Codesign of Dependable Systems
A Component-Based Modelling Language

Marco Bozzano\(^2\) Alessandro Cimatti\(^2\) Marco Roveri\(^2\)
Joost-Pieter Katoen\(^1\) Viet Yen Nguyen\(^1\) Thomas Noll\(^1\)

\(^1\)Software Modelling and Verification Group
RWTH Aachen University, Germany

\(^2\)Embedded Systems Group
Fondazione Bruno Kessler, Italy

MEMOCODE 2009, 15 July 2009
Codesign of Dependable Systems
A Component-Based Modelling Language

1. Scope
2. SLIM Syntax
3. Formal Characterisation
4. Injecting Faults
5. COMPASS Toolset
6. Conclusions
Scope
## Verification of Fault-Tolerant Hardware/Software Systems

<table>
<thead>
<tr>
<th>Component</th>
<th>SMV, Promela, VHDL</th>
<th>NuSMV, SPIN, XilinxFmilx</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensor, protocol, controller</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>AADL, SysML, SLIM</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>satellite, robot, airplane</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Verification of Fault-Tolerant Hardware/Software Systems

<table>
<thead>
<tr>
<th>Component</th>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensor, protocol, controller</td>
<td>SMV, Promela, VHDL</td>
<td>NuSMV, SPIN, Xilinx</td>
</tr>
<tr>
<td>System</td>
<td>satellite, robot, airplane</td>
<td>AADL, SysML, SLIM</td>
</tr>
</tbody>
</table>

![Diagram showing integration of software and hardware]
## Verification of Fault-Tolerant Hardware/Software Systems

### Component
| Sensor, protocol, controller | SMV, Promela, VHDL | NuSMV, SPIN, Xilinx |

### System
| Satellite, robot, airplane | AADL, SysML, SLIM | ... |

![Diagram of Software and Hardware](image-url)
## Verification of Fault-Tolerant Hardware/Software Systems

### Component

| Sensor, protocol, controller | SMV, Promela, VHDL | NuSMV, SPIN, Xilinx |

### System

| Satellite, robot, airplane | AADL, SysML, SLIM | ... |

![Diagram](image)

- **FDIR**
- **Software**
- **Hardware**

2009, Viet Yen Nguyen
## Verification of Fault-Tolerant Hardware/Software Systems

<table>
<thead>
<tr>
<th>Component</th>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensor, protocol, controller</td>
<td>SMV, Promela, VHDL</td>
<td>NuSMV, SPIN, Xilinx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>satellite, robot, airplane</td>
<td>AADL, SysML, SLIM</td>
<td>COMPASS</td>
</tr>
</tbody>
</table>

![Diagram of FDIR](image)

2009, Viet Yen Nguyen
AADL: Industry Standard for Modelling Embedded Systems

- **1989** MetaH
- **1998** SAE AS-2C
- **2004** AADL 1.0
- **2006** Error Annex
- **2009** AADL 2.0

**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
### Integrated and Coherent Approach for Codesigning Systems

<table>
<thead>
<tr>
<th>SLIM</th>
<th>COMPASS Toolset</th>
</tr>
</thead>
<tbody>
<tr>
<td>AADL + Error Annex</td>
<td>NuSMV</td>
</tr>
<tr>
<td>Hardware/Software</td>
<td>FSAP</td>
</tr>
<tr>
<td>Error Propagation</td>
<td>RAT</td>
</tr>
<tr>
<td>Recovery Mechanisms</td>
<td>Sigref</td>
</tr>
<tr>
<td>Timing, Probability, Hybrid</td>
<td>MRMC</td>
</tr>
<tr>
<td>Formal Semantics</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analyses</th>
<th>Case Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbolic Model Checking</td>
<td>Satellite Thermal Regulation Manager</td>
</tr>
<tr>
<td>SAT-Solving</td>
<td>Satellite FDIR</td>
</tr>
<tr>
<td>Probabilistic Model Checking</td>
<td>European Train Control System Level 3</td>
</tr>
<tr>
<td>FTA</td>
<td></td>
</tr>
<tr>
<td>FMEA</td>
<td></td>
</tr>
</tbody>
</table>
# Integrated and Coherent Approach for Codesigning Systems

