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

Sam Procter sprocter@sei.cmu.edu

Jérôme Hugues jjhugues@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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

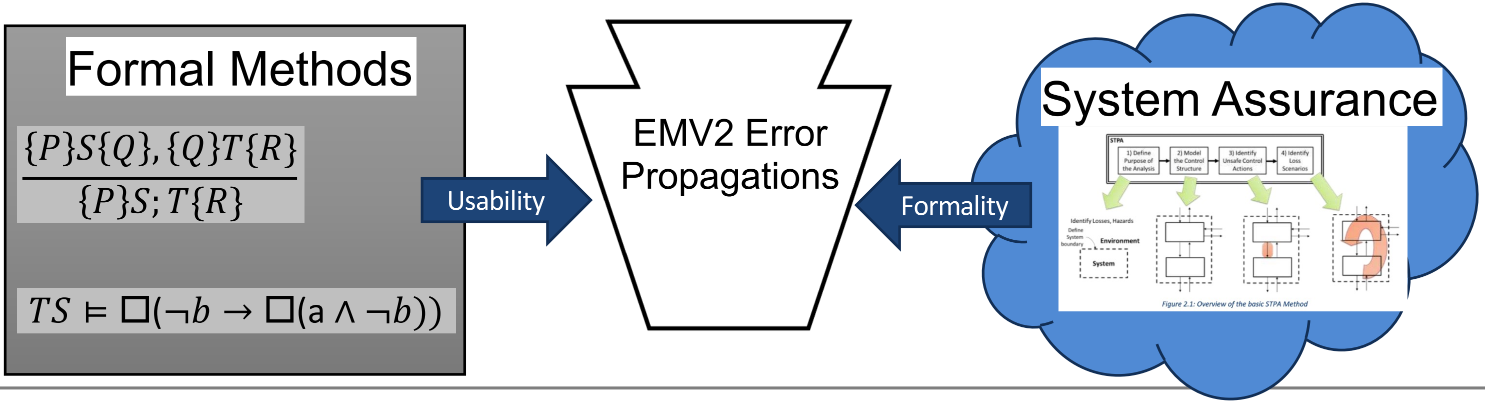
Challenge 3: Assurance evidence should be modular and composable

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.


Formal Methods
$\frac{\{P\}S\{Q\}, \{Q\}T\{R\}}{\{P\}S;T\{R\}}$
$TS \vDash \Box(\neg b \rightarrow \Box(a \wedge \neg b))$
Usability
EMV2 Error Propagations
Formality
System Assurance
STPA
1) Define Purpose of the Analysis
2) Model the Control Structure
3) Identify Unsafe Control Actions
4) Identify Loss Scenarios
Identify Losses, Hazards
Define System boundary
Environment
System
Figure 2.1: Overview of the basic STPA Method

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

STPA & SAFE

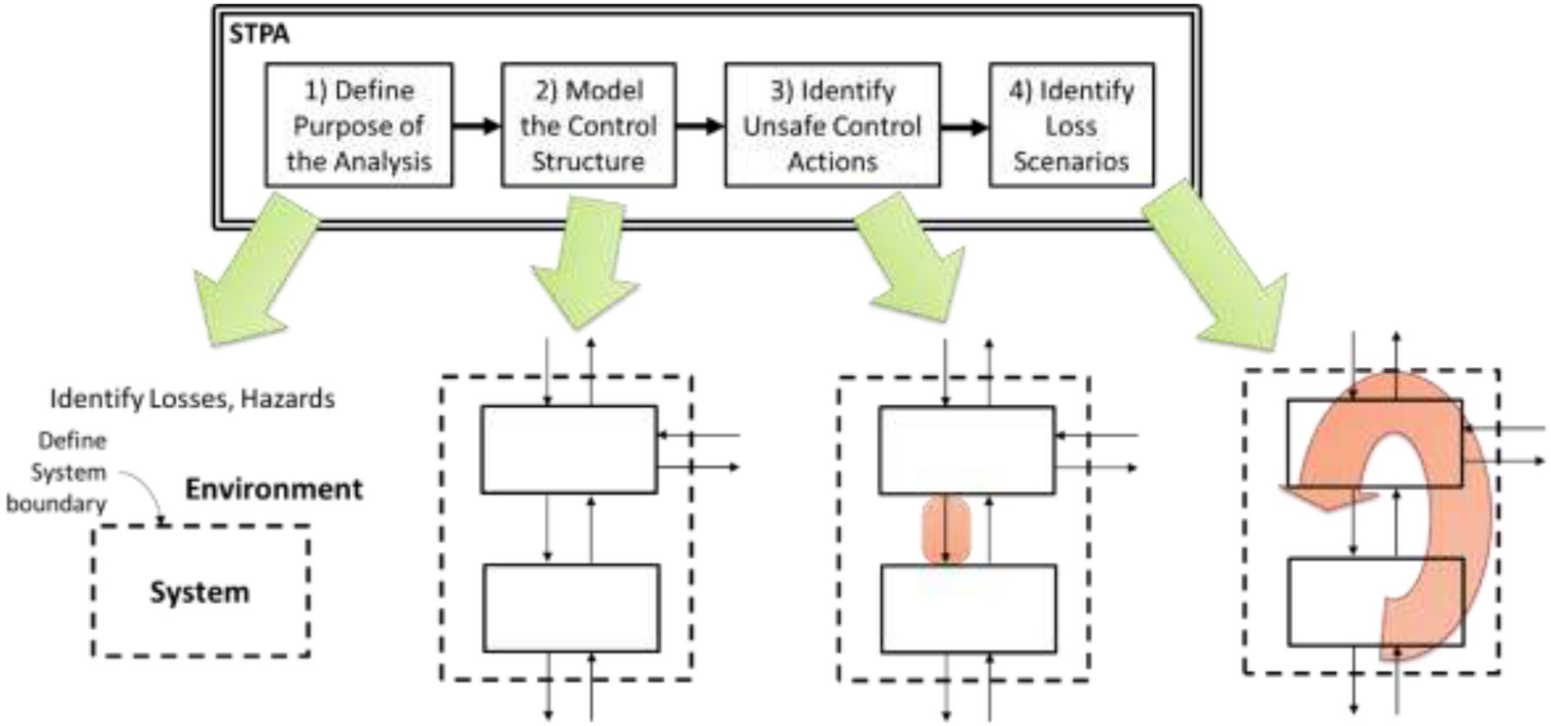
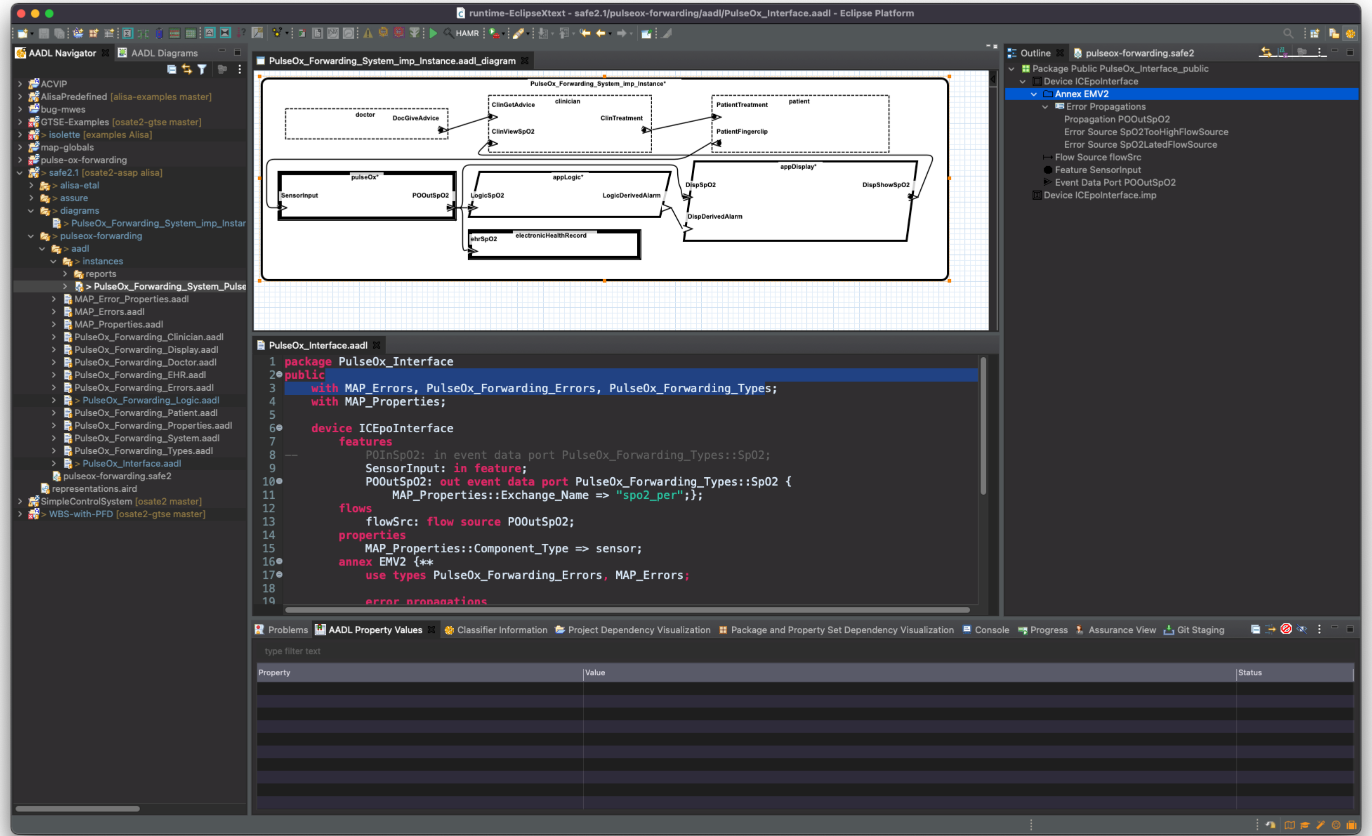

