Architecture-Supported Audit Processor: Interactive, Query-Driven Assurance

Sam Procter  sprocter@sei.cmu.edu
Jérôme Hugues  jihuques@sei.cmu.edu
Objective

Goal: To ease understanding of safety and assurance argumentation

**Challenge 1:** Assurance evidence should be contextualized within explicit safety arguments.

**Challenge 2:** Assurance argumentation should be hierarchical.

**Challenge 3:** Assurance evidence should be modular and composable.
Research Challenge 1: Linking System Safety to Architecture

Most system safety techniques (e.g., STPA, FMEA, FTA) are too abstract to be automated using formal methods.

Extensive system behavior specifications would enable formal reasoning, but would be prohibitively difficult to produce for large-scale systems.

What is needed, then, is a “middle path” which would contain only enough behavioral information necessary to support system safety automation. We have explored using EMV2 error propagations as the keystone joining formal methods and system assurance.

\[
\begin{align*}
\{P\}S\{Q\}, \{Q\}T\{R\} \\
\{P\}S; T\{R\}
\end{align*}
\]

\[
TS \models \Box(\neg b \rightarrow \Box(a \land \neg b))
\]
Outline

1. Context
2. Background
   1. STPA and SAFE
   2. AADL and OSATE
   3. The Pulse Oximeter Example
3. Architecture-Supported Audit Processor (ASAP)
4. Next steps
Figure 2.1: Overview of the basic STPA Method

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
The PulseOx Forwarding Example

Pulse oximeter reads blood-oxygen saturation from a patient, monitoring software displays an alarm if values are out of expected range
The PulseOx Forwarding Example
The PulseOx Forwarding Example
The PulseOx Forwarding Example
The PulseOx Forwarding Example

- Safety problem to avoid: Incorrect SpO\textsubscript{2} displayed
The PulseOx Forwarding Example

- Safety problem to avoid: Incorrect SpO₂ displayed
The PulseOx Forwarding Example

- Safety problem to avoid: Incorrect SpO₂ displayed
- AADL’s “Error Modeling” (EMV2) annex can model these error propagations
Outline

1. Context
2. Background
3. Architecture-Supported Audit Processor (ASAP)
   1. Viewpoints
      1. Fundamentals
      2. Connected Neighbors
      3. Unsafe Control Actions
   2. Research Challenge: Linking System Safety to Architecture
4. Next steps
ASAP’s Viewpoints

ASAP is a collection of “viewpoints” of a system

• Similar, in some ways, to views of the system’s logical structure or physical implementation

• ASAP’s focus is on safety, rather than functionality or other system aspects

• There are three viewpoints (though more are planned) which align with STPA
ASAP’s Viewpoints

ASAP is a collection of “viewpoints” of a system

• Similar, in some ways, to views of the system’s logical structure or physical implementation

• ASAP’s focus is on safety, rather than functionality or other system aspects

• There are three viewpoints (though more are planned) which align with STPA

Figure 2.1: Overview of the basic STPA Method

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
ASAP’s Viewpoints

ASAP is a collection of “viewpoints” of a system

• Similar, in some ways, to views of the system’s logical structure or physical implementation
• ASAP’s focus is on safety, rather than functionality or other system aspects
• There are three viewpoints (though more are planned) which align with STPA

Viewpoint 1: Fundamentals

Figure 2.1: Overview of the basic STPA Method

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
ASAP’s Viewpoints

ASAP is a collection of “viewpoints” of a system

• Similar, in some ways, to views of the system’s logical structure or physical implementation
• ASAP’s focus is on safety, rather than functionality or other system aspects
• There are three viewpoints (though more are planned) which align with STPA

Viewpoint 1: Fundamentals

Viewpoint 2: Connected Neighbors

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
ASAP’s Viewpoints

ASAP is a collection of “viewpoints” of a system

- Similar, in some ways, to views of the system’s logical structure or physical implementation
- ASAP’s focus is on safety, rather than functionality or other system aspects
- There are three viewpoints (though more are planned) which align with STPA

Viewpoint 1: Fundamentals

Viewpoint 2: Connected Neighbors

Viewpoint 3: Unsafe Control Actions

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
Viewpoint 1: Fundamentals (Hierarchy)