<table>
<thead>
<tr>
<th>SLIM</th>
<th>COMPASS Toolset</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ AADL + Error Annex</td>
<td>▶ NuSMV</td>
</tr>
<tr>
<td>▶ Hardware/Software</td>
<td>▶ FSAP</td>
</tr>
<tr>
<td>▶ Error Propagation</td>
<td>▶ RAT</td>
</tr>
<tr>
<td>▶ Recovery Mechanisms</td>
<td>▶ Sigref</td>
</tr>
<tr>
<td>▶ Timing, Probability, Hybrid</td>
<td>▶ MRMC</td>
</tr>
<tr>
<td>▶ Formal Semantics</td>
<td></td>
</tr>
</tbody>
</table>

### Analyses

- Symbolic Model Checking
- SAT-Solving
- Probabilistic Model Checking
- FTA
- FMEA

### Case Studies

- Satellite Thermal Regulation Manager
- Satellite FDIR
- European Train Control System Level 3

---

2009, Viet Yen Nguyen
# Integrated and Coherent Approach for Codesigning Systems

## SLIM
- AADL + Error Annex
- Hardware/Software
- Error Propagation
- Recovery Mechanisms
- Timing, Probability, Hybrid
- Formal Semantics

## COMPASS Toolset
- NuSMV
- FSAP
- RAT
- Sigref
- MRMC

## Analyses
- Symbolic Model Checking
- SAT-Solving
- Probabilistic Model Checking
- FTA
- FMEA

## Case Studies
- Satellite Thermal Regulation Manager
- Satellite FDIR
- European Train Control System Level 3
SLIM Syntax
We shall show:

- hybrid behaviour of the batteries,
- composition of the power system,
- formalisation to automata,
- semantics as transition systems,
- interweaving of errors.
SLIM: Modelling the Battery Component Type and Implementation

device type Battery

end Battery;

device implementation Battery.Imp

end Battery.Imp;
device type Battery
  features
    empty: out event port;
    voltage: out data port real initially 6.0;
end Battery;

device implementation Battery.Imp

end Battery.Imp;
SLIM: Modelling the Battery
Adding Modes Behaviour

device type Battery
features
    empty: out event port;
    voltage: out data port real initially 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;
device type Battery
  features
    empty: out event port;
    voltage: out data port real initially 6.0;
end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous initially 100.0;
  modes
    charged: activation mode
      while energy’=-0.02 and energy>=20.0;
    depleted: mode
      while energy’=-0.03;
  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;
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;
SLIM: Modelling 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;
SLIM: Modelling the Redundant Power System

Adding **Port Connections**

```plaintext
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;
```
Formal Characterisation
Formalising SLIM Components as Event-Data Automata

Definition (Event-Data Automaton)

An event-data automaton (EDA) is a tuple

$$\mathcal{A} = (M, m_0, X, v_0, \iota, E, \rightarrow)$$

with

- $M$ finite set of modes
- $m_0 \in M$ initial mode
- $X = IX \uplus OX \uplus LX$ finite set of input/output/local variables
- $V := \{ v \mid v : X \rightarrow \ldots \}$ valuations
- $v_0 \in V$ initial valuation
- $\iota : M \rightarrow (V \rightarrow \mathbb{B})$ mode invariants (where $\iota(m_0, v_0) = \text{true}$)
- $E = IE \uplus OE$ finite set of input/output events
- $\rightarrow \subseteq M \times E_{\tau} \times (V \rightarrow \mathbb{B}) \times (V \rightarrow V) \times M$

(mode) transition relation (where $E_{\tau} := E \cup \{\tau\}$)
Formalising SLIM Components as Event-Data Automata

- SLIM modes/invariants/transitions
  → EDA modes/invariants/transitions

Example (Battery)