STPA
1) Define Purpose of the Analysis
2) Model the Control Structure
3) Identify Unsafe Control Actions
4) Identify Loss Scenarios
Identify Losses, Hazards
Define System boundary
Environment
System

Figure 2.1: Overview of the basic STPA Method

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

AADL & OSATE



```
> ACVIP
> AlisaPredefined [alisa-examples master]
> bug-mwes
> GTSE-Examples [osate2-gtse master]
> > isolette [examples Alisa]
> map-globals
> pulse-ox-forwarding
v > safe2.1 [osate2-asap alisa]
    > > alisa-etal
    > > assure
    v > diagrams
          > PulseOx_Forwarding_System_imp_Instar
    v > pulseox-forwarding
        v > aadl
            v > instances
                > reports
                > > PulseOx_Forwarding_System_Pulse
            > MAP_Error_Properties.aadl
            > MAP_Errors.aadl
            > MAP_Properties.aadl
            > PulseOx_Forwarding_Clinician.aadl
            > PulseOx_Forwarding_Display.aadl
            > PulseOx_Forwarding_Doctor.aadl
            > PulseOx_Forwarding_EHR.aadl
            > PulseOx_Forwarding_Errors.aadl
            > > PulseOx_Forwarding_Logic.aadl
            > PulseOx_Forwarding_Patient.aadl
            > PulseOx_Forwarding_Properties.aadl
            > PulseOx_Forwarding_System.aadl
            > PulseOx_Forwarding_Types.aadl
            > > PulseOx_Interface.aadl
          pulseox-forwarding.safe2
        representations.aird
> SimpleControlSystem [osate2 master]
> > WBS-with-PFD [osate2-gtse master]
```

