MODELING AND ANALYSIS OF
FAULT-TOLERANT SYSTEMS USING
THE COMPASS TOOLSET

Viet Yen Nguyen

Software Modeling and Verification Group
RWTH Aachen University
System Software Engineering Section
European Space Agency

WebEx meeting on 16 Sept. 2011

joint work with Marco Bozzano, Alessandro Cimatti,
Joost-Pieter Katoen, Thomas Noll, Xavier Olive, Marco Roveri
and Yuri Yushtein
AGENDA

BIRTH OF COMPASS

MODELING

REQUIREMENTS SPECIFICATION

ANALYSIS & TOOLING

CASE STUDIES

EPILOGUE

LIVE DEMONSTRATION
BIRTH OF COMPASS
## Consortium

- **RWTH Aachen University**  
  Software Modeling and Verification Group

- **Fondazione Bruno Kessler**  
  Embedded Systems Group

- **Thales Alenia Space**  
  World-wide #1 in satellite systems

- **European Space Agency**  
  System Software Engineering Section
ExoMars Rover
- 4 to 21 min. for radio latency to earth.
- XXX Martian days autonomous survival.

Autonomous Transfer Vehicle
- human-rated: multiple-failure tolerant design.
- autonomous docking with four overrides: HOLD, RETREAT, ESCAPE, ABORT
OVERVIEW: CURRENT LIMITATIONS VERSUS COMPASS SOLUTIONS

Limitation

Hardware

Software

Safety/Dependability

System

SimuLink

UML

RtUML

SysML

Relex

Stochastic

Timed

Petri Net

PPT

Shapes

RtUML

Limitation

HW verified independently of SW with exaggerated mutual assumptions.

Safety & dependability analyses are isolated from HW/SW models.

Multiple modeling formalisms for different system aspects (e.g. real-time, probabilistic, hybrid).

Non-nominal operational modes are overly abstracted to fit various models.
OVERVIEW: CURRENT LIMITATIONS VERSUS COMPASS SOLUTIONS

Limitation

HW verified independently of SW with exaggerated mutual assumptions.
OVERVIEW: CURRENT LIMITATIONS VERSUS COMPASS SOLUTIONS

Limitation

HW verified independently of SW with exaggerated mutual assumptions.

Safety & dependability analyses are isolated from HW/SW models.
OVERVIEW: CURRENT LIMITATIONS VERSUS COMPASS SOLUTIONS

Limitation

- HW verified independently of SW with exaggerated mutual assumptions.
- Safety & dependability analyses are isolated from HW/SW models.
- Multiple modeling formalisms for different system aspects (e.g. real-time, probabilistic, hybrid).
OVERVIEW: CURRENT LIMITATIONS VERSUS COMPASS SOLUTIONS

Limitation

- HW verified independently of SW with exaggerated mutual assumptions.
- Safety & dependability analyses are isolated from HW/SW models.
- Multiple modeling formalisms for different system aspects (e.g. real-time, probabilistic, hybrid).
- Non-nominal operational modes are overly abstracted to fit various models.
Solution

Combine HW, SW and their bindings + ...

error models ... +

real-time, probabilistic and hybrid aspects ... +

non-nominal modes in a single integrated model.
ALL-IN-ONE FOR RAISING THE BAR OF SYSTEM-LEVEL ENGINEERING

SLIM = AADL + Error + Behavior + Formal Semantics

COMPASS Project

Specification Patterns for Requirements

Tooling & Analysis Model Checking

Industrial Case Studies

2011, Viet Yen Nguyen
FOR PUBLIC USE
MODELING
AADL: INDUSTRY STANDARD FOR MODELING EMBEDDED SYSTEMS

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

1989 MetaH

1998 SAE AS-2C

2004 AADL 1.0

2006 Error Annex

2009 AADL 2.0
SLIM: AADL-BASED MODELING LANGUAGE FOR FORMAL ANALYSIS

System-Level Integrated Modeling:

