Using STPA to Support Risk Management for Interoperable Medical Systems

STAMP Workshop 2015, MIT

Sam Procter, John Hatcliff
SAnToS Lab
Kansas State University

Anura Fernando
Underwriters Laboratories

Sandy Weininger
US Food and Drug Administration

Support:
This work is supported in part by the US National Science Foundation (NSF) (#1239543), the NSF US Food and Drug Administration Scholar-in-Residence Program (#1355778) and the National Institutes of Health / NIBIB Quantum Program.
Health Care Involves
A Variety of System Components

Clinical Protocols
Actuators
Sensor Data
Displays
Clinicians
Information Systems
Patient!
Motivation

- What are the types of things we could do with device integration?
  - Information forwarding
  - Automation of clinical workflows
  - Closed loop control between devices
- Unlike personal computing, medical devices are not designed to work together
- Integrating medical devices would bring myriad benefits
- ... how can we do so safely?
Outline

- Background
  - PCA Interlock Scenario
  - Medical Application Platforms
  - Tooling
- Hazard Analysis In AADL
- Architectural Integration
PCA Interlock Scenario

- Patients are commonly given patient-controlled analgesics after surgery
- Crucial to care, but numerous issues related to safety
- Data for disabling the pump exists now (just a system invariant) -- we just need to integrate it
PCA Pump Safety Interlock

Fully leverage device data streams and the ability to *control* devices

**Devices**

- PCA Pump
- Monitoring Data + Alarm Information (Capnograph)
- Monitoring Data + Alarm Information (Pulse Oximeter)

**Device Task controller**
- Enable Pump for safe time window
- Enable bolus dose only when ticket present
- PCA Bolus "Enable" Ticket
- Combined PCA Vitals Monitoring
- Aggregated Monitoring Status

**Clinician / Monitoring**
- Status Display for PCA Monitoring Application
A Medical Application Platform is a safety- and security-critical real-time computing platform for...

- Integrating heterogeneous devices, medical IT systems, and information displays via communications infrastructure, and
- Hosting applications ("apps") that provide medical utility via the ability to acquire information from and update/control integrated devices, IT systems, and displays
Unique aspects of MAP domain

- Software based
  - Hardware is interchangeable
- Component oriented
- Unclear how FTA / FMEA might apply
- Early, firm notion of system architecture
  - Standardized in UL 2800
We use medicine in our examples

... but this can extend to other compositional systems

Core idea:

Integration of heterogeneous

- Sensors,
- Actuators, and
- Complete systems,

by small chunks of software,

in a verifiable manner
Background

PCA Pump Interlock Architecture

Medical Application Platform

App

SUI App Display

Data for Display

View Display

Data should arrive once per second

Start / Stop Commands

Sensor + Alarm Data

Pulse Oximeter, Capnograph, and Patient Controlled Analgesia Pump

PCA, PR, SPO₂, ETCO₂, RR

Patient

Clinician (App Administrator)

View Display

Configuration, Alarm Clear

Attach Sensors
Tooling Vision

Analyses and Regulatory Artifacts

Clinical Use Case / Workflow Description
Requirements
Hazard Analysis
Risk Assessment

App Developer

Medical Device Coordination Framework

Assurance Case
3rd Party ICE Conformance & Safety Certification Submission Package
FDA 510K Submission Package

App Deployment

3rd Party Certifiers
FDA Evaluators
Tooling Vision

Code Generation

A. The app’s architecture is specified in a suitable formalism
   1. Components as AADL Devices / Processes
   2. Connections are specified
   3. RT/QoS Parameters are via AADL’s property-specification mechanism

B. The app is programmatically translated to Java and XML
   1. Only “Business Logic” is written by the developer

C. The app is launched on a compatible MAP
Outline

- Background
- Hazard Analysis In AADL
  - Correspondence with manual HA
  - STPA Fundamentals
  - Report Generation
- Architectural Integration
Hazard Analysis

Leveraging Semiformal Architectural Descriptions

Clinical Use Case / Workflow Description

Requirements

3rd Party Certifiers

Assurance Case

3rd Party ICE Conformance & Safety Certification Submission Package

FDA 510K Submission Package

FDA Evaluators

App Developer

MDCF

App Deployment

Risk Assessment
Hazard Analysis in AADL

What if we could draw control loops with code?