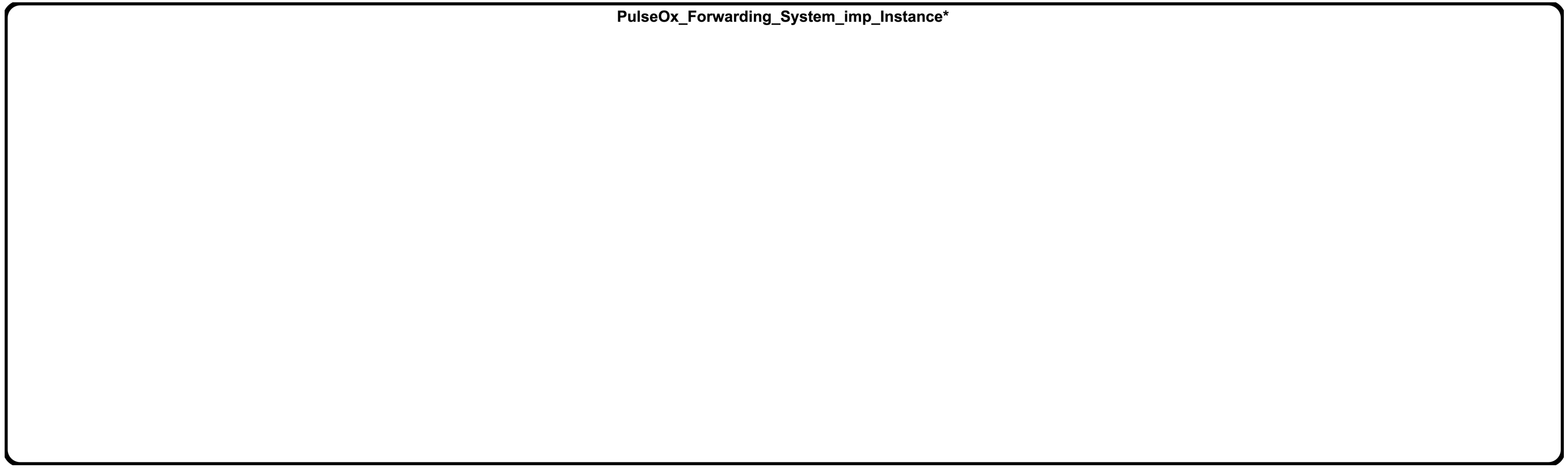
```
package PulseOx_Interface
public
    with MAP_Errors, PulseOx_Forwarding_Errors, PulseOx_Forwarding_Types;
    with MAP_Properties;

    device ICEpoInterface
        features
--          POInSpO2: in event data port PulseOx_Forwarding_Types::SpO2;
            SensorInput: in feature;
            POOutSpO2: out event data port PulseOx_Forwarding_Types::SpO2 {
                MAP_Properties::Exchange_Name => "spo2_per";};
        flows
            flowSrc: flow source POOutSpO2;
        properties
            MAP_Properties::Component_Type => sensor;
        annex EMV2 {**
            use types PulseOx_Forwarding_Errors, MAP_Errors;

            error propagations
```

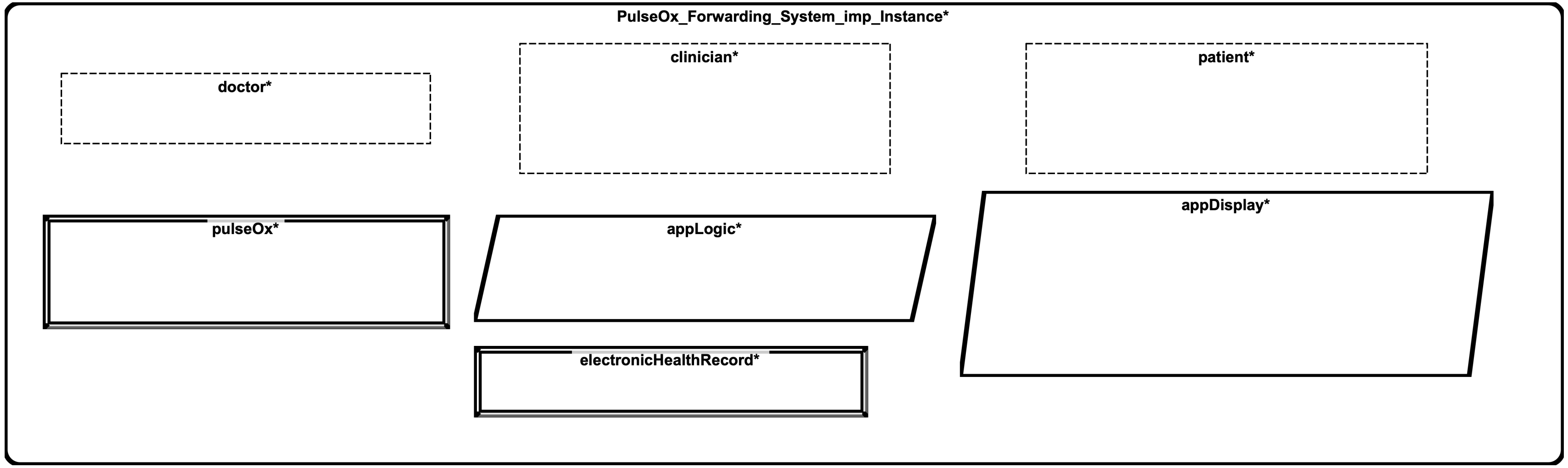
```
v Package Public PulseOx_Interface_public
    v Device ICEpoInterface
        v Annex EMV2
            v Error Propagations
                  Propagation POOutSpO2
                  Error Source SpO2TooHighFlowSource
                  Error Source SpO2LatedFlowSource
            Flow Source flowSrc
            Feature SensorInput
            Event Data Port POOutSpO2
      Device ICEpoInterface.imp
```

The PulseOx Forwarding Example

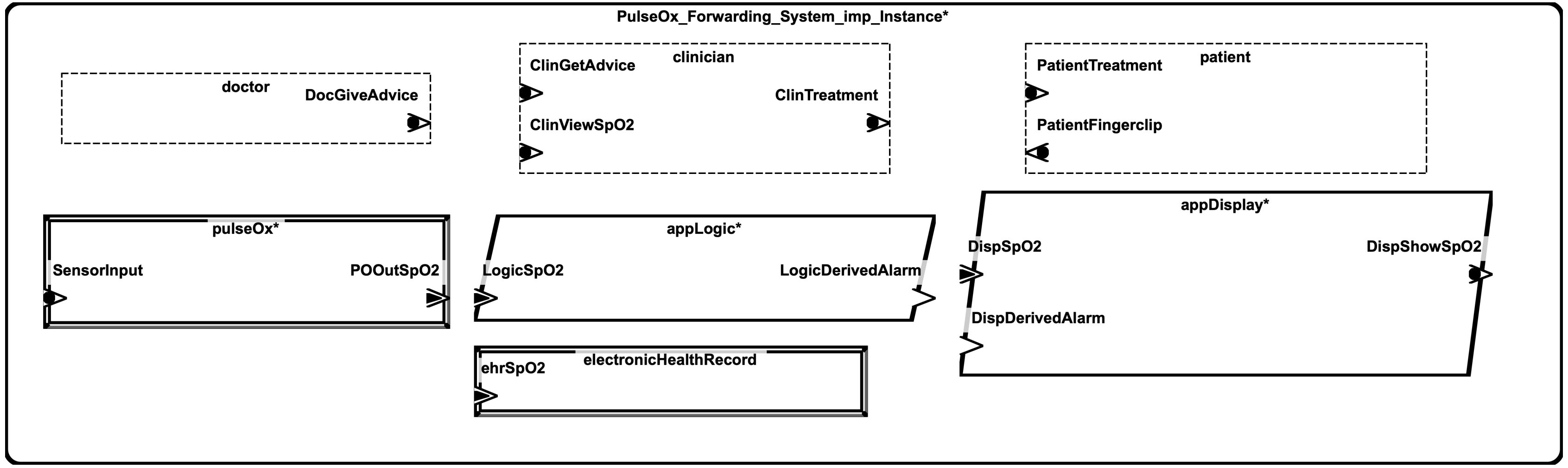


Pulse oximeter reads blood-oxygen saturation from a patient, monitoring software displays an alarm if values are out of expected range