- Covers major part of AADL 1.0.
- Functional, real-time and hybrid behavior.
- Poisson-distributed errors.
- Mathematical characterization of behavior.
- Minor adaptations needed for formalization.
We shall show:
- hybrid behaviour of the batteries,
- composition of the power system,
- semantics as transition systems,
- interweaving of errors.
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;
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;
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;
Network of Event-Data Automata ≈ communicating state machines
+ linear differential equations on states
+ clocks and real-timed transitions
+ Markovian transitions (probabilistic)

Formalization by mapping:
- 1 SLIM component ⇛ Event-Data Automaton
- SLIM hierarchy ⇛ Network of Event-Data Automaton
SLIM: TRANSITIONS BY A NETWORK OF EVENT-DATA AUTOMATA

- **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)

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

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

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

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

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

\[\downarrow \ldots\]
SLIM: TRANSITIONS BY A NETWORK OF EVENT-DATA AUTOMATA

• 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)

\[\langle m = \text{primary}, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \mid \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle\]
SLIM: TRANSITIONS BY A NETWORK OF EVENT-DATA AUTOMATA

- 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)

\[
\langle m = \text{primary}, v = 6.0 \rangle | \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle \downarrow 40.0 \\
\langle m = \text{primary}, v = 6.0 \rangle \hspace{1cm} \langle m = \text{charged}, e = 20.0, v = 6.0 \rangle \hspace{1cm} \langle m = \text{charged}, e = 100.0, v = 6.0 \rangle
\]
SLIM: TRANSITIONS BY A NETWORK OF EVENT-DATA AUTOMATA

- 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)

\[
\langle m_{\text{primary}}, v = 6.0 \rangle | \langle m_{\text{charged}}, e = 100.0, v = 6.0 \rangle | \langle m_{\text{charged}}, e = 100.0, v = 6.0 \rangle \downarrow 40.0 \\
\langle m_{\text{primary}}, v = 6.0 \rangle | \langle m_{\text{charged}}, e = 20.0, v = 6.0 \rangle | \langle m_{\text{charged}}, e = 100.0, v = 6.0 \rangle \downarrow \tau \langle \text{voltage} := \ldots \rangle \\
\langle m_{\text{primary}}, v = 4.4 \rangle | \langle m_{\text{charged}}, e = 20.0, v = 4.4 \rangle | \langle m_{\text{charged}}, e = 100.0, v = 6.0 \rangle
\]

2011, Viet Yen Nguyen
FOR PUBLIC USE
MODELING - 16/50
SLIM: TRANSITIONS BY A NETWORK OF EVENT-DATA AUTOMATA

- 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)

\[
\langle m=\text{primary}, v=6.0 \rangle \mid \langle m=\text{charged}, e=100.0, v=6.0 \rangle \mid \langle m=\text{charged}, e=100.0, v=6.0 \rangle \\
\downarrow 40.0
\]
\[
\langle m=\text{primary}, v=6.0 \rangle \mid \langle m=\text{charged}, e=20.0, v=6.0 \rangle \mid \langle m=\text{charged}, e=100.0, v=6.0 \rangle \\
\downarrow \tau \langle \text{voltage:=...} \rangle
\]
\[
\langle m=\text{primary}, v=4.4 \rangle \mid \langle m=\text{charged}, e=20.0, v=4.4 \rangle \mid \langle m=\text{charged}, e=100.0, v=6.0 \rangle \\
\downarrow \tau \langle \text{empty} \rangle
\]
\[
\langle m=\text{backup}, v=6.0 \rangle \mid \langle m=\text{depleted}, e=20.0, v=4.4 \rangle \mid \langle m=\text{charged}, e=100.0, v=6.0 \rangle
\]
SLIM: TRANSITIONS BY A NETWORK OF EVENT-DATA AUTOMATA

- 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)

