Space system requirements

- Must offer **service without interruption** for a very long time – typically years or decades
- **Faults** are costly and may severely damage reputations:
  - Ariane 5 crash in 1996 due to arithmetic overflow
  - Launch failure of recent Phobos-Grunt sample return mission
- “Five nines” (99.999 %) dependability **not** sufficient
Space system requirements

- Must offer **service without interruption** for a very long time – typically years or decades
- **Faults** are costly and may severely damage reputations:
  - Ariane 5 crash in 1996 due to arithmetic overflow
  - Launch failure of recent Phobos-Grunt sample return mission
- “Five nines” (99.999 %) dependability **not** sufficient

Challenges

- Rigorous **design support and analysis** techniques are called for
- Bugs must be found **as early as possible** in the design process
- Check **performance and reliability guarantees** whenever possible
- Effect of **Fault Diagnosis, Isolation and Recovery (FDIR)** measures must be quantifiable
Current Limitations

Limitations

- HW verified independently of SW with exaggerated mutual assumptions
- Safety & dependability analyses isolated from HW/SW models
- Multiple modeling formalisms for different system aspects (e.g. real-time, probabilistic, hybrid)
- No coherent approach to study effectiveness of FDIR
Possible Solutions

Solutions
Combination of
- HW, SW and their bindings +
- real-time, hybrid and probabilistic aspects +
- error models +
- non-nominal modes
in a single integrated model
1 Introduction
2 Project Overview
3 System Specifications
4 Formal Semantics of Nominal Specifications
5 Error Modeling
6 Conclusion
The COMPASS mission

Develop a model-based approach to system-software co-engineering while focusing on a coherent set of modeling and analysis techniques for evaluating system-level correctness, safety, dependability, and performance of on-board computer-based aerospace systems.
The COMPASS mission

Develop a model-based approach to system-software co-engineering while focusing on a coherent set of modeling and analysis techniques for evaluating system-level correctness, safety, dependability, and performance of on-board computer-based aerospace systems.

Derived objectives

1. Modeling formalism: SLIM
   (System-Level Integrated Modeling Language; variant of AADL)
2. Verification methodology based on state-of-the-art formal methods
3. Toolset supporting the analysis of SLIM models
4. Evaluation on industrial-size case studies from aerospace domain
COMPASS Project Partners

Consortium
- RWTH Aachen University
  Software Modeling and Verification Group
- Fondazione Bruno Kessler
  Embedded Systems Group
- Thales Alenia Space
  World-wide #1 in satellite systems
- Ellidiss

Funding & supervision
- European Space Agency
COMPASS Project Phases

1. Project kick-off
2. Language design
3. Software tool specification + software design document
4. Formal semantics
5. Prototype tool implementation
6. Prototype evaluation
7. Final tool implementation
8. Final tool evaluation
9. Project extension
10. New projects (NPI, CGM)

February 2008
October 2008
April 2009
December 2009
March 2010
until March 2011
until August 2012

Total budget: \( \approx 900 \text{ kEuro} \); \( \approx 10 \) programmers involved at peak times
Outline

1 Introduction
2 Project Overview
3 System Specifications
4 Formal Semantics of Nominal Specifications
5 Error Modeling
6 Conclusion
Redundant power system:

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

![Diagram of power system with two batteries and power switches](image)
Redundant power system:
- contains two batteries \textit{batt1}/\textit{batt2}
- used in \textit{primary}/\textit{backup} mode
- power switches from \textit{primary} to \textit{backup} (and back) when \textit{batt1} (\textit{batt2}) empty
- additionally provides \textit{voltage} information

We shall see:
- hybrid behavior of the \textit{batteries}
- composition of the \textit{power system}
- formalization by \textit{automata}
- \textit{semantics} as transition systems
- interweaving of \textit{errors}
Component type and implementation:

device type Battery

end Battery;

device implementation Battery.Imp

end Battery.Imp;
Modeling a Battery

Type defines the interface:

