Modelling of a HIPPS with AltaRica
RAMS seminar 03.03.2017
Yun Zhang
Antoine Rauzy
Issues

• Reliability assessment of complex system raises a number of challenging issues:
  – How to design model efficiently?
  – How to reuse parts/components of models?
  – How to validate models?
  – How to have a quality assurance on the outputs of the modeling process?
  – How to maintain models throughout the life-cycle of systems?
  – How to better integrate reliability engineering with other disciplines? …
Categories of modelling languages

- **Boolean Formalisms**
  - Fault Trees
  - Event Trees
  - Reliability Block Diagrams

- **Transitions Systems**
  - Markov Chains
  - Dynamic Fault Trees
  - Stochastic Petri Nets
  - ...

- **Universal Languages**
  - Agent Based Models
  - Matlab
  - Java/C++
  - ...

Is there a generic/unified *modeling framework* to assess reliability of complex systems?

Guarded Transitions Systems as implemented in AltaRica
Problem statement

• We focus on modelling of a HIPPS with AltaRica
• The problem is mainly about
  o how to design a maintenance model to help assessing condition monitoring and optimization of maintenance policies given the description of constraints

• But it could also be relevant for problems like
  o how to analyze production of the system
  o how to do risk analysis
  o …
System description

Logic solver (LS)

Shut down valves (SDV)

Sensors (S)

1002

2003
Features of the HIPPS

• **Components:**
  – Continuously monitored components (logic solver and sensors)
  – Periodically tested components (shutdown valves)

• **Objectives:**
  – Reduce number of maintenance interventions
  – Minimize system downtime
Step 0. Block diagram
Step 1. Repairable component

Shut down valves (SDV)

Sensors (S)

Logic solver (LS)
Finite state automaton for repairable component

- _state = WORKING
- _state = FAILED
Step 2. Tested component

Logic solver (LS)

Shut down valves (SDV)

Sensors (S)
Finite state automaton for a tested component

```
(startTest) _state = WORKING _phase = TEST
   |         |         |
   v         v         v
(completeTest) _state = WORKING _phase = TEST

(failure) _state = FAILED_UNDETECTED _phase = TEST
   |         |         |
   v         v         v
(continue) _state = FAILED_UNDETECTED _phase = TEST

(repair) _state = FAILED_DETECTED _phase = TEST
   |         |         |
   v         v         v
(continue) _state = FAILED_DETECTED _phase = TEST

(startTest) _state = WORKING _phase = OPERATION
   |         |         |
   v         v         v
(completeTest) _state = WORKING _phase = OPERATION

(failure) _state = FAILED_UNDETECTED _phase = OPERATION
   |         |         |
   v         v         v
(continue) _state = FAILED_UNDETECTED _phase = OPERATION

(repair) _state = FAILED_DETECTED _phase = OPERATION
   |         |         |
   v         v         v
(continue) _state = FAILED_DETECTED _phase = OPERATION
```
Step 3. Periodic maintenance

Logic solver (LS)

1002

Shut down valves (SDV)

Sensors (S)