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

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

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

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

\[
\langle m=\text{backup}, v=6.0 \rangle \mid \langle m=\text{depleted}, e=20.0, v=4.4 \rangle \mid \langle m=\text{charged}, e=20.0, v=6.0 \rangle
\]
SLIM: TRANSITIONS BY A NETWORK OF EVENT-DATA AUTOMATA

- 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)

\[
\begin{align*}
\langle m=\text{primary}, v=6.0\rangle &\mid \langle m=\text{charged}, e=100.0, v=6.0\rangle \mid \langle m=\text{charged}, e=100.0, v=6.0\rangle \\
&\downarrow 40.0 \\
\langle m=\text{primary}, v=6.0\rangle &\mid \langle m=\text{charged}, e=20.0, v=6.0\rangle \mid \langle m=\text{charged}, e=100.0, v=6.0\rangle \\
&\downarrow \tau\langle \text{voltage=...}\rangle \\
\langle m=\text{primary}, v=4.4\rangle &\mid \langle m=\text{charged}, e=20.0, v=4.4\rangle \mid \langle m=\text{charged}, e=100.0, v=6.0\rangle \\
&\downarrow \tau\langle \text{empty}\rangle \\
\langle m=\text{backup}, v=6.0\rangle &\mid \langle m=\text{depleted}, e=20.0, v=4.4\rangle \mid \langle m=\text{charged}, e=100.0, v=6.0\rangle \\
&\downarrow 40.0 \\
\langle m=\text{backup}, v=6.0\rangle &\mid \langle m=\text{depleted}, e=20.0, v=4.4\rangle \mid \langle m=\text{charged}, e=20.0, v=6.0\rangle \\
&\downarrow \ldots
\end{align*}
\]
SLIM: INCLUDING ERRONEOUS TO NOMINAL BEHAVIOR

Extended Model = nominal + error effects + degraded behavior

Nominal Model = SLIM components
Error Models
Fault Injections
Automatic Model Extension

Extended Model = nominal + error effects + degraded behavior

2011, Viet Yen Nguyen FOR PUBLIC USE MODELING - 17/50
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;

Fault Injection

In error state dead, voltage:=0
SLIM: 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;
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;
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;
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;
SLIM: 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;
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;
REQUIREMENTS SPECIFICATION
Patterns

- The system shall have a behavior where with probability higher than \(0.98\) it is the case that \(\text{voltage} \geq 80\) holds continuously within time bound \([0,10]\).
- The system shall have a behavior where \(x \leq \text{voltage} \leq y\) globally holds.

(by automatic transformation)

Logic

- \(P_{>0.98}[\square^{[0,10]}(\text{voltage} \geq 80)]\) (Continuous Stochastic Logic)
- \(\square(x \leq \text{voltage} \leq y)\) (Linear Temporal Logic)
ANALYSIS & TOOLING
COMPASS TOOLSET 2.2 FEATURES WITH RELATION TO INPUTS

Nominal Model
Error Model
Fault Injections
Req's (patterns)

System/ HW/SW
Safety/ Dependability

Extended Model

State Space (LTS/SMT)
Markov Chain
Req's (logic)

Correctness
- Model checking (hybrid/discrete)
- Simulation
- Deadlock checking

FDIR Effectiveness
- Fault detection
- Fault isolation
- Fault recovery
- Diagnosability

Safety/Depend.
- Dynamic FTA
- Dynamic FMEA
- Fault Tolerance

Performability
- Performance evaluation
- Probabilistic risk assessment of fault tree.

Validation
- Consistency check
- Assertion check
- Simulation

Input
Intermediate Artefact
Analyses/ Output

Formal Canonical Models
VALIDATION: ARE WE BUILDING THE RIGHT SYSTEM?

Find specification inconsistencies in the Requirements Specification before detailed design starts.

Analyses

- Property consistency: *do the requirements contradict each other?*
- Property assurance: *are the requirements too strict/permissive?*
CORRECTNESS: ARE WE BUILDING THE SYSTEM RIGHT?