- \( M = \{\text{charged, depleted}\}, m_0 = \text{charged} \)
Formalising SLIM Components as Event-Data Automata

- SLIM modes/invariants/transitions
  - EDA modes/invariants/transitions
- Incoming/outgoing data ports similar to input/output variables

Example (Battery)

- $M = \{\text{charged, depleted}\}$, $m_0 = \text{charged}$
- $IX = \emptyset$, $OX = \{\text{voltage}\}$
Formalising SLIM Components as Event-Data Automata

- SLIM modes/invariants/transitions
  \[\sim\] EDA modes/invariants/transitions
- Incoming/outgoing data ports \[\sim\] input/output variables
- Data subcomponents \[\sim\] local variables

**Example (Battery)**

- \(M = \{\text{charged, depleted}\}, m_0 = \text{charged}\)
- \(IX = \emptyset, OX = \{\text{voltage}\}\)
- \(LX = \{\text{energy}\}\)
Formalising SLIM Components as Event-Data Automata

- SLIM modes/invariants/transitions → EDA modes/invariants/transactions
- Incoming/outgoing data ports → input/output variables
- Data subcomponents → local variables
- SLIM incoming/outgoing event ports → EDA input/output events

**Example (Battery)**

- \( M = \{\text{charged, depleted}\}, m_0 = \text{charged} \)
- \( IX = \emptyset, OX = \{\text{voltage}\} \)
- \( LX = \{\text{energy}\} \)
- \( IE = \emptyset, OE = \{\text{empty}\} \)
LTS Semantics of Event-Data Automata

- **States**: $M \times V$
- **Transitions**: timed or internal or event-labeled
LTS Semantics of Event-Data Automata

- States := $M \times V$
- Transitions: timed or internal or event-labeled

**Example (Battery)**

$\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle$
LTS Semantics of Event-Data Automata

- States := $M \times V$
- Transitions: timed or internal or event-labeled

Example (Battery)

\[
\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle \\
\downarrow 30.0 \\
\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle
\]
LTS Semantics of Event-Data Automata

- States := $M \times V$
- Transitions: timed or internal or event-labeled

Example (Battery)

\[
\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle \\
\downarrow 30.0
\]

\[
\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle \\
\downarrow \tau \langle \text{voltage} := \ldots \rangle
\]

\[
\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle
\]
LTS Semantics of Event-Data Automata

- States := $M \times V$
- Transitions: timed or internal or event-labeled

Example (Battery)

\[
\begin{align*}
\langle & \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle \\
& \downarrow 30.0 \\
\langle & \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle \\
& \downarrow \tau \langle \text{voltage} := \ldots \rangle \\
\langle & \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle \\
& \downarrow 10.0 \\
\langle & \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.8 \rangle 
\end{align*}
\]
LTS Semantics of Event-Data Automata

- States := $M \times V$
- Transitions: timed or internal or event-labeled

**Example (Battery)**

<table>
<thead>
<tr>
<th>State</th>
<th>Charged Energy Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle$</td>
<td>$\downarrow 30.0$</td>
</tr>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle$</td>
<td>$\downarrow \tau \langle \text{voltage} := \ldots \rangle$</td>
</tr>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle$</td>
<td>$\downarrow 10.0$</td>
</tr>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.8 \rangle$</td>
<td>$\downarrow \tau \langle \text{voltage} := \ldots \rangle$</td>
</tr>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle$</td>
<td>$\downarrow$</td>
</tr>
</tbody>
</table>
LTS Semantics of Event-Data Automata

- States := $M \times V$
- Transitions: timed or internal or event-labeled

### Example (Battery)

<table>
<thead>
<tr>
<th>State</th>
<th>Time</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle$</td>
<td>$\Downarrow 30.0$</td>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle$</td>
</tr>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle$</td>
<td>$\Downarrow \tau \langle \text{voltage}:=... \rangle$</td>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle$</td>
</tr>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle$</td>
<td>$\Downarrow 10.0$</td>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.8 \rangle$</td>
</tr>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.8 \rangle$</td>
<td>$\Downarrow \tau \langle \text{voltage}:=... \rangle$</td>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle$</td>
</tr>
<tr>
<td>$\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle$</td>
<td>$\Downarrow \text{empty}$</td>
<td>$\langle \text{mode} = \text{depleted}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle$</td>
</tr>
</tbody>
</table>
LTS Semantics of Event-Data Automata