Control Algorithms
Set Points

Controller

Actuators

Sensors

Benefits:

Managers: Constrains developers so style and architectural assumptions are consistent

Developers: Guides analysis so “starting from scratch” isn’t necessary

Controlled Process

Disturbances

Measured Variables

Process Outputs

Managers:

Developers:
Hazard Analysis in AADL

AADL Equivalents of STPA’s Objects

AADL Connection

Control Algorithms
Set Points

AADL Devices

Actuators

Controlled Variables

Process Inputs

Disturbances

Controller

Process Outputs

AADL Process

Sensors

Measured Variables

AADL Abstract

Nancy Leveson. Figure 3.2, Page 66, Engineering A Safer World. MIT Press, 2011
STPA in AADL

Fundamentals

- Accident Levels
- Accidents
- System Boundaries
- **Hazards**
- Safety Constraints
- Control Actions
- Control Structure

Example

1. An inadvertent “Pump Normally” command is sent to the pump [PatientHarmed]

2. Commands are sent to the pump too quickly [PCADoS]

```plaintext
InadvertentPumpNormally : constant MAP_ErrorProperties::Hazard => [
    Number => 1;
    Description => "An inadvertent \`Pump Normally\` command is sent to the pump.\";
    Accident => PulseOx_Forwarding_ErrorProperties::PatientHarmed;
];
```

**Benefits:**

**Regulators:** Supports strong traceability both in code and in (hypertext) reports
STPA in AADL

Fundamentals

- Fundamentals
- Accident Levels
- Accidents
- System Boundaries
- Hazards
- Safety Constraints
- **Control Actions**
- Control Structure

Example

- App -> Pump: Pump Normally

Developers: Hazard Analysis artifacts are automatically in-sync with system architecture
STPA in AADL

Identifying Hazardous Control Actions

- Hazardous Control Action Table
  - Cross-product of control actions and STPA guidewords

<table>
<thead>
<tr>
<th>Control Action</th>
<th>Providing</th>
<th>Not Providing</th>
<th>Applied too Long</th>
<th>Stopped too Soon</th>
<th>Early</th>
<th>Late</th>
</tr>
</thead>
<tbody>
<tr>
<td>App -&gt; Pump: Pump Normally</td>
<td>PH</td>
<td>Not Hazardous</td>
<td>PH</td>
<td>Not Hazardous</td>
<td>PH</td>
<td>Not Hazardous</td>
</tr>
<tr>
<td>App -&gt; Disp: Patient Ok</td>
<td>BID</td>
<td>BID</td>
<td>BID</td>
<td>BID</td>
<td>BID</td>
<td>BID</td>
</tr>
<tr>
<td>PulseOx-&gt;App: Provide SpO₂</td>
<td>Not Hazardous</td>
<td>PH, BID</td>
<td>Not Hazardous</td>
<td>PH, BID</td>
<td>Not Hazardous</td>
<td>PH, BID</td>
</tr>
<tr>
<td>PulseOx-&gt;App: Provide Pulse Rate</td>
<td>Not Hazardous</td>
<td>PH, BID</td>
<td>Not Hazardous</td>
<td>PH, BID</td>
<td>Not Hazardous</td>
<td>PH, BID</td>
</tr>
</tbody>
</table>

PH = Patient Harmed  
BID = Bad Info Displayed
Control Action: App -> Pump: Pump Normally