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

device implementation Battery.Imp

end Battery.Imp;
```
Adding modes behavior:

device type 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;
Adding **hybrid behavior**:

device type 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:

```plaintext
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;
```

Correctness, Safety and Fault Tolerance in Aerospace Systems
July 2, 2012 14/28
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;
```
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;
```
Outline

1 Introduction

2 Project Overview

3 System Specifications

4 Formal Semantics of Nominal Specifications

5 Error Modeling

6 Conclusion
Definition (Event-data automaton)

An event-data automaton (EDA) is a tuple

\[ \mathcal{A} = (M, X, V, \iota, E, \rightarrow) \]

with

- **M** finite set of modes
- **X** = \( IX \cup OX \cup LX \) finite set of input/output/local variables
- **V** := \( \{ v \mid v : X \rightarrow \ldots \} \) valuations
- **\iota** : \( M \rightarrow (V \rightarrow \mathbb{B}) \) mode invariants
- **E** = \( IE \cup OE \) finite set of input/output events
- \( \rightarrow \subseteq M \times (E \cup \{\tau\}) \times (V \rightarrow \mathbb{B}) \times (V \rightarrow V) \times M \) transition relation

trigger \quad guard \quad effect
An event-data automaton (EDA) is a tuple

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

with

- $M$ finite set of modes
- $X = I X \uplus O X \uplus L X$ finite set of input/output/local variables
- $V := \{ v \mid v : X \to \ldots \}$ valuations
- $\iota : M \to (V \to \mathbb{B})$ mode invariants
- $E = I E \uplus O E$ finite set of input/output events
- $\rightarrow \subseteq M \times (E \cup \{\tau\}) \times (V \to \mathbb{B}) \times (V \to V) \times M$ transition relation

where $\{\text{charged, depleted}\}$
Definition (Event-data automaton)

An **event-data automaton (EDA)** is a tuple

\[ \mathcal{A} = (M, X, V, \iota, E, \rightarrow) \]

with

- \( M \): finite set of modes
- \( X = IX \sqcup OX \sqcup LX \): finite set of input/output/local variables
- \( V := \{ v \mid v : X \rightarrow \ldots \} \): valuations
- \( \iota : M \rightarrow (V \rightarrow \mathbb{B}) \): mode invariants
- \( E = IE \sqcup OE \): finite set of input/output events
- \( \rightarrow \subseteq M \times (E \cup \{\tau\}) \times (V \rightarrow \mathbb{B}) \times (V \rightarrow V) \times M \): transition relation

**Correctness, Safety and Fault Tolerance in Aerospace Systems**

July 2, 2012 16/28
Definition (Event-data automaton)

An event-data automaton (EDA) is a tuple

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

with

- $M$ finite set of modes
- $X = IX \cup OX \cup LX$ finite set of input/output/local variables
- $V := \{v \mid v : X \to \ldots\}$ valuations
- $\iota : M \to (V \to \mathbb{B})$ mode invariants
- $E = IE \cup OE$ finite set of input/output events
- $\rightarrow \subseteq M \times (E \cup \{\tau\}) \times (V \to \mathbb{B}) \times (V \to V) \times M$ transition relation

\[\text{while energy)>=20.0}\]

\[\text{then voltage:=...}\]
Definition (Event-data automaton)

An event-data automaton (EDA) is a tuple

$$\mathfrak{A} = (M, X, V, \iota, E, \rightarrow)$$

with

- $M$ finite set of modes
- $X = IX \uplus OX \uplus LX$ finite set of input/output/local variables
- $V := \{v \mid v : X \rightarrow \ldots\}$ valuations
- $\iota : M \rightarrow (V \rightarrow \mathbb{B})$ mode invariants
- $E = IE \uplus OE$ finite set of input/output events
- $\rightarrow \subseteq M \times (E \cup \{\tau\}) \times (V \rightarrow \mathbb{B}) \times (V \rightarrow V) \times M$ transition relation

Correctness, Safety and Fault Tolerance in Aerospace Systems
July 2, 2012 16/28
An event-data automaton (EDA) is a tuple

\[ \mathcal{A} = (M, X, V, \iota, E, \rightarrow) \]

with

- \( M \) finite set of modes
- \( X = IX \cup OX \cup LX \) finite set of input/output/local variables
- \( V := \{ v \mid v : X \rightarrow \ldots \} \) valuations
- \( \iota : M \rightarrow (V \rightarrow \mathbb{B}) \) mode invariants
- \( E = IE \cup OE \) finite set of input/output events