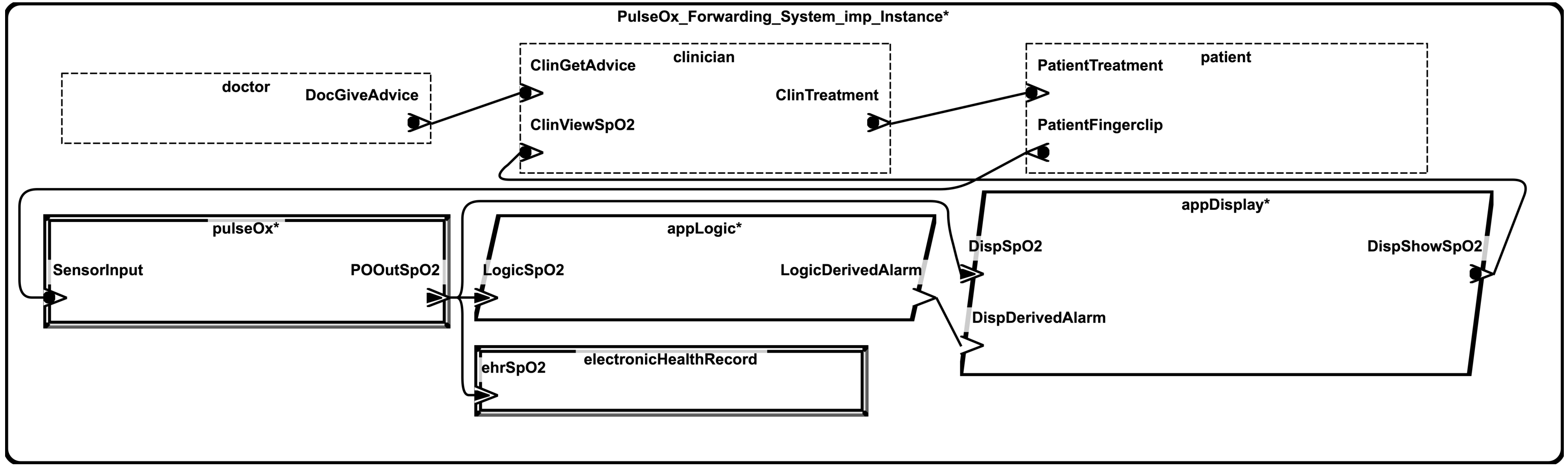
The PulseOx Forwarding Example



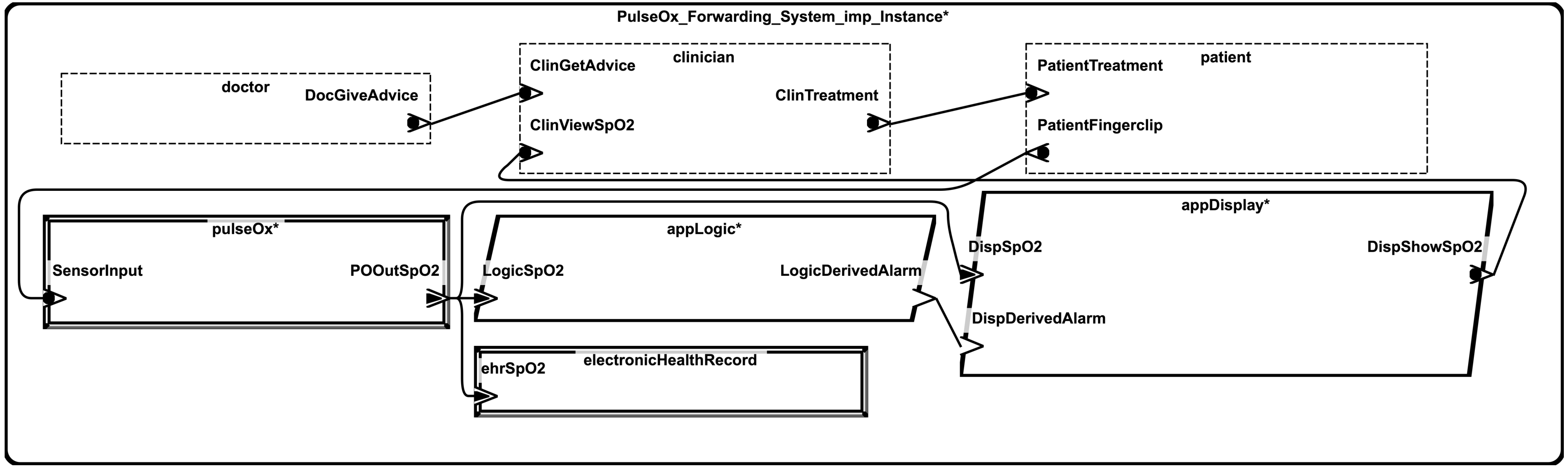
The PulseOx Forwarding Example



The PulseOx Forwarding Example

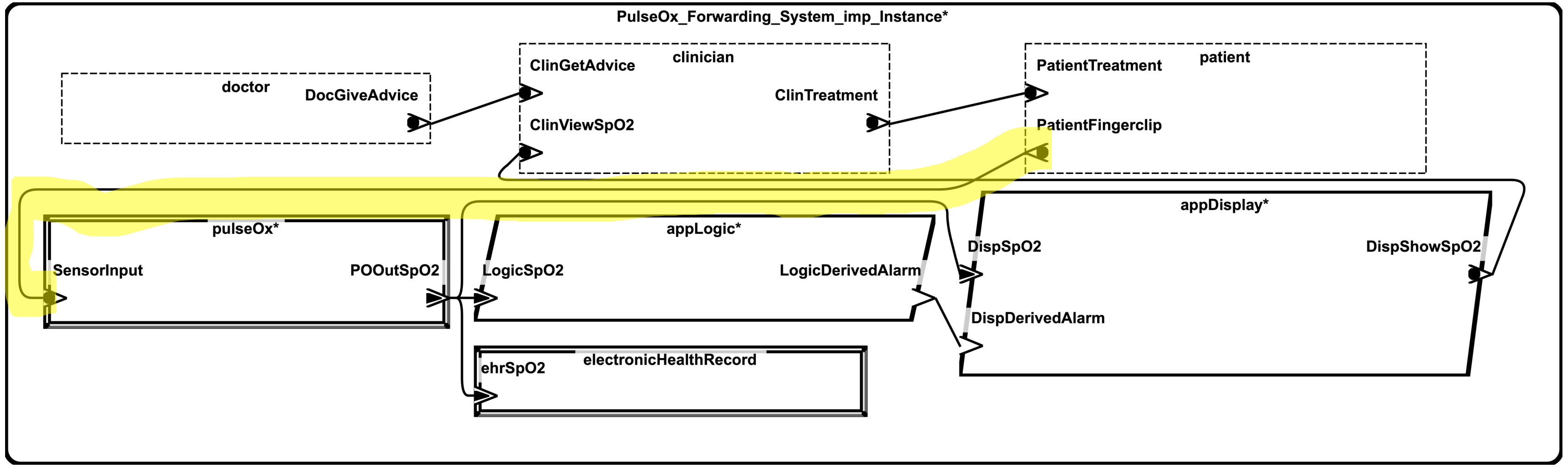


The PulseOx Forwarding Example



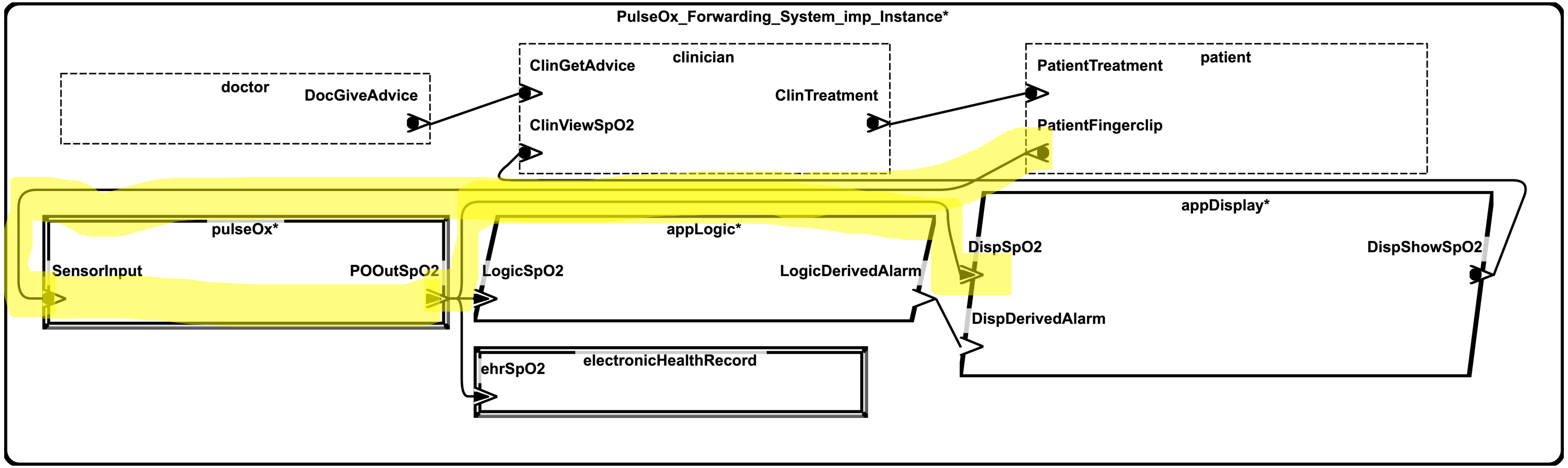
- Safety problem to avoid: Incorrect SpO_2 displayed

The PulseOx Forwarding Example



