Safety, Dependability and Performance Analysis of Extended AADL Models\(^1\)

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

ROCKS Kick-Off Meeting
28 September 2009

\(^1\) Funded by ESA/ESTEC under Contract No. 21171/07/NL/JD
Safety, Dependability and Performance Analysis of Extended AADL Models

1. Scope
2. AADL Syntax
3. Formal Characterisation
4. Injecting Faults
5. COMPASS Toolset
6. Conclusions
How System Engineers Build Space Systems (in Europe)

Design Process

Mission Requirements & Constraints
- Objectives
- Environment
- Lifetime
- Payload
- Reliability
- Schedule
- Technology
- Budget

Study Requirements
- Products
- Study Level
- Planning
- Resources

Study Results
- S/C Design
- S/C Configuration
- Launcher
- Risk
- Cost
- Simulation
- Programmatics
- Options

Conceptual model of mission & spacecraft design process

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1989</td>
<td>MetaH</td>
</tr>
<tr>
<td>1998</td>
<td>SAE AS-2C</td>
</tr>
<tr>
<td>2004</td>
<td>AADL 1.0</td>
</tr>
<tr>
<td>2006</td>
<td>Error Annex</td>
</tr>
<tr>
<td>2009</td>
<td>AADL 2.0</td>
</tr>
</tbody>
</table>

#### 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

![Logos of sponsors](image)
# Integrated and Coherent Approach for Codesigning Systems

## Modelling Language
- 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
## Integrated and Coherent Approach for Codesigning Systems

<table>
<thead>
<tr>
<th>Modelling Language</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

## Modelling Language
- 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
AADL Syntax
We shall show:

- hybrid behaviour of the batteries,
- composition of the power system,
- formalisation to automata,
- semantics as transition systems,
- interweaving of errors.
device type Battery

device implementation Battery.Imp

end Battery.Imp;
AADL: Modelling the Battery
Component Type Defines the **Interface**

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;
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

deprecated: mode

transitions
charged -[]-> charged;
charged -[empty]-> depleted;
deprecated -[]-> depleted;
end Battery.Imp;
AADL: Modelling the Battery

Adding Hybrid Behaviour

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;
AADL: Modelling 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;
```
AADL: Modelling the Redundant Power System
Adding **Dynamic Reconfiguration**

```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);