- Providing:
  - Inadequate Sensor Operation:
    - Cause:
      - Incorrect values are gathered from one of the physiological sensors
    - Compensation:
      - Rely on multiple sensed physiological parameters to provide redundancy

- Not Providing:
  - Not hazardous
The Annotated Control Loop

Control Action: App → PCA Pump

*Inappropriate Control Action:*
Inadvertent “Pump Normally” command

Feedback: PulseOx → App

*Inadequate Feedback:*
Sends bad SpO₂

Controller: App Logic

*Process Model Incorrect:*
Wrongly believes patient to be healthy

Actuator: PCA Pump

*Inadequate Operation:*
Pumps Normally

Sensor: Pulse Oximeter

*Inadequate Operation:*
SpO₂ value incorrect

Controlled Process: Patient

Control Action: App → PCA Pump

Inadequate Operation: Pumps Normally

Feedback: PulseOx → App

Inadequate Feedback: Sends bad SpO₂
Where should we start?

Control Action: App -> PCA Pump

Sensor: Pulse Oximeter

Actuator: PCA Pump

Controller: App Logic

Feedback Message: PulseOx -> App

Controlled Process: Patient

A control action is provided in an unsafe way

How would the control action be unsafe?
What constraint would be violated?
What should the occurrence be named?
What would cause this to occur?
How can this occurrence be compensated for?
Hazard Analysis

Annotating our Architectural Model

How would the control action be unsafe?
What constraint would be violated?
What should the occurrence be named?
What would cause this to occur?
How can this occurrence be compensated for?

We’ll come back to this one in a moment
Report Generation Development

- Development of component architecture using AADL / OSATE2
- Addition of Hazard Analysis Annotations
- Automatic generation of STPA-Styled Hazard Analysis Report
- Very strong traceability between system and HA report

Automatic Report Generation

Fundamentals

Accident Levels

1. **AL1**: Death or serious injury to a human

Accidents

1. **A1**: Patient is killed or seriously injured. [AL1]

Hazards

1. **H1**: Commands for dosage exceeding the patient’s tolerance are sent to the pump. [A1]
2. **H2**: Incorrect information is sent to the display. [A1]

Safety Constraints

1. **C3**: The app must inform the display of the pump command status. [H2]
2. **C1**: The app must command the pump to stop if the patient’s vital signs indicate over-infusion. [H1]
3. **C2**: The app must inform the display of the status of the patient’s vital signs. [H2]
## Unsafe Control Action Table

<table>
<thead>
<tr>
<th>CONTROL ACTION</th>
<th>PROVIDING</th>
<th>NOT PROVIDING</th>
<th>APPLIED TOO LONG</th>
<th>STOPPED TOO SOON</th>
<th>EARLY</th>
<th>LATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>spo2_disp</td>
<td>H2 (Wrong Values (Undetected))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pulseox_fail_disp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>etc02_logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pumpcommand_disp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>respiratory_rate_logic</td>
<td>H1 (Wrong values (Detected)), H1 (Wrong values (Detection Dropped))</td>
<td></td>
<td>H1 (Network Drop)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>capnograph_fail_logic</td>
<td></td>
<td>H1 (Device Alarm Unsent)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spo2_logic</td>
<td>H1 (Wrong values (Detected)), H1 (Wrong values (Detection Dropped))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pulseox_fail_logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pumpcommand_logic</td>
<td>H1 (High Physio Params)</td>
<td></td>
<td>H1 (Network Drop)</td>
<td>H1 (Software Error)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>etc02_disp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>respiratory_rate_disp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>capnograph_fail_logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Background
- Hazard Analysis In AADL
- Architectural Integration
  - EM Fault Types
  - Deeply Integrated Hazard Analysis
  - Tool Support
Annotated Control Loop

Controlled Process

1. Control input or external information wrong or missing

2. Inadequate Control Algorithm
   (Flaws in creation, process changes, incorrect modification or adaptation)

3. Process Model inconsistent, incomplete, or incorrect
   Inadequate or missing feedback
   Feedback Delays
   Incorrect or no information provided
   Measurement inaccuracies
   Feedback delays