Verification of the SLIM model (with optionally failures) against requirements.

Analyses

- Model simulation
- Model checking (also known as Exhaustive Testing)
- Deadlock checking
SAFETY & DEPENDABILITY: MAPPING FAILURE MODES ↔ FAULTS?

Generation of safety & dependability artefacts from the SLIM model.

Analyses

- (Probabilistic) dynamic Fault Tree Analysis (FTA)
- (Dynamic) Failure modes and effects analysis (FMEA)
- Fault tolerance evaluation
SAFETY & DEPENDABILITY: MAPPING FAILURE MODES ↔ FAULTS?

Generation of safety & dependability artefacts from the SLIM model.

Analyses

- (Probabilistic) dynamic Fault Tree Analysis (FTA)
- (Dynamic) Failure modes and effects analysis (FMEA)
- Fault tolerance evaluation
FDIR: WILL THE SYSTEM RECOVER FROM A FAULT?

Analysis of the FDIR part in the SLIM model containing the observable tags.

Analyses

- Fault detection analysis: *how does the system detect faults?*
- Fault isolation analysis: *can the system distinguish faults?*
- Fault recovery analysis: *does the system recover after a fault?*
- Diagnosability: *are there sufficient sensors to detect faults?*
Analysis of the FDIR part in the SLIM model containing the observable tags.

Analyses

- Fault detection analysis: *how does the system detect faults?*
- Fault isolation analysis: *can the system distinguish faults?*
- Fault recovery analysis: *does the system recover after a fault?*
- Diagnosability: *are there sufficient sensors to detect faults?*
FDIR: WILL THE SYSTEM RECOVER FROM A FAULT?

Analysis of the FDIR part in the SLIM model containing the observable tags.

Analyses

- Fault detection analysis: *how does the system detect faults?*
- Fault isolation analysis: *can the system distinguish faults?*
- Fault recovery analysis: *does the system recover after a fault?*
- Diagnosability: *are there sufficient sensors to detect faults?*
FDIR: WILL THE SYSTEM RECOVER FROM A FAULT?

Analysis of the FDIR part in the SLIM model containing the observable tags.

Analyses

- Fault detection analysis: *how does the system detect faults?*
- Fault isolation analysis: *can the system distinguish faults?*
- Fault recovery analysis: *does the system recover after a fault?*
- Diagnosability: *are there sufficient sensors to detect faults?*
Markovian performance evaluation of the SLIM model for availability and sensitivity.

Analyses

- Performability: *how long and with which probability can the system sustain degraded operations?*
CASE STUDIES
CASE 1: SATELLITE REGULATION SYSTEM

Challenges:

- Hardware (sensors, heaters) and software (control) co-engineering
- Hybrid behavior (temperatures)
- Dynamic reconfiguration (redundancy)
- State-space explosion
# CASE 2: SATELLITE FDIR SYSTEM

## Goal
Assess effectiveness of FDIR measures

## Model components:
- satellite mode management during transfer-to-orbit phase
- AOCS (Attitude and Orbit Control System) mode management
- abstraction of AOCS equipment (sensors, gyroscope, ...)
- FDIR action sequence

## Analysis problems:
- identification of failures leading to a given FDIR level
- identification of failures entailing a system reconfiguration
- impact of reconfiguration on satellite and AOCS mode
CASE 3: SATELLITE PLATFORM

Payload is mission-specific equipment, e.g.:
- telecom transponders,
- navigation signals,
- earth observation telemetry (weather, radiation, salinity).

Platform keeps the satellite orbiting in space, consists of:
- attitude & orbital control,
- power distribution,
- data handling,
- communications,
- thermal regulation,
- propulsion.
CASE 3: SATELLITE PLATFORM

Verification & Validation Objectives

- Ensure nominal and degraded conditions are handled correctly by the fault management system.
- Ensure performance and risks are within specified limits.