- \( \rightarrow \subseteq M \times (E \cup \{\tau\}) \times (V \rightarrow \mathbb{B}) \times (V \rightarrow V) \times M \) transition relation

Correctness, Safety and Fault Tolerance in Aerospace Systems
July 2, 2012 16/28
Operational Semantics of EDAs

- States: $M \times V$
- Transitions: timed or internal or synchronized

Example (Battery)

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

Correctness, Safety and Fault Tolerance in Aerospace Systems
Operational Semantics of EDAs

- States: $M \times V$
- Transitions: timed or internal or synchronized

Example (Battery)

\[
\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle
\]
Operational Semantics of EDAs

- States: $M \times V$
- Transitions: timed or internal or synchronized

Example (Battery)

\[
\langle \text{mode} = \text{charged, energy} = 100.0, \text{voltage} = 6.0\rangle \\
\downarrow 30.0 \\
\langle \text{mode} = \text{charged, energy} = 40.0, \text{voltage} = 6.0\rangle
\]
Operational Semantics of EDAs

- States: $M \times V$
- Transitions: timed or internal or synchronized

Example (Battery)

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

$\downarrow 30.0$

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

$\downarrow \tau \langle \text{voltage} = \ldots \rangle$

$\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle$
Operational Semantics of EDAs

- States: $M \times V$
- Transitions: timed or internal or synchronized

Example (Battery)

\[
\langle \text{mode} = \text{charged}, \text{energy} = 100.0, \text{voltage} = 6.0 \rangle \\
\downarrow 30.0 \\
\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 6.0 \rangle \\
\downarrow \tau \langle \text{voltage} := \ldots \rangle \\
\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle \\
\downarrow 10.0 \\
\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.8 \rangle
\]
Operational Semantics of EDAs

- States: $M \times V$
- Transitions: timed or internal or synchronized

Example (Battery)

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

- States: $M \times V$
- Transitions: timed or internal or synchronized

Example (Battery)

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

\[ \downarrow 30.0 \]

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

\[ \downarrow \tau \langle \text{voltage} := \ldots \rangle \]

$\langle \text{mode} = \text{charged}, \text{energy} = 40.0, \text{voltage} = 4.8 \rangle$

\[ \downarrow 10.0 \]

$\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.8 \rangle$

\[ \downarrow \tau \langle \text{voltage} := \ldots \rangle \]

$\langle \text{mode} = \text{charged}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle$

\[ \downarrow \text{empty} \]

$\langle \text{mode} = \text{depleted}, \text{energy} = 20.0, \text{voltage} = 4.4 \rangle$
Operational Semantics of EDAs

- States: $M \times V$
- Transitions: timed or internal or synchronized

Example (Battery)

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

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 \( \mathcal{A}_i = (M_i, X_i, V_i, \nu_i, E_i, \rightarrow_i) \)
- \( M := \prod_{i=1}^{n} M_i \) set of global modes
- \( \alpha : M \to 2^{[n]} \) activation mapping
- \( EC : M \to (\{i.e \mid i \in [n], e \in E_i\})^2 \) event connection mapping
- \( DC : M \to (\{i.x \mid i \in [n], x \in X_i\})^2 \) data connection mapping
The Activation Mapping

Definition (Activation mapping)

\( \alpha \) determined by subcomponent declarations:

- Root component always active
- Component \( c \) active and in mode \( m \), subcomponent \( c' \) of \( c \) activated in \( m \)
  \( \Rightarrow \) \( c' \) active

Example (Power system)

For \( \text{Power}/\text{Battery1}/\text{Battery2} \) (\( m_1, m_2 \in \{\text{charged}, \text{depleted}\} \)):

\( \alpha(\text{primary}, m_1, m_2) = \{\text{Power}, \text{Battery1}\} \)

\( \alpha(\text{backup}, m_1, m_2) = \{\text{Power}, \text{Battery2}\} \)
Definition (Activation mapping)

$\alpha$ determined by subcomponent declarations:

- Root component always active
- Component $c$ active and in mode $m$,
  subcomponent $c'$ of $c$ activated in $m$
  $\Rightarrow c'$ active

Example (Power system)

For Power/Battery1/Battery2 ($m_1, m_2 \in \{\text{charged}, \text{depleted}\}$):

