Correctness, Modelling and Performability of Aerospace Systems

Marco Bozzano\textsuperscript{2}  Alessandro Cimatti\textsuperscript{2}  Marco Roveri\textsuperscript{2}
Joost-Pieter Katoen\textsuperscript{1}  Viet Yen Nguyen\textsuperscript{1}  Thomas Noll\textsuperscript{1}

\textsuperscript{1}Software Modelling and Verification Group
RWTH Aachen University, Germany

\textsuperscript{2}Embedded Systems Group
Fondazione Bruno Kessler, Italy

Marktoberdorf Summerschool 2009

\textsuperscript{1}Funded by ESA/ESTEC under Contract No. 21171/07/NL/JD
## AADL: Industry Standard for Modelling Embedded Systems

<table>
<thead>
<tr>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Component-oriented</td>
</tr>
<tr>
<td>▶ HW/SW bindings</td>
</tr>
<tr>
<td>▶ Degraded operations</td>
</tr>
<tr>
<td>▶ Dynamic reconfiguration</td>
</tr>
<tr>
<td>▶ Error handling</td>
</tr>
<tr>
<td>▶ Fault injections</td>
</tr>
<tr>
<td>▶ Timing, probability, hybrid</td>
</tr>
<tr>
<td>▶ Formal semantics</td>
</tr>
</tbody>
</table>
AADL: Industry Standard for Modelling Embedded Systems

- Component-oriented
- HW/SW bindings
- Degraded operations
- Dynamic reconfiguration
- Error handling
- Fault injections
- Timing, probability, hybrid
- Formal semantics
AADL: Industry Standard for Modelling Embedded Systems

Features

▶ Component-oriented
▶ HW/SW bindings
▶ Degraded operations
▶ Dynamic reconfiguration
▶ Error handling
▶ Fault injections
▶ Timing, probability, hybrid
▶ Formal semantics

```plaintext
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: initial 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;
```

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

Features
- Component-oriented
- HW/SW bindings
- Degraded operations
- Dynamic reconfiguration
- Error handling
- Fault injections
- Timing, probability, hybrid
- Formal semantics

Power

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: initial 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;
Analyses over AADL Models

- AADL Model
- Fault Injections
- Requirement Patterns

- Compiler
  - SMV
  - NuSMV

- (...)
  - LTL/CTL
  - CSL

- State Space
  - Sigref
  - Markov Chain

- MRMC

Validation
- Consistency
- Simulation

Correctness
- Verification (discrete/hybrid)
- Simulation

Diagnosability
- FDIR
- Observability

Safety
- Dynamic Fault Trees
- FMEA tables

Performability
- Probabilistic DFT
- Performability measures
Analyses over AADL Models

AADL Model
- Fault Injections
  - Compiler
    - SMV
      - NuSMV
        - State Space
          - Sigref
            - Markov Chain
              - MRMC

Requirement Patterns
- (…)
  - LTL/CTL
    - CSL
  - Validation
    - Consistency
    - Simulation
  - Correctness
    - Verification (discrete/hybrid)
    - Simulation
  - Diagnosability
    - FDIR
    - Observability
  - Safety
    - Dynamic Fault Trees
    - FMEA tables
  - Performability
    - Probabilistic DFT
    - Performability measures
Analyses over AADL Models

- AADL Model
- Fault Injections
- Compiler
- Requirement Patterns
  - LTL/CTL
  - CSL
- NuSMV
- SMV

Validation
- Consistency
- Simulation

Correctness
- Verification (discrete/hybrid)
- Simulation

Diagnosability
- FDIR
- Observability

Safety
- Dynamic Fault Trees
- FMEA tables

Performability
- Probabilistic DFT
- Performability measures
Analyses over AADL Models

AADL Model → Fault Injections

Compiler

SMV → LTL/CTL

NuSMV

SignRef

Markov Chain

MRMC

Validation
- Consistency
- Simulation

Correctness
- Verification (discrete/hybrid)
- Simulation

Diagnosability
- FDIR
- Observability

Safety
- Dynamic Fault Trees
- FMEA tables

Performability
- Probabilistic DFT
- Performability measures
Analyses over AADL Models

- **AADL Model**
- **Fault Injections**
- **Requirement Patterns**

- **Compiler**
  - **SMV**
  - **NuSMV**

- **LTL/CTL**
  - **CSL**
  - **State Space**
  - **Sigref**
  - **Markov Chain**