Model Characteristics

✓ Functional
✓ Probabilistic
✓ Continuous Real-Time
✓ Hybrid

Components: 99
Modes: 217
Faults: 21
Recoveries: 9

State Space of Nominal Behaviour: 48,421,100 states

Requirement Metrics

- Functional properties: 32
- Probabilistic: 2
CASE 3: SATELLITE PLATFORM


<table>
<thead>
<tr>
<th>Analysis</th>
<th>Time (min:sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadlock checking</td>
<td>3:41</td>
</tr>
<tr>
<td>Model checking “recovers from sensor failure”</td>
<td>10:34</td>
</tr>
<tr>
<td>Fault tree analysis “sensor reading incorrect”</td>
<td>6:35</td>
</tr>
<tr>
<td>Fault tree evaluation “sensor reading incorrect”</td>
<td>0:02</td>
</tr>
<tr>
<td>Fault tolerance evaluation</td>
<td>0:06</td>
</tr>
<tr>
<td>Dynamic fault tree analysis “sensor reading incorrect”</td>
<td>10:51</td>
</tr>
<tr>
<td>Dynamic fault tree evaluation “sensor reading incorrect”</td>
<td>0:03</td>
</tr>
<tr>
<td>Fault detection “sensor failed”</td>
<td>22:47</td>
</tr>
<tr>
<td>Fault isolation “sensor failed”</td>
<td>4:59</td>
</tr>
<tr>
<td>Fault recovery “sensor failed”</td>
<td>10:24</td>
</tr>
<tr>
<td>FMEA “sensor failed”</td>
<td>18:10</td>
</tr>
<tr>
<td>Performability</td>
<td>&gt; 552:00(^1)</td>
</tr>
<tr>
<td>Diagnosability</td>
<td>&gt; 3823:00(^2)</td>
</tr>
</tbody>
</table>

\(^1\)ran out of memory  
\(^2\)aborted after 63 hours
GENERAL: MORE COMPLEX MODELS MEANS MORE TIME TO ANALYZE

<table>
<thead>
<tr>
<th>Verification Time (Seconds)</th>
<th>Degree of Redundancy</th>
<th>NuSMV</th>
<th>SigRef</th>
<th>MRMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7500</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15000</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22500</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30000</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2011, Viet Yen Nguyen FOR PUBLIC USE CASE STUDIES - 35/50
**GENERAL: EVALUATION OUTCOME**

+ **Abstraction level** of models is appropriate
  - mode transition systems
  - *not* source code

+ **Support of incremental approach** to system design
  - start with abstract functional representation
  - refinement without breaking structure of model
  - separation of component interface and implementation

+ Analyses give **valuable feedback** to system designer

− **Missing (automatic) link** between AADL and engineering models (UML, Matlab Simulink)
  - integration into engineering tool suite

− **Toolset performance** on timed and hybrid systems need further study
ALL-IN-ONE FOR RAISING THE BAR OF SYSTEM-LEVEL ENGINEERING

SLIM = AADL + Error + Behavior + Formal Semantics

Specification Patterns for Requirements

Tooling & Analysis Model Checking

COMPASS Project

Industrial Case Studies
COMPass Project: 2008-2011

1. Project Kick-Off
February 2008
2. SLIM Language Design
3. Software Tool Specification
4. Software Design Document
5. Formal SLIM Semantics
October 2008
6. Prototype Tool Implementation
April 2009
7. Prototype Evaluation
8. Final Tool Implementation
December 2009
9. Final Tool Evaluation
March 2010
10. CCN Kick-Off
October 2010
11. CCN End
March 2011
• SLIM slicing  
  Smaller models by semantics-preservation reduction using requirements as driver.  
  March 2010

• Compositional model checking  
  Faster analysis of models by exploiting the hierarchical and component-oriented structure.  
  September 2010

• SLIM graphical modeler  
  User-friendly and graphical drag-&-drop construction of SLIM models.  
  January 2011

• Impact analysis  
  Understand scope of cause-&-effect of (hardware/software) functions for classification of criticalities/priorities.  
  April 2011

• Contribution to AADL standard  
  Leverage COMPASS-experience of building theoretically solid analyses & tools.  
  October 2011
COMPASS Toolset binary & source availability is restricted to legal persons of ESA-member states. For non-ESA member states, a technology transfer procedure needs to be initiated. Contact me for details.