- Safety problem to avoid: Incorrect SpO_2 displayed

The PulseOx Forwarding Example



- Safety problem to avoid: Incorrect SpO_2 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

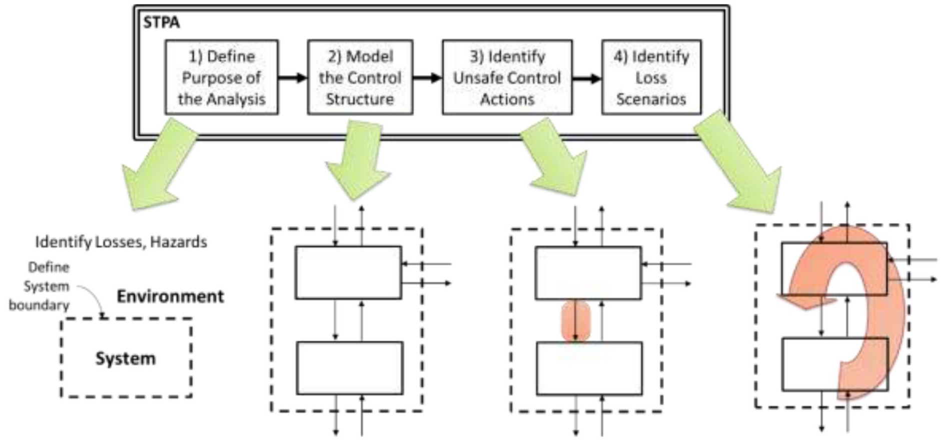

STPA
1) Define Purpose of the Analysis
2) Model the Control Structure
3) Identify Unsafe Control Actions
4) Identify Loss Scenarios
Identify Losses, Hazards
Define System boundary
Environment
System