    modes
        primary: initial mode;
        backup: mode;
    transitions
        primary -[batt1.empty]-> backup;
        backup -[batt2.empty]-> primary;
end Power.Imp;
```
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 AADL 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
- $\rightarrow \subseteq M \times E_\tau \times (V \rightarrow \mathbb{B}) \times (V \rightarrow V) \times M$
  - trigger
  - guard
  - effect

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

- AADL modes/invariants/transitions
  - EDA modes/invariants/transitions

Example (Battery)
- $M = \{\text{charged, depleted}\}$, $m_0 = \text{charged}$
Formalising AADL Components as Event-Data Automata

- AADL modes/invariants/transitions
  ~ EDA modes/invariants/transitions
- Incoming/outgoing data ports ~ input/output variables

Example (Battery)

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

- AADL modes/invariants/transactions
  \(\sim\) EDA modes/invariants/transactions
- 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 AADL Components as Event-Data Automata

- AADL modes/invariants/transitions \(\leadsto\) EDA modes/invariants/transitions
- Incoming/outgoing data ports \(\leadsto\) input/output variables
- Data subcomponents \(\leadsto\) local variables
- AADL incoming/outgoing event ports \(\leadsto\) EDA input/output events

Example (Battery)

- \(M = \{\text{charged, depleted}\}, \quad m_0 = \text{charged}\)
- \(IX = \emptyset, \quad OX = \{\text{voltage}\}\)
- \(LX = \{\text{energy}\}\)
- \(IE = \emptyset, \quad OE = \{\text{empty}\}\)
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)

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

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

Example (Battery)

\[
\begin{align*}
\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle & \ \downarrow 30.0 \\
\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle & \ \downarrow \tau\langle \text{voltage}:=... \rangle \\
\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle & \ \downarrow 10.0 \\
\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.8 \rangle & \ \downarrow \tau\langle \text{voltage}:=... \rangle \\
\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle & \ \downarrow \text{empty} \\
\langle \text{mode} = \text{depleted}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle
\end{align*}
\]
LTS Semantics of Event-Data Automata

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

Example (Battery)

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

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

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 AADL Specifications as Networks of EDAs

- AADL subcomponent in modes declarations
  \( \rightsquigarrow \) activation mapping:
  - root component always active
  - \( c \) active and in mode \( m \), \( sc \) is subcomponent of \( c \), \( sc \) in modes \( m \)
    \[ \implies sc \text{ active} \]

Example (Power System)

For \( \text{Power/Battery1/Battery2} \):

\[ 1 \quad 2 \quad 3 \]

- \( \alpha(\text{primary, charged, charged}) = \{1, 2\} \)
- \( \alpha(\text{primary, charged, depleted}) = \{1, 2\} \)
- \( \alpha(\text{backup, charged, depleted}) = \{1, 3\} \)
Complete AADL Specifications as Networks of EDAs

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

Example (Power System)

For $\text{Power/Battery}_1/\text{Battery}_2$:

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

\[
\begin{align*}
\langle m = \text{primary}, v = 6.0 \rangle & \quad \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow & \quad \langle m = \text{primary}, v = 4.4 \rangle \\
\langle m = \text{charged}, e = 100.0, v = 6.0 \rangle & \quad \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow & \quad \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\end{align*}
\]
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 $\rightarrow$ 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 → target data port)

Example (Power system)

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

\[
\langle m = \underline{primary}, v = 6.0 \rangle \mid \langle m = \underline{charged}, e = 100.0, v = 6.0 \rangle \mid \langle m = \underline{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow 40.0 \\
\langle m = \underline{primary}, v = 6.0 \rangle \mid \langle m = \underline{charged}, e = 20.0, v = 6.0 \rangle \mid \langle m = \underline{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow \tau\langle voltage := \ldots \rangle \\
\langle m = \underline{primary}, v = 4.4 \rangle \mid \langle m = \underline{charged}, e = 20.0, v = 4.4 \rangle \mid \langle m = \underline{charged}, e = 100.0, v = 6.0 \rangle
\]
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)**

\[
\langle m = \text{primary}, v = 6.0 \rangle \downarrow \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \downarrow \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\langle m = \text{primary}, v = 6.0 \rangle \downarrow \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle \downarrow \tau \langle \text{voltage:} = \ldots \rangle \\
\langle m = \text{primary}, v = 4.4 \rangle \downarrow \langle m = \text{charged}, e = 20.0, v = 4.4 \rangle \downarrow \tau \langle \text{empty} \rangle \\
\langle m = \text{backup}, v = 6.0 \rangle \downarrow \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \downarrow \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]
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)

\[
\langle m = \text{primary}, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]
\[\downarrow 40.0\]
\[
\langle m = \text{primary}, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]
\[\downarrow \tau \langle \text{voltage} = \ldots \rangle\]
\[
\langle m = \text{primary}, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 20.0, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]
\[\downarrow \tau \langle \text{empty} \rangle\]
\[
\langle m = \text{backup}, v = 6.0 \rangle \mid \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]
\[\downarrow 40.0\]
\[
\langle m = \text{backup}, v = 6.0 \rangle \mid \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \mid \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle
\]
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)

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

\[
\langle m = \underline{\text{primary}}, v = 6.0 \rangle | \langle m = \underline{\text{charged}}, e = 20.0, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \downarrow \tau \langle \text{voltage:=...} \rangle
\]

\[
\langle m = \underline{\text{primary}}, v = 4.4 \rangle | \langle m = \underline{\text{charged}}, e = 20.0, v = 4.4 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \downarrow \tau \langle \text{empty} \rangle
\]

\[
\langle m = \underline{\text{backup}}, v = 6.0 \rangle | \langle m = \underline{\text{depleted}}, e = 20.0, v = 4.4 \rangle | \langle m = \underline{\text{charged}}, e = 100.0, v = 6.0 \rangle \downarrow 40.0
\]

\[
\langle m = \underline{\text{backup}}, v = 6.0 \rangle | \langle m = \underline{\text{depleted}}, e = 20.0, v = 4.4 \rangle | \langle m = \underline{\text{charged}}, e = 20.0, v = 6.0 \rangle \downarrow \ldots
\]
Injecting Faults
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;
Specifying Faulty Behavior

```plaintext
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 -[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 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;
```
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;
    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;
    depleted#dead -[batteryDied]-> depleted#dead;
    charged#dead -[batteryDied]-> charged#dead;
end Battery.Imp;
COMPASS Toolset & Conclusions
First Version of Toolset is Up & Running as of April 2009

Analyses

- Requirements Consistency
- Simulation
- BDD + SAT-based Bounded Model Checking
- Hybrid Systems SMT-based Bounded Model Checking
- Probabilistic Model Checking
- (Prob.) Dynamic Fault Tree
- Failure Modes and Effects Tables
- Fault Tolerance
- Diagnosability
- Fault Detection, Isolation & Recovery
## Summary + What Hasn’t Been Discussed

<table>
<thead>
<tr>
<th>First Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal semantics of AADL and its Error Annex in terms of (Network of) Event-Data Automata.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Second Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyses for correctness, performance, dependability and RAMS aspects over these models with graphical tool support.</td>
</tr>
</tbody>
</table>
Summary + What Hasn’t Been Discussed

First Result
Formal semantics of AADL and its Error Annex in terms of (Network of) Event-Data Automata.

Second Result
Analyses for correctness, performance, dependability and RAMS aspects over these models with graphical tool support.

Also
- AADL Standards Body (plans to incorporate our extensions)
- Underlying formal models (TwinPlant, Markov Chains, etc.)
- Underlying algorithms (lumping, transient, SMT, etc.)
- Issues (numerical stability, bottlenecks, usability, etc.)
- Case studies (satellite, ETCS)
- Demo of toolset (GUI and console)
- Comparison to other tools and approaches (BIP, Arcade, etc.)
- Methodological integration (into ECSS framework)
References

By Marco Bozzano, Alessandro Cimatti, Joost-Pieter Katoen, Viet Yen Nguyen, Thomas Noll and Marco Roveri:


Slides of COMPASS 2009 workshop talks at ETAPS’09 are available:

compass.informatik.rwth-aachen.de