- Management summary  
  (Katoen & Noll, 11th Public Service Review: EU Science & Tech. 2011)

- Technical overview  
  (Yushtein et. al, IEEE Space Mission Challenges 2011)

- Technical realization  
  (Bozzano et. al, Oxford Computer Journal 2011)

- SLIM formal semantics  
  (Bozzano et. al, IEEE MEMOCODE 2009)

- SLIM model checker  
  (Bozzano et. al, CAV 2010)

- Website  
  (http://compass.informatik.rwth-aachen.de/)
• What kind of models do you want to capture?  
  What characteristics? (probabilistic, real-timed, . . . )
• Kickstarting a formal modeling & analysis pilot?
• How does feature/analysis . . . work?
• How does COMPASS fit in the traditional engineering/development process?
• . . . ?
LIVE DEMONSTRATION
REDUNDANT SENSOR-FILTER ACQUISITION SYSTEM WITH FDIR
system Sensors
  features
    output: out data port int default 1;
    switch: in event port;
end Sensors;

system implementation Sensors.Impl
  subcomponents
    sensor1: device Sensor in modes (Primary);
    sensor2: device Sensor in modes (Backup);
  connections
    data port sensor1.output -> output in modes (Primary);
    data port sensor2.output -> output in modes (Backup);
  modes
    Primary: activation mode; Backup: mode;
  transitions
    Primary -[switch]-> Backup;
end Sensors.Impl;
MODELING A SENSOR IN SLIM

device Sensor
    features
        output: out data port int default 1;
    end Sensor;

device implementation Sensor.Impl
    modes
        Cycle: activation mode;
    transitions
        Cycle -[when output < 5 then output := output + 1]-> Cycle;
    end Sensor.Impl;
ERROR MODELS OF SENSOR & FILTER FAILURES

SensorFailures

- OK
- Drifted
- Dead

Drifted -> OK with probability 0.083
Drifted -> Dead with probability 0.00015
OK -> Dead with probability 0.00015

FilterFailures

- OK
- Degrade
- Dead

OK -> Degrade with probability 0.051
Degrade -> Dead with probability 0.007
Degrade -> OK with probability 0.051
OK -> Dead with probability 0.007
error model SensorFailures
features
  OK: initial state;
  Drifted: error state;
  Dead: error state;
end SensorFailures;

error model implementation SensorFailures.Impl
events
  drift: error event occurrence poisson 0.083;
  die: error event occurrence poisson 0.00001;
  dieByDrift: error event occurrence poisson 0.00015;
transitions
  OK -[ die ]-> Dead;
  OK -[ drift ]-> Drifted;
  Drifted -[ dieByDrift ]-> Dead;
end SensorFailures.Impl;
EXTENDED MODEL OF DATA-ACQUISITION SYSTEM

- Sensors
  - sensor1: Dead: output := 15
  - sensor2: Dead: output := 15

- Filters
  - filter1: Dead: output := 0
  - filter2: Dead: output := 0

- Monitor
- Acquisition
- value
- alarmS
  - switchS
  - alarmF
- switchF

Dead: output := 15
Dead: output := 0
PROPERTIES OF INTEREST

- Are the alarms raised when the filters fail?
- Which errors lead to a sensor failure?
- What are the system effects upon a sensor failure?
- Probability that either sensor or the filters die within 76h?
- Probability that sensors die before the filters die within 512h?
- Probability that the first sensor’s output $\geq 15$ within 240h?
- Which observables are raised upon a filter failure?
- Are the sensor observables sufficient for isolating a failure?
- Can we diagnose a sensor failure from the overall system?