$$\alpha(\text{primary}, m_1, m_2) = \{\text{Power}, \text{Battery1}\}$$
$$\alpha(\text{backup}, m_1, m_2) = \{\text{Power}, \text{Battery2}\}$$
**Definition (Event/data connection mappings)**

EC/DC determined by following all end-to-end chains of port connections:

- **in-to-in**
- **out-to-in**
- **out-to-out**

**Example (Power system)**

For Power/Battery1/Battery2 (m1, m2 ∈ {charged, depleted}):  
- EC (primary, m1, m2) = {Battery1.empty → Power.empty}
- EC (backup, m1, m2) = {Battery2.empty → Power.empty}
- DC (primary, m1, m2) = {Battery1.voltage → Power.voltage}
- DC (backup, m1, m2) = {Battery2.voltage → Power.voltage}
The Connection Mappings

Definition (Event/data connection mappings)

**EC**/**DC** determined by following all end-to-end chains of port connections:

\[
c' \quad c \quad \text{root} \quad c' \quad \text{in-to-in} \\
\text{out-to-in} \quad \text{out-to-out}
\]

Example (Power system)

For **Power/Battery1/Battery2** \((m_1, m_2 \in \{\text{charged}, \text{depleted}\})\):  

\[
\begin{align*}
\text{EC(} \text{primary}, m_1, m_2 \text{)} &= \{ \text{Battery1.empty} \rightarrow \text{Power.empty} \} \\
\text{EC(} \text{backup}, m_1, m_2 \text{)} &= \{ \text{Battery2.empty} \rightarrow \text{Power.empty} \} \\
\text{DC(} \text{primary}, m_1, m_2 \text{)} &= \{ \text{Battery1.voltage} \rightarrow \text{Power.voltage} \} \\
\text{DC(} \text{backup}, m_1, m_2 \text{)} &= \{ \text{Battery2.voltage} \rightarrow \text{Power.voltage} \}
\end{align*}
\]
Operational Semantics of Networks of EDAs

- 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
     - multi-way 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)
\[
\langle m = \text{primary}, v = 6.0 \rangle \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\] \(\Downarrow\)
\[
\langle m = \text{primary}, v = 6.0 \rangle \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\] \(\Downarrow\)
\[
\langle m = \text{primary}, v = 4.4 \rangle \langle m = \text{charged}, e = 20.0, v = 4.4 \rangle \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\] \(\Downarrow\)
\[
\langle m = \text{backup}, v = 6.0 \rangle \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle
\] \(\Downarrow\)
\[
\langle m = \text{backup}, v = 6.0 \rangle \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle
\] \(\Downarrow\)
\[
\langle m = \text{backup}, v = 6.0 \rangle \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle
\]
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
   - multi-way 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)

\(\langle m = \text{primary}, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle\)
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
   - multi-way 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)

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

\[
\langle m = \underline{primary}, v = 6.0 \rangle | \langle m = \underline{charged}, e = 20.0, v = 6.0 \rangle | \langle m = \underline{charged}, e = 100.0, v = 6.0 \rangle
\]
Operational Semantics of Networks of EDAs

- 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
     - multi-way 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)

\[
\begin{align*}
\langle m = \text{primary}, v = 6.0 \rangle & \quad | \quad \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle & \quad | \quad \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow 40.0 & \\
\langle m = \text{primary}, v = 6.0 \rangle & \quad | \quad \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle & \quad | \quad \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow \tau\langle \text{voltage:...} \rangle & \\
\langle m = \text{primary}, v = 4.4 \rangle & \quad | \quad \langle m = \text{charged}, e = 20.0, v = 4.4 \rangle & \quad | \quad \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\end{align*}
\]
Operational Semantics of Networks of EDAs

- **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
     - multi-way 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)**

\[
\begin{align*}
\langle m = \text{primary}, v = 6.0 \rangle &| \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow & \quad 40.0 \\