- States := $M \times V$
- Transitions: timed or internal or event-labeled

Example (Battery)

\[
\begin{align*}
\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle & \downarrow 30.0 \\
\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle & \downarrow \tau \langle \text{voltage}:=\ldots \rangle \\
\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle & \downarrow 10.0 \\
\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.8 \rangle & \downarrow \tau \langle \text{voltage}:=\ldots \rangle \\
\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle & \downarrow \text{empty} \\
\langle \text{mode} = \text{depleted}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle & \downarrow \ldots
\end{align*}
\]
Complete SLIM Specifications as Networks of EDAs

Dynamic reconfiguration
⇒ component activity and port connections mode dependent

Definition (Networks of Event-Data Automata)

A network of event-data automata (NEDA) is a tuple

\[ \mathcal{N} = \left( (\mathcal{A}_i)_{i \in [n]}, \alpha, EC, DC \right) \]

with \( n \geq 1, [n] := \{1, \ldots, n\} \), and

- each \( \mathcal{A}_i \) an EDA \( \mathcal{A}_i = (M_i, m^i_0, X_i, v^i_0, \iota_i, E_i, \rightarrow_i) \)
- \( M := \prod_{i=1}^{n} M_i \) set of global modes
- \( \alpha : M \rightarrow 2^{[n]} \) activation mapping
- \( EC : M \rightarrow (\{i.e \mid i \in [n], e \in E_i\})^2 \) event connection mapping
- \( DC : M \rightarrow (\{i.x \mid i \in [n], x \in X_i\})^2 \) data connection mapping
Complete SLIM Specifications as Networks of EDAs

- SLIM subcomponent in modes declarations
  - activation mapping:
    - root component always active
    - $c$ active and in mode $m$, $sc$ is subcomponent of $c$, $sc$ in modes $m$
      $\implies sc$ active

Example (Power System)

For $\underbrace{\text{Power/Battery1/Battery2}}_{1\ 2\ 3}$:

- $\alpha(\text{primary, charged, charged}) = \{1, 2\}$
- $\alpha(\text{primary, charged, depleted}) = \{1, 2\}$
  ...
- $\alpha(\text{backup, charged, depleted}) = \{1, 3\}$
  ...

2009, Viet Yen Nguyen
Complete SLIM Specifications as Networks of EDAs

- SLIM event/data connections $\sim EC/DC$ mappings: follow all end-to-end chains of port connections

Example (Power System)

For $\text{Power/Battery1/Battery2}$:

- $EC(\text{primary}, m_1, m_2) = \{(2.\text{empty}, 1.\text{batt1.empty})\}$
- $EC(\text{backup}, m_1, m_2) = \{(3.\text{empty}, 1.\text{batt2.empty})\}$
- $DC(\text{primary}, m_1, m_2) = \{(2.\text{voltage}, 1.\text{voltage})\}$
- $DC(\text{backup}, m_1, m_2) = \{(3.\text{voltage}, 1.\text{voltage})\}$
LTS Semantics of NEDAs

- **States**: \((M_1 \times V_1) \times \ldots \times (M_n \times V_n)\)

- **Transitions** determined by active EDAs:
  1. Perform local transitions:
     - timed local transition in all EDAs or
     - internal transition in EDA or
     - multiway event communication from EDA to \(\geq 1\) connected EDAs
  2. Initialize (re-)activated subcomponents
  3. Establish consistency w.r.t. \(DC\) (copy source \(\rightarrow\) target data port)