4. Component failures
   Changes over time
   Unidentified or out-of-range disturbance
   Process output contributes to system hazard

Controller

Actuator

Sensor

Controller 2

Inappropriate, ineffective or missing control action

Delayed operation

Nancy Leveson. Figure 4.8, Page 93, *Engineering A Safer World*. MIT Press, 2011
## AADL EM Fault Types

### Type Hierarchy

<table>
<thead>
<tr>
<th>Error Library Type</th>
<th>STPA Error Type</th>
<th>App Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors with Physiological Monitors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LateDelivery</td>
<td>DelayedOperation</td>
<td>SpO2ValueLate</td>
</tr>
<tr>
<td>IncorrectValue</td>
<td>IncorrectInformation</td>
<td>SpO2ValueLow</td>
</tr>
<tr>
<td>N/A</td>
<td>NoInformation</td>
<td>NoSpO2Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Errors with App Logic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ServiceCommission</td>
<td>InappropriateCtrlAction</td>
<td>InadvertentPumpNormally</td>
</tr>
<tr>
<td>ServiceOmission</td>
<td>MissingCtrlAction</td>
<td>InadvertentPumpMinimally</td>
</tr>
</tbody>
</table>

### AADL Standard Error Types

- LateDelivery
- IncorrectValue
- N/A

### STPA Guidewords

- DelayedOperation
- IncorrectInformation
- NoInformation

### App Specific Error Types

- SpO2ValueLate
- SpO2ValueLow
- NoSpO2Data
package PCA_Shutoff_Errors
public
with MAP_Errors, PCA_Shutoff_ErrorProperties, MAP_Errors
    PCA_Shutoff;

annex EMV2
{**
    error types
    InadvertentPumpNormally : type extends MAP_Errors::InappropriateControlAction;

    -- Could also be inadequate feedback
    Sp02ValueHigh : type extends MAP_Errors::InadequateSensorOperation;
    Sp02ValueLow : type extends MAP_Errors::InadequateSensorOperation;
    ETCO2ValueLow : type extends MAP_Errors::InadequateSensorOperation;
    ETCO2ValueHigh : type extends MAP_Errors::InadequateSensorOperation;
    RespiratoryRateLow : type extends MAP_Errors::InadequateSensorOperation;
    RespiratoryRateHigh : type extends MAP_Errors::InadequateSensorOperation;
    DeviceAlarmFailsOn : type extends MAP_Errors::InadequateSensorOperation;
    DeviceAlarmFailsOff : type extends MAP_Errors::InadequateSensorOperation;

    end types;

    **};

end PCA_Shutoff_Errors;
STPA in AADL

Using our fault type

Control Action: App → PCA Pump

Feedback Message: PulseOx → App

Controller: App Logic

Actuator: PCA Pump

Sensor: Pulse Oximeter

Controlled Process: Patient

Inadvertent Pump Normally
Integrated Hazard Analysis

Using our fault type

```
package PCA_Interlock_System
public

system PCA_Interlock_System
end PCA_Interlock_System;

system implementation PCA_Interlock_System.imp
subcomponents
    pulseOx : device PulseOx_Interface::MAP_PulseOx_Interface.imp;
pcaPump : device PCAPump_Interface::MAP_PCAPump_Interface.imp;
appLogic : process PCA_Interlock_Logic::PCA_Interlock_Logic.imp;
connections
    spo2_data : port pulseOx.SpO2 -> appLogic.SpO2;
pump_cmd : port appLogic.pumpCmd -> pcaPump.cmd;
annex EMV2 {**
    use types PCA_Interlock_Errors;
    properties
        MAP_Error_Properties::Occurrence => [
            Guideword => Providing;
            ViolatedConstraint => PCA_ShutOff_Error_Properties::DontLetPumpRunWhenUnsafe;
            Title => "High Physio Params";
            ErrorType => reference(InadvertentPumpNormally);
            Description => "One or more physiological parameters are too high, leading the app logic to
                incorrectly believe the patient is healthy";
            Compensation => "Physiological values are cross-checked with other";
        ] applies to pump_cmd;
    **};
end PCA_Interlock_System.imp;
end PCA_Interlock_System;
```

What specific fault will result?

