, [FACS 2011](#))

Ongoing activities

- Contribution to AADL standardization
- Modeling features for launcher systems
- Security aspects in AADL (D-MILS Project)