Example (Power system)

\[
\langle m = \text{primary}, v = 6.0 \rangle \;
\langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \;
\langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \;
\downarrow 40.0 \;
\langle m = \text{primary}, v = 6.0 \rangle \;
\langle m = \text{charged}, e = 20.0, v = 6.0 \rangle \;
\langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \;
\downarrow \tau \langle \text{voltage} := \ldots \rangle \;
\langle m = \text{primary}, v = 6.4 \rangle \;
\langle m = \text{charged}, e = 20.0, v = 6.4 \rangle \;
\langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \;
\downarrow \tau \langle \text{empty} \rangle \;
\langle m = \text{backup}, v = 6.0 \rangle \;
\langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \;
\langle m = \text{charged}, e = 20.0, v = 6.0 \rangle \;
\downarrow 40.0 \;
\langle m = \text{backup}, v = 6.0 \rangle \;
\langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \;
\langle m = \text{charged}, e = 20.0, v = 6.0 \rangle \;
\downarrow \cdots \]
LTS Semantics of NEDAs

- States := \((M_1 \times V_1) \times \ldots \times (M_n \times V_n)\)
- Transitions determined by active EDAs:
  1. Perform local transitions:
     - timed local transition in all EDAs or
     - internal transition in EDA or
     - multiway event communication from EDA to \(\geq 1\) connected EDAs
  2. Initialize (re-)activated subcomponents
  3. Establish consistency w.r.t. \(DC\) (copy source \(\rightarrow\) target data port)

Example (Power system)

\[
\langle m = \text{primary}, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]
LTS Semantics of NEDAs

- States := \((M_1 \times V_1) \times \ldots \times (M_n \times V_n)\)
- Transitions determined by active EDAs:
  1. Perform local transitions:
     - timed local transition in all EDAs or
     - internal transition in EDA or
     - multiway event communication from EDA to \(\geq 1\) connected EDAs
  2. Initialize (re-)activated subcomponents
  3. Establish consistency w.r.t. \(DC\) (copy source \(\rightarrow\) target data port)

Example (Power system)

\[
\langle m = \text{primary}, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow 40.0
\]

\[
\langle m = \text{primary}, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]
LTS Semantics of NEDAs

- States := \((M_1 \times V_1) \times \ldots \times (M_n \times V_n)\)
- Transitions determined by active EDAs:
  1. Perform local transitions:
     - timed local transition in all EDAs or
     - internal transition in EDA or
     - multiway event communication from EDA to \(\geq 1\) connected EDAs
  2. Initialize (re-)activated subcomponents
  3. Establish consistency w.r.t. \(DC\) (copy source \(\rightarrow\) target data port)

Example (Power system)

\[
\langle m = \text{primary}, v = 6.0 \rangle \downarrow 40.0 \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]

\[
\langle m = \text{primary}, v = 6.0 \rangle \downarrow \tau \langle \text{voltage} := \ldots \rangle \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle
\]

\[
\langle m = \text{primary}, v = 4.4 \rangle \downarrow \tau \langle \text{voltage} := \ldots \rangle \langle m = \text{charged}, e = 20.0, v = 4.4 \rangle
\]

\[
\langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \downarrow 40.0 \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle
\]

\[
\langle m = \text{charged}, e = 20.0, v = 4.4 \rangle \downarrow \tau \langle \text{voltage} := \ldots \rangle \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]
LTS Semantics of NEDAs

- States := \((M_1 \times V_1) \times \ldots \times (M_n \times V_n)\)
- Transitions determined by active EDAs:
  1. Perform local transitions:
     - timed local transition in all EDAs or
     - internal transition in EDA or
     - multiway event communication from EDA to \(\geq 1\) connected EDAs
  2. Initialize (re-)activated subcomponents
  3. Establish consistency w.r.t. DC (copy source \(\rightarrow\) target data port)

Example (Power system)

\[
\begin{align*}
\langle m = \text{primary}, v = 6.0 \rangle & | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
& \Downarrow 40.0 \\
\langle m = \text{primary}, v = 6.0 \rangle & | \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
& \Downarrow \tau\langle\text{voltage}:=\ldots\rangle \\
\langle m = \text{primary}, v = 4.4 \rangle & | \langle m = \text{charged}, e = 20.0, v = 4.4 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
& \Downarrow \tau\langle\text{empty}\rangle \\
\langle m = \text{backup}, v = 6.0 \rangle & | \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\end{align*}
\]
LTS Semantics of NEDAs