<table>
<thead>
<tr>
<th>Semantic</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazard BadInfoDisplayed</td>
<td>Accident PatientHarmend</td>
<td>Constraint ShowGoodInfo</td>
</tr>
<tr>
<td></td>
<td>Constraint</td>
<td>Incorrect information is sent to the display</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Abstract patient</td>
</tr>
<tr>
<td></td>
<td>Error Type</td>
<td>Error Type SpO2ValueHigh</td>
</tr>
<tr>
<td></td>
<td>Explanations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hazardous Factor</td>
<td>SpO2 Information</td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td>BadInfoDisplayed</td>
</tr>
<tr>
<td></td>
<td>System Element</td>
<td>Event Data Port DispSpO2</td>
</tr>
</tbody>
</table>

Accident Level 1

Accident 1-1  Accident 1-2  ...  Accident 1-1

Hazard 1-1-1  Hazard 1-1-2  ...  Hazard 1-1-m

Constraint 1-1-1-1  Constraint 1-1-1-2  ...  Constraint 1-1-1-n
Viewpoint 1: Fundamentals (Link to system)

Hazard BadInfoDisplayed

<table>
<thead>
<tr>
<th>Semantic</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantics</td>
<td>Hazard BadInfoDisplayed</td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Accident</td>
<td>Accident PatientHarmed</td>
</tr>
<tr>
<td>Constraint</td>
<td>Constraint</td>
<td>Constraint ShowGoodInfo</td>
</tr>
<tr>
<td>Description</td>
<td>Incorrect information is sent to the display</td>
<td></td>
</tr>
<tr>
<td>Environment Element</td>
<td>Abstract patient</td>
<td></td>
</tr>
<tr>
<td>Error Type</td>
<td>Error Type SpO2ValueHigh</td>
<td></td>
</tr>
<tr>
<td>Explanations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hazardous Factor</td>
<td>SpO2 Information</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>BadInfoDisplayed</td>
<td></td>
</tr>
<tr>
<td>System Element</td>
<td>Event Data Port DispSpO2</td>
<td></td>
</tr>
</tbody>
</table>

Hazard = System State + Environment State
(Error Type + Port) + (Component)
Linking System Safety to Architecture with EMV2 Error Types

Hazard BadInfoDisplayed

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazard BadInfoDisplayed</td>
<td></td>
</tr>
<tr>
<td>Accident</td>
<td>Accident PatientHarmened</td>
</tr>
<tr>
<td>Constraint</td>
<td>Constraint ShowGoodInfo</td>
</tr>
<tr>
<td>Description</td>
<td>Incorrect information is sent to the display</td>
</tr>
<tr>
<td>Environment Element</td>
<td>Abstract patient</td>
</tr>
<tr>
<td>Error Type</td>
<td>Error Type SpO2ValueHigh</td>
</tr>
<tr>
<td>Explanations</td>
<td></td>
</tr>
<tr>
<td>Hazardous Factor</td>
<td>SpO2 Information</td>
</tr>
<tr>
<td>Name</td>
<td>BadInfoDisplayed</td>
</tr>
<tr>
<td>System Element</td>
<td>Event Data Port DispSpO2</td>
</tr>
</tbody>
</table>

Hazard = System State + Environment State
(Error Type + Port) + (Component)
Linking System Safety to Architecture with EMV2 Error Types

Hazard BadInfoDisplayed

<table>
<thead>
<tr>
<th>Semantic</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazard BadInfoDisplayed</td>
<td>Accident</td>
<td>Accident PatientHarmmed</td>
</tr>
<tr>
<td></td>
<td>Constraint</td>
<td>Constraint ShowGoodInfo</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Incorrect information is sent to the display</td>
</tr>
<tr>
<td></td>
<td>Environment Element</td>
<td>Abstract patient</td>
</tr>
<tr>
<td></td>
<td>Error Type</td>
<td>Error Type SpO2ValueHigh</td>
</tr>
<tr>
<td></td>
<td>Explanations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hazardous Factor</td>
<td>SpO2 Information</td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td>BadInfoDisplayed</td>
</tr>
<tr>
<td></td>
<td>System Element</td>
<td>Event Data Port DispSpO2</td>
</tr>
</tbody>
</table>