- **MRMC**

**Validation**
- Consistency
- Simulation

**Correctness**
- Verification (discrete/hybrid)
- Simulation

**Diagnosability**
- FDIR
- Observability

**Safety**
- Dynamic Fault Trees
- FMEA tables

**Performability**
- Probabilistic DFT
- Performability measures

2009, Viet Yen Nguyen
Analyses over AADL Models

- AADL Model
- Fault Injections
- Requirement Patterns

Compiler

SMV

NuSMV

- LTL/CTL
- CSL

State Space

Sigref

Markov Chain

MRMC

Validation
• Consistency
• Simulation

Correctness
• Verification (discrete/hybrid)
• Simulation

Diagnosability
• FDIR
• Observability

Safety
• Dynamic Fault Trees
• FMEA tables

Performability
• Probabilistic DFT
• Performability measures
Analyses over AADL Models

- AADL Model
- Fault Injections
- Requirement Patterns

- Compiler
- SMV
- NuSMV

- (…)
- LTL/CTL
- CSL

- State Space
- Sigref
- Markov Chain
- MRMC

Validation
- Consistency
- Simulation

Correctness
- Verification (discrete/hybrid)
- Simulation

Diagnosability
- FDIR
- Observability

Safety
- Dynamic Fault Trees
- FMEA tables

Performability
- Probabilistic DFT
- Performability measures

2009, Viet Yen Nguyen
Comprehensive approach for formal analysis of (critical) embedded system architectures (like aerospace systems)
Summary + What hasn’t been discussed

Comprehensive approach for formal analysis of (critical) embedded system architectures
(like aerospace systems)

- Fault injections (through model extension)
Summary + What hasn’t been discussed

Comprehensive approach for formal analysis of (critical) embedded system architectures

(like aerospace systems)

- Fault injections (through model extension)
- Formal semantics (in terms of NEDA automata)
Summary + What hasn’t been discussed

Comprehensive approach for formal analysis of (critical) embedded system architectures

(like aerospace systems)

- Fault injections (through model extension)
- Formal semantics (in terms of NEDA automata)
- Underlying formal models (TwinPlants, Markov Chains, etc.)
Summary + What hasn’t been discussed

Comprehensive approach for formal analysis of (critical) embedded system architectures
(like aerospace systems)

- Fault injections (through model extension)
- Formal semantics (in terms of NEDA automata)
- Underlying formal models (TwinPlant, Markov Chains, etc.)
- Underlying algorithms (lumping, transient, SMT, etc.)
Summary + What hasn’t been discussed

Comprehensive approach for formal analysis of (critical) embedded system architectures
(like aerospace systems)

- Fault injections (through model extension)
- Formal semantics (in terms of NEDA automata)
- Underlying formal models (TwinPlant, Markov Chains, etc.)
- Underlying algorithms (lumping, transient, SMT, etc.)
- Issues (numerical stability, bottlenecks, usability, etc.)
Comprehensive approach for formal analysis of (critical) embedded system architectures (like aerospace systems)

- Fault injections (through model extension)
- Formal semantics (in terms of NEDA automata)
- Underlying formal models (TwinPlant, Markov Chains, etc.)
- Underlying algorithms (lumping, transient, SMT, etc.)
- Issues (numerical stability, bottlenecks, usability, etc.)
- Case studies (satellite, ETCS)
Comprehensive approach for formal analysis of (critical) embedded system architectures (like aerospace systems)

- Fault injections (through model extension)
- Formal semantics (in terms of NEDA automata)
- 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 prototype (GUI and console)
Comprehensive approach for formal analysis of (critical) embedded system architectures (like aerospace systems)

- Fault injections (through model extension)
- Formal semantics (in terms of NEDA automata)
- 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 prototype (GUI and console)
- Comparison to other tools and approaches (BIP, Arcade, etc.)
Comprehensive approach for formal analysis of (critical) embedded system architectures

(like aerospace systems)

- Fault injections (through model extension)
- Formal semantics (in terms of NEDA automata)
- 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 prototype (GUI and console)
- Comparison to other tools and approaches (BIP, Arcade, etc.)
- Methodological integration (into ECSS framework)
Summary + What hasn’t been discussed

Comprehensive approach for formal analysis of (critical) embedded system architectures

(like aerospace systems)

► Fault injections (through model extension)
► Formal semantics (in terms of NEDA automata)
► 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 prototype (GUI and console)
► Comparison to other tools and approaches (BIP, Arcade, etc.)
► Methodological integration (into ECSS framework)
► Anecdotes
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