What can we do with our model + specific fault information?
Where would the bad control action come from?

Controller: App Logic

Process Model Incorrect: Wrongly believes patient to be healthy

Control Action: App → PCA Pump

Feedback Message: PulseOx → App

Actuator: PCA Pump

Sensor: Pulse Oximeter

Controlled Process: Patient

Propagates error out
Integrated Hazard Analysis

Specification Step 1: Out Propagation

package PCA_Shutoff_Logic
public
with PCA_Shutoff_Types, PCA_Shutoff_Properties, MAP_Properties;

process ICEpcaShutoffProcess
features
  SpO2 : in event data port PCA_Shutoff_Types::SpO2;
  CommandPumpNormal : out event data port PCA_Shutoff_Types::PumpNormalCommand;
properties
  MAP_Properties::Component_Type => logic;
annex EMV2 {**
  use types PCA_Shutoff_Errors;
  error propagations
    SpO2 : in propagation {SpO2ValueHigh};
    CommandPumpNormal : out propagation {InadvertentPumpNormally};
  flows
    HighSpO2LeadsToD competitive path SpO2{SpO2ValueHigh} -> CommandPumpNormal{InadvertentPumpNormally};
  **};
end ICEpcaShutoffProcess;
-- Process implementation redacted
end PCA_Shutoff_Logic;
Where would the bad control action come from?

Control Action: App -> PCA Pump

Feedback Message: PulseOx -> App

Controller: App Logic

Process Model Incorrect: Wrongly believes patient to be healthy

Actuator: PCA Pump

Sensor: Pulse Oximeter

Controlled Process: Patient

Bad information in
package PCA_Shutoff_logic;
public
with PCA_Shutoff_Types, PCA_Shutoff_Properties, MAP_Properties;

process ICEpcaShutoffProcess
features
    SpO2 : in event data port PCA_Shutoff_Types::SpO2;
    CommandPumpNormal : out event data port PCA_Shutoff_Types::PumpNormalCommand;
properties
    MAP_Properties::Component_Type => logic;
annex EMV2 {
    use types PCA_Shutoff_Errors;
    error propagations
        SpO2 : in propagation {SpO2ValueHigh};
        CommandPumpNormal : out propagation {InadvertentPumpNormally};
    flows
        HighSpO2LeadsToOD : error path SpO2{SpO2ValueHigh} -> CommandPumpNormal{InadvertentPumpNormally};
    **);
end propagations;
end ICEpcaShutoffProcess;

-- Process implementation redacted
end PCA_Shutoff_Lo
Integrated Hazard Analysis

Specification Step 3: Relation between incoming and outgoing

```plaintext
package PCA_Shutoff_Look
public
with PCA_Shutoff_Types, PCA_Shutoff_Errors, MAP_Properties;

process ICEpcaShutoffProcess
features
  SpO2 : in event data port PCA_Shutoff_Types::SpO2;
  CommandPumpNormal : out event data port PCA_Shutoff_Types::PumpNormal;
properties
  MAP_Properties::Component_Type =>
  annex EMV2 {**
    use types PCA_Shutoff_Errors;
    error propagations
      SpO2 : in propagation {SpO2ValueHigh};
      CommandPumpNormal : out propagation {InadvertentPumpNormal};
    flows
      HighSpO2LeadsToOD : error path SpO2{SpO2ValueHigh} -> CommandPumpNormal{InadvertentPumpNormal};
    **};
  end propagations;
**};
end ICEpcaShutoffProcess;

-- Process implementation redacted
end PCA_Shutoff_Look;
```
STPA in AADL

Where should we go now?

