AltaRica OCAS: 
*Overview & UoY Experiences*

Oleg Lisagor
University of York
MISSA approach

Methodology

- Failure Logic Modelling
- Failure Effects Modelling
- FLM / FEM Hybrids
- Failure Injection
- ...

Model-Based Safety Assessment

Tools

Language

- AltaRica
- StateMate
- SCADE
- Custom Notations

Analysis

- FT Synthesis
- Sequence Generation
- Model-Checking
- ...

The University of York
MISSA approach

Methodology
- Failure Logic Modelling
- Failure Effects Modelling
- FLM / FEM Hybrids
- Failure Injection
-...

Model-Based Safety Assessment

Language
- AltaRica
- StateMate
- SCADE
- Custom Notations

Analysis
- OCAS
- NuSMV-SA
- FT Synthesis
- Sequence Generation
- Model-Checking
-...

Cecilia
OCAS
AltaRica Language

Model Architecture

- Components
  - Hierarchically Nested

- Data Flow
  - Variety of types
  - Connect component i/o
    - In fact are just assertions
  - Propagation is “timeless”

- Synchronisations
  - Between component events
  - And “external cause”

Basic Components

- Key modelling concepts:
  - Events
  - States
  - Input & Output (flows)

- Two types of behaviour:
  - “Transitions”
    - Triggered by events
    - Guarded by predicate over state & input values
  - “Assertions”
    - Output value assignments based on predicates over state and input
## Example Component

### Diagram

```
CONTROL

Pressure_set_point

Controller
Calculates control_pressure supplied to the Actuator

Feedback

Actuator
Braking actuator and pressure feedback sensor unit

Default_set_point

Braking
```

### Table

<table>
<thead>
<tr>
<th>Output Failure Mode</th>
<th>Description / Immediate causes</th>
<th>Input Deviation Logic</th>
<th>Component Malfunction Logic</th>
<th>$\lambda$ (f/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-Control_pressure</td>
<td>Omission of Control pressure. Caused by a failure of the Controller or (omission of pressure set-point and omission of default braking value) or an invalid pressure feedback value</td>
<td>(O-Pressure_set_point &amp; O-Default_set_point)</td>
<td>Controller_failure</td>
<td>5e-6</td>
</tr>
<tr>
<td>Vs_0-Control_pressure</td>
<td>Control pressure stuck at 0. Caused by Controller memory stuck at 0 or pressure set-point stuck at 0 or (omission of Pressure set-point and Default braking value stuck at 0)</td>
<td>(Vs_0-Default_set_point &amp; O-Pressure_set_point)</td>
<td>Controller_memory_stuck_at_0</td>
<td>3e-6</td>
</tr>
</tbody>
</table>
### Example Component

<table>
<thead>
<tr>
<th>Output Failure Mode</th>
<th>Description / Immediate causes</th>
<th>Input Deviation Logic</th>
<th>Component Malfinction Logic</th>
<th>λ (f/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>O-Control_pressure</td>
<td>Omission of Control pressure. Caused by a failure of the Controller or (omission of pressure set-point and omission of default braking value) or an invalid pressure feedback value</td>
<td>(O-Pressure_set_point &amp; O-Default_set_point)</td>
<td>Controller_failure</td>
<td>5e-6</td>
</tr>
<tr>
<td>Vs_0-Control_pressure</td>
<td>Control pressure stuck at 0. Caused by Controller memory stuck at 0 or pressure set-point stuck at 0 or (omission of Pressure set-point and Default braking value stuck at 0)</td>
<td>(Vs_0-Default_set_point &amp; O-Pressure_set_point)</td>
<td>Controller_memory_stuck_at_0</td>
<td>3e-6</td>
</tr>
</tbody>
</table>

---

**In AltaRica must also define a state and transition (trivial)**

**Output Flow (boolean)**

**Input Flow**

**Assertion**

**Event**
<table>
<thead>
<tr>
<th>Name</th>
<th>Comment</th>
<th>Law</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller_failure</td>
<td></td>
<td>exponential(5e-6)</td>
</tr>
<tr>
<td>Controller_memory_stuck</td>
<td></td>
<td>exponential(3e-6)</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Value</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Failed</td>
<td>bool</td>
<td>false</td>
</tr>
<tr>
<td>Stuck</td>
<td>bool</td>
<td>false</td>
</tr>
</tbody>
</table>
Stuck := false;

trans
  // Transitions:
  // Failures causing state change
  // (necessary "intermediate" step in specification of
  // the "component malfunction logic" in AltaRica)
  not Failed |- Controller_failure -> Failed := true;
  not Stuck |- Controller_memory_stuck -> Stuck := true;

assert
  // Assertions:
  // Output Flow value determined by Input Flows
  // and (internal) state
  Omis_Pres = {
    (Omis_PSP and Omis_DSP) or
    Vmax_Feed or Vmin_Feed or
    Failed);