2003

W  F  W  F

W  F  W  F

W  F  W  F

W  F  W  F
Finite state automaton for a monitored component

- \_state = WORKING
  \_phase = MAINTENANCE

- state = FAILED_DETECTED
  \_phase = MAINTENANCE

- \_state = WORKING
  \_phase = OPERATION

- state = FAILED_DETECTED
  \_phase = OPERATION

MAINTENANCE PHASE

OPERATION PHASE
Finite state automaton for a tested component

- **State**: WORKING
  - **Phase**: MAINTENANCE
  - Transition: startMaintenance

- **State**: FAILED_UNDETECTED
  - **Phase**: OPERATION
  - Transition: failure

- **State**: FAILED_DETECTED
  - **Phase**: MAINTENANCE

- **State**: WORKING
  - **Phase**: TEST

- **State**: FAILED_UNDETECTED
  - **Phase**: TEST

- **State**: FAILED_DETECTED
  - **Phase**: TEST

- **State**: WORKING
  - **Phase**: MAINTENANCE

- **State**: FAILED_UNDETECTED
  - **Phase**: OPERATION

- **State**: FAILED_DETECTED
  - **Phase**: OPERATION

- **State**: WORKING
  - **Phase**: TEST

- **State**: FAILED_UNDETECTED
  - **Phase**: TEST

- **State**: FAILED_DETECTED
  - **Phase**: TEST

- **State**: WORKING
  - **Phase**: MAINTENANCE

- **State**: FAILED_UNDETECTED
  - **Phase**: OPERATION

- **State**: FAILED_DETECTED
  - **Phase**: OPERATION

- **State**: WORKING
  - **Phase**: TEST

- **State**: FAILED_UNDETECTED
  - **Phase**: TEST

- **State**: FAILED_DETECTED
  - **Phase**: TEST
Step 4. Condition based maintenance

Logic solver (LS)

Shut down valves (SDV)

Sensors (S)
Our knowledge about the state of the system maybe different from actual state of the system…
State at different levels

System working

- HIPPS
- Sensors
  - S
  - S
  - S
- Controller
  - LS
- Actuators
  - SDV
  - SDV

Legend:
- Working
- Degraded
- Failed

Levels:
- System level
- Sub-system level
- Component level
State at different levels

System working

HIPPS

- Sensors
  - S
  - S
  - S
- Controller
  - LS
- Actuators
  - SDV
  - SDV

- working
- degraded
- failed

System level
Sub-system level
Component level
State at different levels

System working

- **HIPPS**
  - **Sensors**
  - **Controller**
  - **Actuators**
    - **SDV**
    - **SDV**

Legend:
- Green: Working
- Yellow: Degraded
- Red: Failed
State at different levels

System degraded

- HIPPS
  - Sensors
    - S
    - S
    - S
  - Controller
    - LS
  - Actuators
    - SDV
    - SDV

- working
- degraded
- failed

System level
Sub-system level
Component level
State at different levels

System failed

- HIPPS
  - Sensors
    - S
    - S
    - S
  - Controller
    - LS
  - Actuators
    - SDV
    - SDV

System level
Sub-system level
Component level
State at different levels

System failed

- HIPPS
  - Sensors: S S S
  - Controller: LS
  - Actuators: SDV SDV

System level
Sub-system level
Component level
State at different levels

System failed

- HIPPS
  - Sensors
    - S
    - S
    - S
  - Controller
    - LS
  - Actuators
    - SDV
    - SDV
## Component states

<table>
<thead>
<tr>
<th>Components</th>
<th>Status</th>
<th>Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>working</td>
<td>failed detected</td>
</tr>
<tr>
<td>S2</td>
<td>working</td>
<td>failed detected</td>
</tr>
<tr>
<td>S3</td>
<td>working</td>
<td>failed detected</td>
</tr>
<tr>
<td>LS</td>
<td>working</td>
<td>failed detected</td>
</tr>
<tr>
<td>SDV1</td>
<td>working</td>
<td>failed undetected</td>
</tr>
<tr>
<td>SDV2</td>
<td>working</td>
<td>failed undetected</td>
</tr>
</tbody>
</table>
# Sub-system states

<table>
<thead>
<tr>
<th>Sub-system</th>
<th>working</th>
<th>degraded</th>
<th>failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensors</td>
<td>3 working</td>
<td>2 working</td>
<td>1 or 0 working</td>
</tr>
<tr>
<td>Logic solver</td>
<td>1 working</td>
<td></td>
<td>0 working</td>
</tr>
<tr>
<td>Shutdown valves</td>
<td>2 working</td>
<td>1 working</td>
<td>0 working</td>
</tr>
</tbody>
</table>
# System states

<table>
<thead>
<tr>
<th>System</th>
<th>working</th>
<th>degraded</th>
<th>failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensors</td>
<td>(and) working or degraded</td>
<td>(and) degraded</td>
<td>(or) failed</td>
</tr>
<tr>
<td>Logic solver</td>
<td>(and) working</td>
<td>(and) working</td>
<td>(or) failed</td>
</tr>
<tr>
<td>Shutdown valves</td>
<td>(and) working or degraded</td>
<td>(and) degraded</td>
<td>(or) failed</td>
</tr>
</tbody>
</table>
Finite state automaton for decision rules