Figure 2.1: Overview of the basic STPA Method

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

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

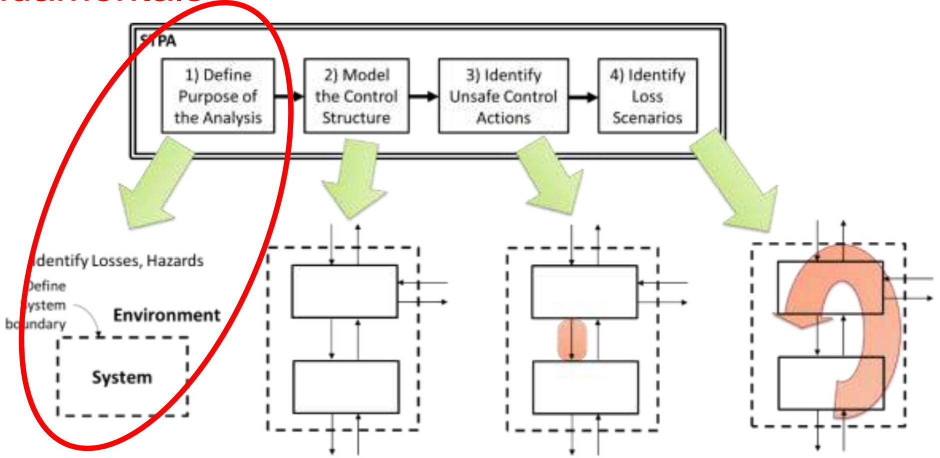

STPA
1) Define Purpose of the Analysis
2) Model the Control Structure
3) Identify Unsafe Control Actions
4) Identify Loss Scenarios
Identify Losses, Hazards
Define System boundary
Environment
System

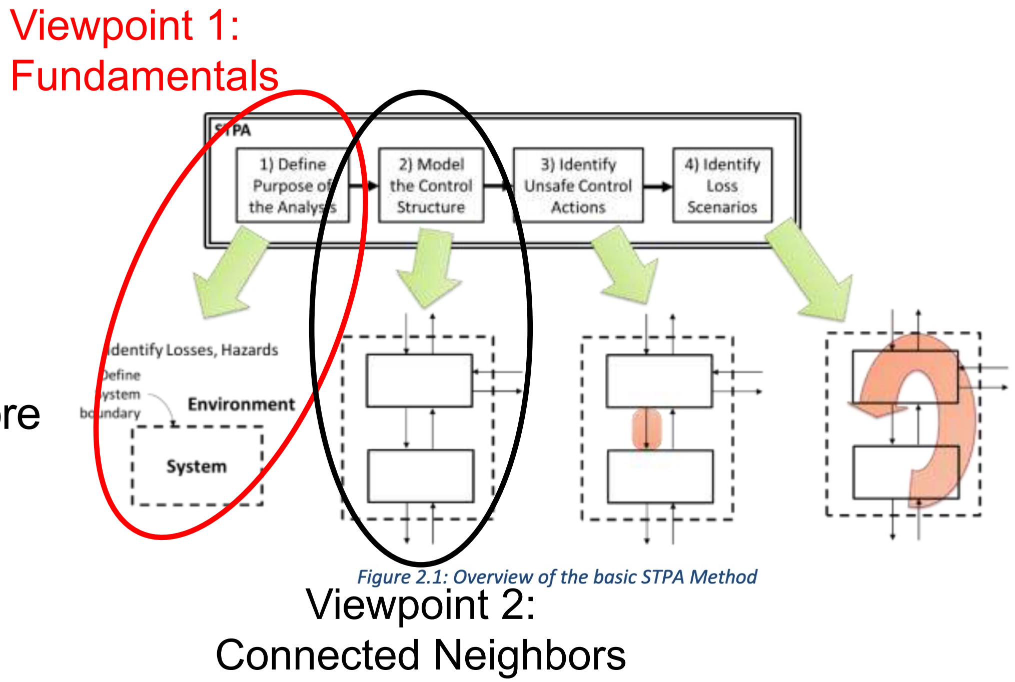
Figure 2.1: Overview of the basic STPA Method

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
[*https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf*](https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf)

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
STPA
1) Define Purpose of the Analysis
2) Model the Control Structure
3) Identify Unsafe Control Actions
4) Identify Loss Scenarios
Identify Losses, Hazards
Define system boundary
Environment
System
Viewpoint 2: Connected Neighbors

Figure 2.1: Overview of the basic STPA Method

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

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

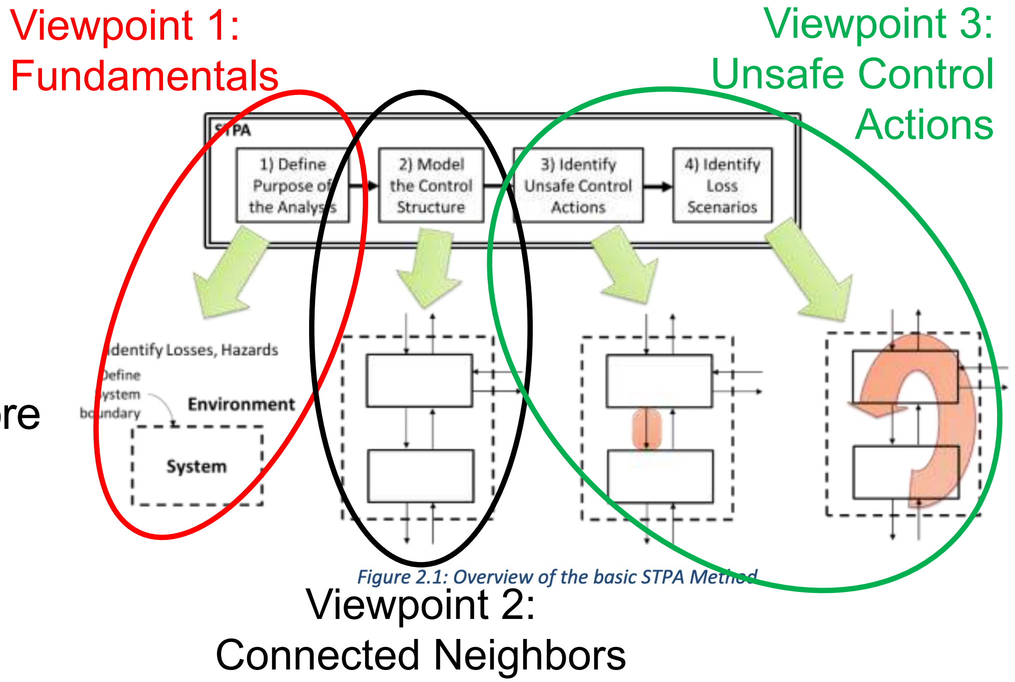
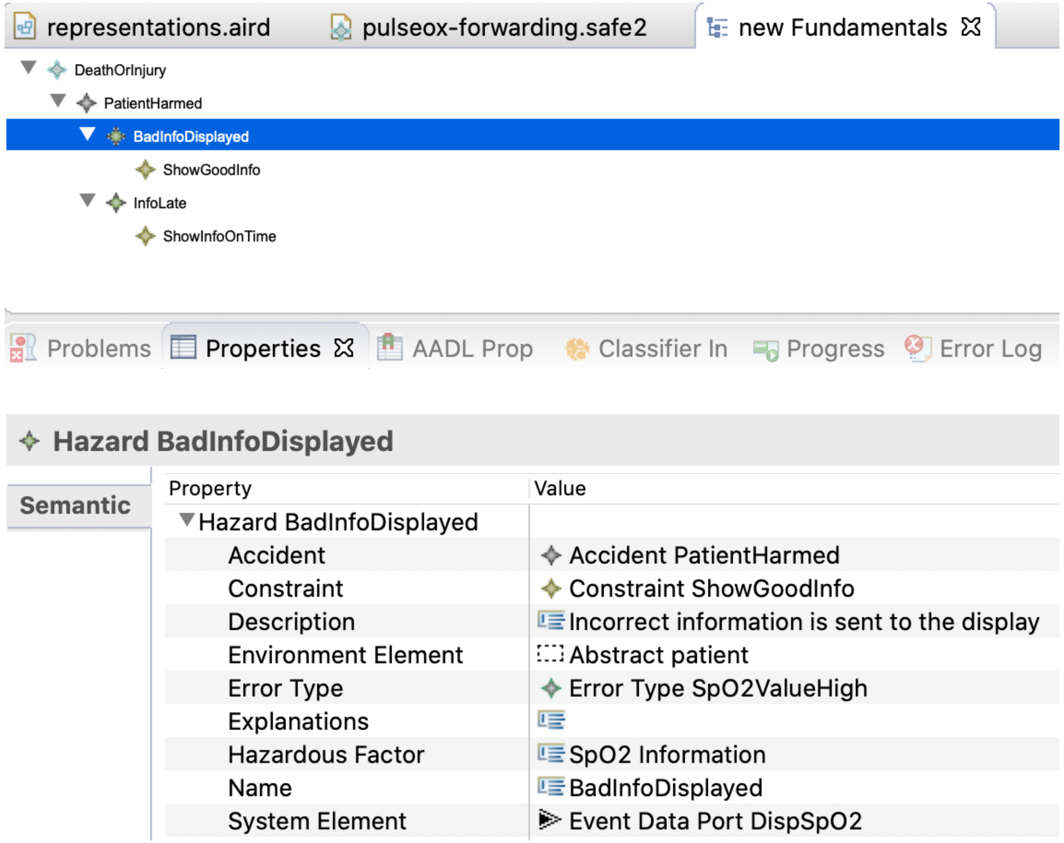