Control Action: App → PCA Pump

Feedback Message: PulseOx → App

Controller: App Logic

Process Model Incorrect: Wrongly believes patient to be healthy

Option 1: Look for the source

Option 2: Look for the impact

Actuator: PCA Pump

Sensor: Pulse Oximeter

Controlled Process: Patient
STPA in AADL

Where should we go now?

Option 3: Look for other sources / impacts

App Logic -> Display

PCA Pump

Pulse Oximeter

Clinician

Patient
Integrated Hazard Analysis

OSATE Remembers A Neglected Connection

```
system implementation PCA_Shutoff_System.imp
subcomponents
  -- Physiological inputs
  pulseOx : device PulseOx_Interface::ICEpoInterface.imp;

  -- App logic
  appLogic : process PCA_Shutoff_Loic::ICEpcaShutoffProcess.imp;
  appDisplay : process PCA_Shutoff_Display::ICEpcaDisplayProcess.imp;

connections
  -- From components to logic
  spo2_logic : port pulseOx.SpO2 -> appLogic.SpO2;

  -- From components to display
  spo2_disp : port pulseOx.SpO2 -> appDisplay.SpO2;

anne
  └─ No incoming error propagation from appDisplay for outgoing propagation SpO2(SpO2ValueHigh). Check for Unhandled Faults.
```

```
-- Errors between the PulseOx's SpO2 channel and the App Logic
MAP_Error_Properties::Occurrence => [
  Kind => ValueHigh;
  Hazard => PCA_Shutoff_Error_Properties::PatientHarmed;
  ViolatedConstraint => PCA_Shutoff_Error_Properties::PumpWhenSafe;
  Title => "Wrong Values (Undetected)"
  Cause => "Incorrect values are gathered from the physiological sensors"
]```
Tool Supported Process

Interaction between Report and Model

1. Here’s an empty cell (STPA Keyword + Control Action)... could anything go wrong?

2. Create occurrence and supporting EM annotations

3. Where else could this fault go?

4. What else could cause this error?
Further Reading

- Source available online at https://github.com/santoslab/aadl-translator
- Installable into OSATE2 via update site: http://santoslab.org/pub/mdcf-architect/updatesite
- Full documentation online at http://santoslab.org/pub/mdcf-architect
- Publications online at http://people.cis.ksu.edu/~samprocter
Using STPA to Support Risk Management for Interoperable Medical Systems

STAMP Workshop 2015, MIT

**Sam Procter**, John Hatcliff  
SAnToS Lab  
Kansas State University

Anura Fernando  
Underwriters Laboratories

Sandy Weininger  
US Food and Drug Administration

**Support:**

This work is supported in part by the US National Science Foundation (NSF) (#1239543), the NSF US Food and Drug Administration Scholar-in-Residence Program (#1355778) and the National Institutes of Health / NIBIB Quantum Program.
Doesn’t use of AADL imply a fully specified architecture?

No. Though some architectural constraints are implied by the domain (e.g., component-based architecture, use of underlying middleware for communication, etc.), architectures in AADL can be rapidly modified. Constructing (or modeling) an architecture in AADL is very much a “design phase” task.
How can apps be certified independently of their environment?

- Much the same way that medical devices are currently certified under some set of assumptions (collectively referred to as intended use), we imagine that MAP apps will have (contra)indications for use.
- There are requirements engineering issues to be addressed, this is a key part of the UL 2800 standardization effort.
What about interactions between devices / apps that are not over input or output ports?

- We rely heavily on a notion of platform to isolate components from one another. This platform technology, developed by our King et al at UPenn, aims to provide complete separation between components (similar to separation kernels / partitioning middleware used in avionics).

- AADL can also model unintended / indirect interactions, like heat.