COMPASS: FORMAL METHODS FOR
SYSTEM-SOFTWARE CO-ENGINEERING

Viet Yen Nguyen
Lehrstuhl für Informatik 2, RWTH Aachen University
nguyen@cs.rwth-aachen.de

Technology Innovation Days, ESA/ESTEC, 2011
ABOUT MYSELF

From Hoorn

B.Sc. + M.Sc.

Ph.D.

Discipline

Computer Science

Field

Formal Methods

Topic

Model Checking

Markov Chains

Language Semantics
NPI: INTERSECT BETWEEN INDUSTRY & ACADEMIA

- **Spacecraft Software** (Industry)
- **Formal Methods** (Academia)

NPI: The intersection between Industry and Academia.
HIGH SAFETY & DEPENDABILITY REQUIREMENTS

More FDIR* Software

- More precision
- More autonomy
- More self-reliance

* Fault Detection, Isolation & Recovery

ATV

Galileo

ExoMars


**FDIR SOFTWARE IS DESIGNED TOO LATE**

<table>
<thead>
<tr>
<th>Activities</th>
<th>Phases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Phase 0</td>
</tr>
<tr>
<td>Mission/Function</td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
</tr>
<tr>
<td>Definition</td>
<td></td>
</tr>
<tr>
<td>Verification</td>
<td></td>
</tr>
<tr>
<td>Production</td>
<td></td>
</tr>
<tr>
<td>Utilization</td>
<td></td>
</tr>
<tr>
<td>Disposal</td>
<td></td>
</tr>
</tbody>
</table>

**FDIR Concepts/FDIR Requirement**

**FDIR Design**

**Nominal system-software maturation**

**FDIR maturation in AR/QR??**
FDIR SOFTWARE IS HOLISTIC
FDIR MATURATION TRIGGERS DESIGN CHANGE

Avoid change to exit vicious circle

Update FDIR
Requirements

Update FDIR
Concepts/
Analysis/Design

Redo Safety/
Dependability Analyses

Update System/
Software Design

Ineffective FDIR, designed to avoid change instead of meeting requirements.

Low confidence safety/dependability analyses for system + FDIR

System design is mature. Necessary changes are avoided.
COMPASS TOOLSET (SOLUTION)
FDIR CALLS FOR MODEL-BASED DESIGN + ANALYSIS

- Develop system/software/FDIR design early.
- Refine model upon design maturation.
- Define requirements concisely.
- Decompose requirements coherently.
- Improve understanding of system behaviour.
- Generate proofs and design artefacts.
- Cheap early V&V.
ARCHITECTURE & ANALYSIS DESIGN LANGUAGE:
ONE LANGUAGE FOR ALL SYSTEM-LEVEL ASPECTS

- Standardised by SAE.
- Component-oriented and hierarchical.
- HW (processors, devices, bus, etc.).
- SW (threads, etc.).
- Modes and mode transitions.
- Communication by event data port connections.
- Dynamic reconfiguration.
- Error events and error states.
FORMAL SEMANTICS: PRECISE & UNAMBIGUOUS

SEMANTICS OF 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
    - 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)

\[
\begin{align*}
\langle m\text{-primary, } v = 6.0 \rangle & \quad \langle m\text{-charged, } e = 100.0, v = 6.0 \rangle \quad \downarrow 40.0 \\
\langle m\text{-primary, } v = 6.0 \rangle & \quad \langle m\text{-charged, } e = 20.0, v = 6.0 \rangle \quad \downarrow \tau\langle\text{voltage:...}\rangle \\
\langle m\text{-primary, } v = 4.4 \rangle & \quad \langle m\text{-charged, } e = 20.0, v = 4.4 \rangle \quad \downarrow \tau\langle\text{empty}\rangle \\
\langle m\text{-backup, } v = 6.0 \rangle & \quad \langle m\text{-depleted, } e = 20.0, v = 4.4 \rangle \quad \downarrow 40.0 \\
\langle m\text{-backup, } v = 6.0 \rangle & \quad \langle m\text{-depleted, } e = 20.0, v = 4.4 \rangle \quad \downarrow \langle m\text{-charged, } e = 20.0, v = 6.0 \rangle
\end{align*}
\]
CAPTURING REQUIREMENTS PRECISELY

PATTERNS: LOGIC-FREE SPECIFICATION OF REQUIREMENTS
By [Dwyer et al. 1999] & [Grunske 2008]

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

• $\mathcal{P} > 0.98 [\Box [0,10] (\text{voltage} \geq 80)]$ (Continuous Stochastic Logic)

• $\Box (x \leq \text{voltage} \leq y)$ (Linear Temporal Logic)
COMPASS TOOLSET: AADL + PATTERNS + ANALYSIS

Nominal Model

Error Model

Fault Injections

Req's (patterns)

Extended Model

Formal Canonical Models

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 eval.
- Probabilistic risk assessment of fault tree.

Validation

- Consistency check
- Assertion check
- Simulation
CHECKING SYSTEM/SW/FDIR CORRECTNESS

Are we designing the system right?

- Simulations
  - Random
  - Step-by-step
  - By constraints

- Model checking (exhaustive testing)
  - Full: all traces
  - Bounded: all traces up to depth $n$
Which faults lead to a failure mode?
Which failure modes are entered because of a fault?