\langle m = \text{primary}, v = 6.0 \rangle &| \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow & \quad \tau \langle \text{voltage:=...} \rangle \\
\langle m = \text{primary}, v = 4.4 \rangle &| \langle m = \text{charged}, e = 20.0, v = 4.4 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \\
\downarrow & \quad \tau \langle \text{empty} \rangle \\
\langle m = \text{backup}, v = 6.0 \rangle &| \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\end{align*}
\]
Operational Semantics of Networks of EDAs

- 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
     - multi-way 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)

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

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

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

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

\[
\langle m = \text{backup}, v = 6.0 \rangle \quad \langle m = \text{depleted}, e = 20.0, v = 4.4 \rangle \quad \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle
\]
Operational Semantics of Networks of EDAs

- 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
     - multi-way 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)

\[
\begin{align*}
\langle m = \text{primary}, v = 6.0 \rangle & \downarrow 40.0 \\
\langle m = \text{primary}, v = 6.0 \rangle & \downarrow \tau \langle \text{voltage} := \ldots \rangle \\
\langle m = \text{primary}, v = 4.4 \rangle & \downarrow \tau \langle \text{empty} \rangle \\
\langle m = \text{backup}, v = 6.0 \rangle & \downarrow 40.0 \\
\langle m = \text{backup}, v = 6.0 \rangle & \downarrow \ldots
\end{align*}
\]
Integrating Erroneous and Nominal Behavior

Nominal Model = SLIM components

Error Models

Fault Injections

Automatic Model Extension

Extended Model = nominal + error effects + degraded 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;
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;

Repair
reset events (not in example) can be sent from nominal to error model of same component to attempt to repair the occurred fault.
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 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
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.
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.
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;
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;
Integrate **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;
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;
Nominal 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;
Add 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;
Achievements

- Component-based modeling framework based on AADL
- Novelties: dynamic reconfiguration, hybridity, error modeling, ...
- Automated correctness, safety, and performability analysis
- Industrial evaluation by third-party company showed maturity

Trustworthy aerospace design = SLIM modeling + analysis
**Achievements**

- Component-based modeling framework based on AADL
- Novelties: dynamic reconfiguration, hybridity, error modeling, ...
- Automated correctness, safety, and performability analysis
- Industrial evaluation by third-party company showed maturity

```
Trustworthy aerospace design = SLIM modeling + analysis
```

**Ongoing/future activities**

- Coherent integration of semantic aspects
  (e.g., probabilistic time vs. non-determinism ⇒ extension of IMCs)
- Compositional verification (cf. Viet Yen’s talk)
- EU FP 7 Project Distributed MILS (Trustworthy ICT)
  ⇒ AADL + security aspects
Conclusion

Achievements

- Component-based modeling framework based on AADL
- Novelties: dynamic reconfiguration, hybridity, error modeling, ...
- Automated correctness, safety, and performability analysis
- Industrial evaluation by third-party company showed maturity

Trustworthy aerospace design = SLIM modeling + analysis

Ongoing/future activities

- Coherent integration of semantic aspects (e.g., probabilistic time vs. non-determinism ⇒ extension of IMCs)
- Compositional verification (cf. Viet Yen’s talk)
- EU FP 7 Project Distributed MILS (Trustworthy ICT) ⇒ AADL + security aspects

Further information & tool download

http://compass.informatik.rwth-aachen.de/