Hazard = System State + Environment State
(Error Type + Port) + (Component)
Linking System Safety to Architecture with EMV2 Error Types

Hazard BadInfoDisplayed

<table>
<thead>
<tr>
<th>Semantic</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazard BadInfoDisplayed</td>
<td>Accident</td>
<td>Accident PatientHarmed</td>
</tr>
<tr>
<td></td>
<td>Constraint</td>
<td>Constraint ShowGoodInfo</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Incorrect information is sent to the display</td>
</tr>
<tr>
<td></td>
<td>Environment Element</td>
<td>Abstract patient</td>
</tr>
<tr>
<td></td>
<td>Error Type</td>
<td>Error Type SpO2ValueHigh</td>
</tr>
<tr>
<td></td>
<td>Explanations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hazardous Factor</td>
<td>SpO2 Information</td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td>BadInfoDisplayed</td>
</tr>
<tr>
<td></td>
<td>System Element</td>
<td>Event Data Port DispSpO2</td>
</tr>
</tbody>
</table>

Hazard = System State + Environment State
(Error Type + Port) + (Component)
Viewpoint 2: Connected Neighbors

[Diagram of connected neighbors showing PulseOx Forwarding System Imp Instance with interactions between doctor, clinician, patient, and various components such as PulseOx, LogicSpO2, LogicDerivedAlarm, DispSpO2, DispDerivedAlarm, ehrSpO2, electronicHealthRecord, and appLogic, appDisplay.]
Viewpoint 2: Connected Neighbors
Viewpoint 2: Connected Neighbors
Viewpoint 3: Unsafe Control Actions

Communication Channels (ie, control actions and sensor feedback)

- patient.PatientFingercip -> pulseOx.SensorInput
- pulseOx.PO0utSpO2 -> electronicHealthRecord.ehrSpO2
- pulseOx.PO0utSpO2 -> appLogic.StoreSpO2Thread.incoming_spo2
- appLogic.CheckSpO2Thread.Alarm -> appDisplay.HandleAlarmThread.Ala...
- pulseOx.PO0utSpO2 -> appDisplay.UpdateSpO2Thread.SpO2

Top-Level Errors (ie, abstract guidewords)

- ItemValueError
- ItemTimingError
- ViolatedConstraint
- ServiceError

X means one or more errors in this family can propagate on this channel
Viewpoint 3: Unsafe Control Actions

Communication Channels (ie, control actions and sensor feedback)

- patient.PatientFingerclip -> pulseOx.SensorInput
- pulseOx.POOutSpO2 -> electronicHealthRecord.ehrSpO2
- pulseOx.POOutSpO2 -> appLogic.StoreSpO2Thread.incoming_spO2
- appLogic.CheckSpO2Thread.Alarm -> appDisplay.HandleAlarmThread.Alarm
- pulseOx.POOutSpO2 -> appDisplay.UpdateSpO2Thread.SpO2

Top-Level Errors (ie, abstract guidewords)

- ItemValueError
- ItemTimingError
- ViolatedConstraint
- ServiceError

X means one or more errors in this family can propagate on this channel

Refined Errors (ie, domain / system-specific guidewords)

Early SpO2
- Cause: … Compensation: …

Undocumented propagation!

Late SpO2
- Cause: … Compensation: …
Outline

1. Context
2. Background
3. Architecture-Supported Audit Processor (ASAP)
4. Next steps
Next Steps

1. Deriving unsafe control actions automatically
   - EMV2’s propagations, if fully specified, describe causal scenarios in which hazards occur
   - We anticipate that integrating these into existing viewpoints, as well as new ones (e.g., FMEA), will be helpful

2. Integration with OSATE assurance case generation
   - Ongoing work at the SEI towards generating assurance cases from AADL models
   - Need to determine overlap and explore possible integrations
Architecture-Supported Audit Processor: Interactive, Query-Driven Assurance

Sam Procter sprocter@sei.cmu.edu
Jérôme Hugues jjhuques@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA  15213