Viewpoint 1: Fundamentals
Viewpoint 3: Unsafe Control Actions
STPA
1) Define Purpose of the Analysis
2) Model the Control Structure
3) Identify Unsafe Control Actions
4) Identify Loss Scenarios
Identify Losses, Hazards
Define system boundary
Environment
System
Viewpoint 2: Connected Neighbors

Figure 2.1: Overview of the basic STPA Method

© John Thomas, Nancy Leveson, STPA Handbook, March 2018
https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

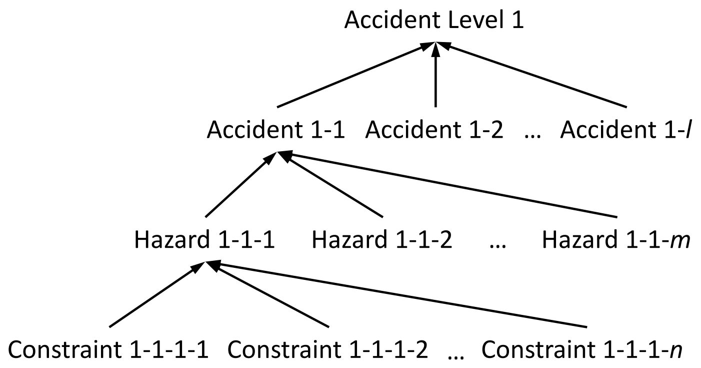
Viewpoint 1: Fundamentals (Hierarchy)



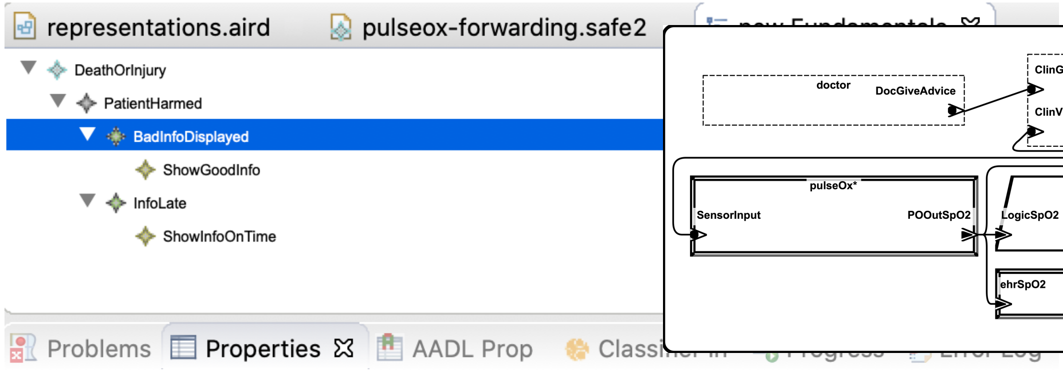
- DeathOrInjury
  - PatientHarmed
    - BadInfoDisplayed
      - ShowGoodInfo
    - InfoLate
      - ShowInfoOnTime

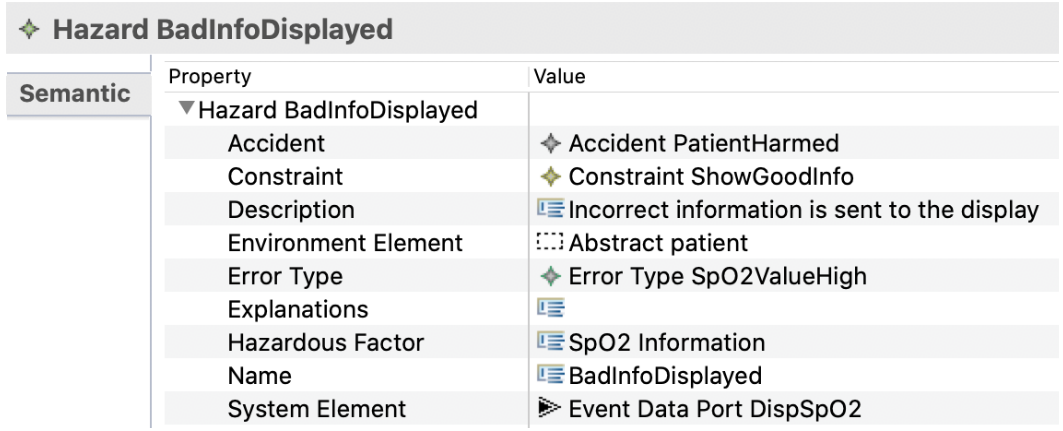
Hazard BadInfoDisplayed

Property	Value
Hazard BadInfoDisplayed	
Accident	Accident PatientHarmed
Constraint	Constraint ShowGoodInfo
Description	Incorrect information is sent to the display
Environment Element	Abstract patient
Error Type	Error Type SpO2ValueHigh
Explanations	
Hazardous Factor	SpO2 Information
Name	BadInfoDisplayed
System Element	Event Data Port DispSpO2