Stuck_Pres = {
  (Stuck_DSP and Omis_PSP) or
  Stuck_PSP or
  Stuck}
PSP : HiPHOPS1_ControlFM : in ;
PSP^0mis : bool : in ;
PSP^Stuck : bool : in ;
DSP : HiPHOPS1_ControlFM : in ;
DSP^0mis : bool : in ;
DSP^Stuck : bool : in ;
Feed : HiPHOPS1_FeedbackFM : in ;
Feed^Vmax : bool : in ;
Feed^Vmin : bool : in ;

trans

// Transitions:
not Failed |- Controller_failure -> Failed := true;
not Stuck |- Controller_memory_stuck -> Stuck := true;

assert

// Assertions:
Pres^0mis = {
    (PSP^0mis and DSP^0mis) or
    Feed^Vmax or Feed^Vmin or
    Failed
};
Loops & Sequences

- Events can be associated with laws
  - Normally: probability distribution laws
    - E.g. “exponential(4e-7)”
  - Special “instantaneous” law: “Dirac(0)”
    - Fires as soon as guard becomes true
    - Can be used for “converting” input flow into state

- Assertion / Flow Loops
  - A potential problem for model analysis and simulation
    - Problem for most dataflow language
  - Simple loops can be resolved by inserting Dirac(0) delay

```
trans
S !\equiv In |\cdot Delay\cdot S := In;
assert
Out = S
```

- Similar approach for modelling sequence constraints over flows
Cecilia OCAS: Model Analysis

● Sequence Generation
  ▪ User specifies condition of interest and “depth” of search
  ▪ Tool calculates Minimal Cut Sets (or sequences)
    ◆ That can cause specified condition
    ◆ Up to a maximum size (“depth”)
  ▪ Acceptably fast for most models
    ◆ Most FEMs
    ◆ FLMs comparable to “basic” HiP-HOPS
    ◆ Some “dynamic constructs” can be “expensive”

● Simulation
  ▪ User triggers (non-instantaneous) events manually
  ▪ Tool calculates the effect
  ▪ Can be visualised very effectively

● Fault Tree Synthesis
  ▪ Under certain conditions (e.g. no flows in transition guards)
UoY Experience

- Performed some modelling experiments
  - A number of teaching examples
  - ARP-4761 Wheel Braking System
  - Aircraft Electrical System
  - Some experiments on A380 Fuel System

- AltaRica OCAS (language)
  - Very efficient for “basic” methodology
  - Some “teething issues” for complex behaviour
    - Synchronisations
    - Temporal events ($\text{Dirac}(x)$)

- Cecilia OCAS (analysis tools)
  - Generally good results
  - Some types of complex dynamic behaviour can be very expensive in terms of computation time
Not meant to be presented, unless called upon in questions
MISSA WP4: Scope

- Safety Assessment of Platform and System Architecture
  - Hardware and Software
  - Earlier iterations of Preliminary System Safety Assessment (PSSA)
  - Preliminary Aircraft Safety Assessment (PASA)
MISSA WP4: Objectives

- Support definition of a single model
  - Provide methodology guidance
    - Tool & language independent
    - Different methodologies / approaches
  - Approaches to model validation and justification
    - Jointly with MISSA WP6

- Support Incremental Assessment
  - Comparison of models at different development / assessment milestones
  - Different authors / different formalisms

- Support composition of Independently Defined Models
  - Composition in absence of well defined interfaces
MISSA approach

Model-Based Safety Assessment

= Methodology + Language + Analysis

<table>
<thead>
<tr>
<th>Failure Logic Modelling</th>
<th>AltaRica</th>
<th>Sequence Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Effects Modelling</td>
<td>StateMate</td>
<td>Fault Tree Synthesis</td>
</tr>
<tr>
<td>Hybrid</td>
<td>SCADE</td>
<td>Model-Checking</td>
</tr>
<tr>
<td>...</td>
<td>Custom Notations</td>
<td>...</td>
</tr>
</tbody>
</table>

Tools
MISSA approach

Model-Based Safety Assessment

= Methodology + Language + Analysis

- Failure Logic Modelling
- Failure Effects Modelling
- Hybrid
- AltaRica
- StateMate
- SCADE
- Custom Notations
- Sequence Generation
- Fault Tree Synthesis
- Model-Checking
- ...