- Fault tree generation
  - Static (AND/OR-gates)
  - Dynamic (AND/OR/PAND-gates)
- FMEA generation
  Up to $n$ combination of failures
Which faults leads to a failure mode?
Which failure modes are entered because of a fault?

- Fault tree generation
  - Static (AND/OR-gates)
  - Dynamic (AND/OR/PAND-gates)
- FMEA generation
  Up to \( n \) combination of failures
PROBABILISTIC RISK ASSESSMENT

What is the probability of a failing system state?

- Markovian evaluation
  Probability of an event

- Markovian verification
  Probability of ordering of system events

Based on FTA to a Continuous Time Markov Chain (CTMC) reduction and then computing transient state probabilities
What is the probability and time to failure under degraded operations?

- **Reliability**
  Probability of failure.

- **Availability**
  Time to failure.

Based on state space to CTMC reduction and then computing transient state probabilities.
FAULT DETECTION AND RECOVERY CORRECTNESS

Can failures be detected?
Are failures recovered from?

- Fault detection
  Observables triggering

- Fault isolation
  Mapping observables to failures

- Fault recovery
  Model checking response property

- Diagnosis constraints
  Deciding sufficiency of observables
ANALYSIS: AUTOMATIC GENERATION OF ARTEFACTS

Model Checking/Simulation

Fault Tree Generation

FMEA Table Generation

Probabilistic Risk Assessment

Performability

Diagnosis

Constraints

Cumulative Distribution Function

Mode Analysis Fault Tolerance Evaluation (Dynamic) Fault Tree Verification (Dynamic) Fault Event
LARGEST CASE STUDY

• Thermal regulation system
• FDIR mode management
• Satellite platform inc. FDIR HW/SW
In case a failure occurs (at any layer of the system hierarchy), then:

- **L0**: minor failures and localized to one subsystem.
- **L1**: multiple subsystems failure, recovery by software.
- **L2**: multiple subsystems failure, recovery (partially) by hardware.
- **L3**: failure on hardware with recovery by hardware.
- **L4**: failure on recovery hardware itself.

SATELLITE PLATFORM INC. FDIR HW/SW

- Needs to have a reliability of at least XXX % in YY years.
- Needs to recover itself from single failures.
WHAT’S MODELED (THIS LIST IS NOT EXHAUSTIVE)

Behaviour:
- Voting algorithms.
- Off-range checkers.
- Initialization sequences.
- Recovery sequences, disabling/enabling system elements.
- Status flags.
- Failure configurations.
- Evolution of physical quantities.
- Timing constraints.

Platform

- **AOCS**
  - Gyro
  - RW
  - Earth Sensors
  - Sun Sensors

- **CDU**
  - Processors
  - Receivers

- **TT&C**
  - Transmitters

- **EPS**
  - Batteries
  - Solar Panels
## MODEL AND REQUIREMENTS METRICS

<table>
<thead>
<tr>
<th>Scope</th>
<th>Metric</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td>Components</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>Ports</td>
<td>937</td>
</tr>
<tr>
<td></td>
<td>Modes</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>Error models</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Recoveries</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Nominal state space</td>
<td>48421100</td>
</tr>
<tr>
<td></td>
<td>LOC (without comments)</td>
<td>3831</td>
</tr>
<tr>
<td><strong>Requirements</strong></td>
<td>Propositional</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Absence</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Universality</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Probabilistic Invariance</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Probabilistic Existence</td>
<td>1</td>
</tr>
</tbody>
</table>
(RE)GENERATION OF ENGINEERING ARTEFACTS

Model Checking/Simulation

Fault Tree Generation

FMEA Table Generation

Probabilistic Risk Assessment

Performability

Diagnosis

Constraints
<table>
<thead>
<tr>
<th>Formal nature pushes for resolving design details.</th>
<th>No link with current engineering models (matlab, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic generation of fault trees &amp; reliability proofs save time.</td>
<td>No progress indicators and estimators on analysis time.</td>
</tr>
<tr>
<td>System-level failures impact computation time of analyses more than lower-level failures.</td>
<td></td>
</tr>
<tr>
<td>Single integrated model of system + FDIR + erroneous behavior avoids mutual assumptions.</td>
<td></td>
</tr>
<tr>
<td>Incremental engineering naturally supported through nesting.</td>
<td></td>
</tr>
</tbody>
</table>
SUMMARY (AND OUR GRAND VIEW)

• Overarching system-level **model** and **analysis** of nominal, degraded and erroneous behaviour:
  
  – Model: Architecture & Analysis Design Language (AADL)
    • i.e. stochastic hybrid automata.
  
  – Requirements: Property patterns
    • i.e. temporal/probabilistic logics.
  
  – Analysis: Simulation, model checking, fault tree generation, FMEA table generation, performability and diagnosis constraints
    • i.e. reachability analysis & transient probabilities computation.
FORMAL METHODS FOR SPACECRAFT SOFTWARE

Overarching system-level model and **automated** analysis of nominal, degraded and erroneous behaviour.

Using **formal methods**:  
- Stochastic hybrid automata  
- Temporal/probabilistic logics  
- (Reversed) reachability  
- TwinPlant reachability  
- Transient probabilities computation
MORE INFORMATION

• My website: moves.cs.rwth-aachen.de/~nguyen

• COMPASS Toolset: compass.informatik.rwth-aachen.de/
  – Freely available for everyone in ESA-member states.
  – Graphical modeller will be released early 2012.