- States := \( (M_1 \times V_1) \times \ldots \times (M_n \times V_n) \)
- Transitions determined by active EDAs:
  1. Perform local transitions:
     - timed local transition in all EDAs or
     - internal transition in EDA or
     - multiway event communication from EDA to \( \geq 1 \) connected EDAs
  2. Initialize (re-)activated subcomponents
  3. Establish consistency w.r.t. \( DC \) (copy source → target data port)

Example (Power system)

\[
\begin{align*}
\langle m=\text{primary}, v=6.0 \rangle &\quad | \quad \langle m=\text{charged}, e=100.0, v=6.0 \rangle &\quad | \quad \langle m=\text{charged}, e=100.0, v=6.0 \rangle \\
\downarrow 40.0
\end{align*}
\]

\[
\begin{align*}
\langle m=\text{primary}, v=6.0 \rangle &\quad | \quad \langle m=\text{charged}, e=20.0, v=6.0 \rangle &\quad | \quad \langle m=\text{charged}, e=100.0, v=6.0 \rangle \\
\downarrow \tau \langle \text{voltage:=...} \rangle
\end{align*}
\]

\[
\begin{align*}
\langle m=\text{primary}, v=4.4 \rangle &\quad | \quad \langle m=\text{charged}, e=20.0, v=4.4 \rangle &\quad | \quad \langle m=\text{charged}, e=100.0, v=6.0 \rangle \\
\downarrow \tau \langle \text{empty} \rangle
\end{align*}
\]

\[
\begin{align*}
\langle m=\text{backup}, v=6.0 \rangle &\quad | \quad \langle m=\text{depleted}, e=20.0, v=4.4 \rangle &\quad | \quad \langle m=\text{charged}, e=100.0, v=6.0 \rangle \\
\downarrow 40.0
\end{align*}
\]

\[
\begin{align*}
\langle m=\text{backup}, v=6.0 \rangle &\quad | \quad \langle m=\text{depleted}, e=20.0, v=4.4 \rangle &\quad | \quad \langle m=\text{charged}, e=20.0, v=6.0 \rangle
\end{align*}
\]
LTS Semantics of NEDAs

- States := \((M_1 \times V_1) \times \ldots \times (M_n \times V_n)\)
- Transitions determined by active EDAs:
  1. Perform local transitions:
     - timed local transition in all EDAs or
     - internal transition in EDA or
     - multiway event communication from EDA to \(\geq 1\) connected EDAs
  2. Initialize (re-)activated subcomponents
  3. Establish consistency w.r.t. \(DC\) (copy source \(\rightarrow\) target data port)

Example (Power system)

\[
\begin{array}{c}
\langle \underline{m=\text{primary}}, v=6.0 \rangle | \langle \underline{m=\text{charged}}, e=100.0, v=6.0 \rangle | \langle m=\text{charged}, e=100.0, v=6.0 \rangle \\
\downarrow 40.0 \\
\langle \underline{m=\text{primary}}, v=6.0 \rangle | \langle m=\text{charged}, e=20.0, v=6.0 \rangle | \langle \underline{m=\text{charged}}, e=100.0, v=6.0 \rangle \\
\downarrow \tau\langle \text{voltage:=...}\rangle \\
\langle m=\text{primary}, v=4.4 \rangle | \langle m=\text{charged}, e=20.0, v=4.4 \rangle | \langle \underline{m=\text{charged}}, e=100.0, v=6.0 \rangle \\
\downarrow \tau\langle \text{empty}\rangle \\
\langle \underline{m=\text{backup}}, v=6.0 \rangle | \langle m=\text{depleted}, e=20.0, v=4.4 \rangle | \langle \underline{m=\text{charged}}, e=100.0, v=6.0 \rangle \\
\downarrow 40.0 \\
\langle \underline{m=\text{backup}}, v=6.0 \rangle | \langle m=\text{depleted}, e=20.0, v=4.4 \rangle | \langle \underline{m=\text{charged}}, e=20.0, v=6.0 \rangle \\
\downarrow \ldots
\end{array}
\]
Injecting Faults
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;
Specifying Faulty Behavior

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
Battery Component

Nominal Specification

device type Battery
features
  empty: out event port;
  voltage: out data port real initially 6.0;
end Battery;

device implementation Battery.Imp
subcomponents
  energy: data continuous initially 100.0;
modes
  charged: activation mode while ...;
  depleted: mode while ...;
transitions
  charged -> charged;
  charged -> depleted;
  depleted -> depleted;
end Battery.Imp;
Battery Component After Model Extension

Product Construction for Modes and Error States

device type Battery
features
  empty: out event port;
  voltage: out data port real initially 6.0;
end Battery;

device implementation Battery.Imp
subcomponents
  energy: data continuous initially 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
Integrate Nominal Transitions

device type Battery
features
  empty: out event port;
  voltage: out data port real initially 6.0;
end Battery;

device implementation Battery.Imp
subcomponents
  energy: data continuous initially 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**

**Add Fault Injections**

```plaintext
device type Battery
  features
    empty: out event port;
    voltage: out data port real initially 6.0;

end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous initially 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 -[then voltage:=0]-> charged#dead;
    depleted#ok -[then voltage:=0]-> depleted#dead;

end Battery.Imp;
```

2009, Viet Yen Nguyen
Battery Component After Model Extension
Nominal Transitions with Fault Effects

device type Battery
  features
    empty: out event port;
    voltage: out data port real initially 6.0;
  end Battery;

device implementation Battery.Imp
  subcomponents
    energy: data continuous initially 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 -[then voltage:= 0] -> charged#dead;
    depleted#ok -[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**

Add **Error Propagations**

device type Battery
features
  empty: out event port;
  voltage: out data port real initially 6.0;
  batteryDied: out event port;
end Battery;

device implementation Battery.Imp
subcomponents
  energy: data continuous initially 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;
  charged#ok -[then voltage:=...]-> depleted#ok;
  charged#ok -[then voltage:=0]-> charged#dead;
  depleted#ok -[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;
  charged#dead -[batteryDied]-> charged#dead;
end Battery.Imp;
COMPASS Toolset & Conclusions
Prototype is Up & Running as of April 2009

SAFETY ANALYSIS
- Dynamic Fault Trees
- FMEA Tables

CORRECTNESS VERIFICATION
- Property verification
- Simulation

DIAGNOSABILITY ANALYSIS
- FDIR effectiveness measures
- Synthesis of Observability Requirements

PERFORMABILITY ANALYSIS
- Performability measures
- Probabilistic fault trees

REQUIREMENTS VALIDATION
- Property Assurance
- Property Simulation

Slim2SMV
Instantiator
Slim Property
Table
Symbol
Sigref2MRMC
SMV2Sigref
NuSMV
MRMC
RAT
Slim Model
Property Pattern
Property Instantiator
Slim Property Instantiator
Model Extension

NuSMV

SMV2Slim
Trace Viewer
Fault Tree Viewer

SMV2Sigref
Sigref2MRMC

2009, Viet Yen Nguyen
Conclusions

Main

SLIM covers a greater part of AADL + Error Annex and has clearly defined formal semantics in terms of (Networks of) Event-Data Automata and Labelled Transition Systems.
Conclusions

Main

SLIM covers a greater part of AADL + Error Annex and has clearly defined formal semantics in terms of (Networks of) Event-Data Automata and Labelled Transition Systems.

Also

- SLIM is being brought to the AADL Standards Body.
- A toolset for formal verification of SLIM specifications exists. Approach me after the talk for a demonstration.
- SLIM is part of the COMPASS project which runs until 3/2010.
- And [http://compass.informatik.rwth-aachen.de/](http://compass.informatik.rwth-aachen.de/)