Viewpoint 1: Fundamentals (Link to system)



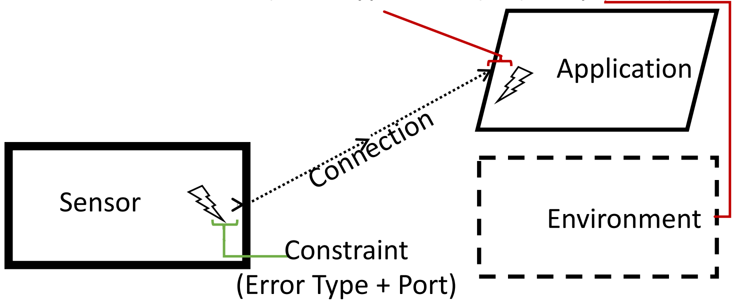


Hazard BadInfoDisplayed

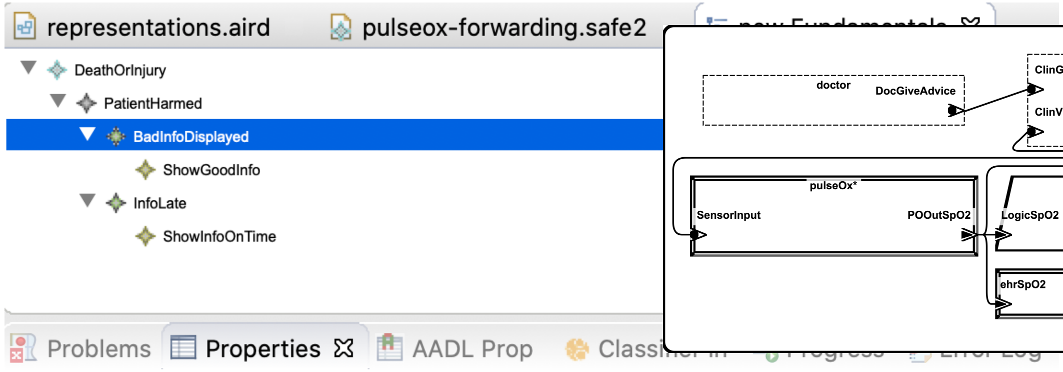
Semantic

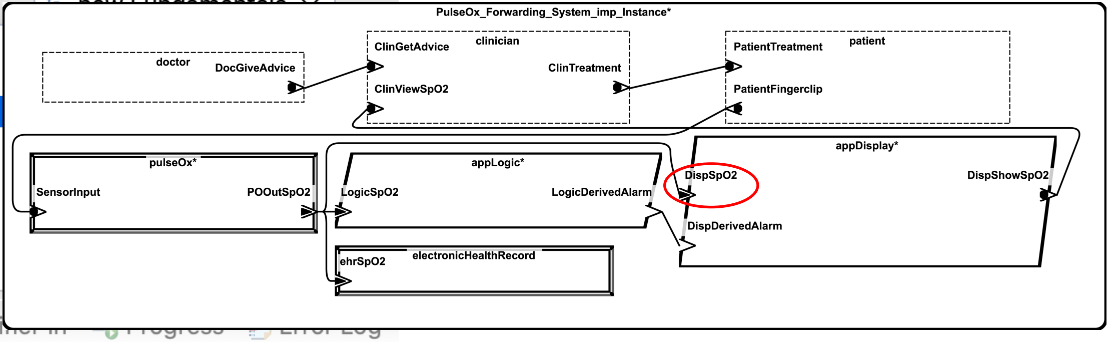
Property	Value
Hazard BadInfoDisplayed	
Accident	Accident PatientHarmed
Constraint	Constraint ShowGoodInfo
Description	Incorrect information is sent to the display
Environment Element	Abstract patient
Error Type	Error Type SpO2ValueHigh
Explanations	
Hazardous Factor	SpO2 Information
Name	BadInfoDisplayed
System Element	Event Data Port DispSpO2

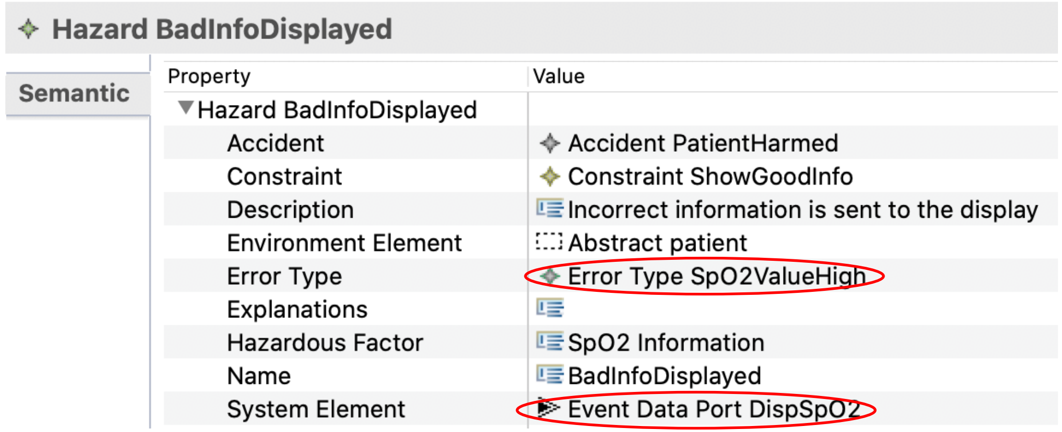
Hazard = System State + Environment State
(Error Type + Port) + (Component)



Linking System Safety to Architecture with EMV2 Error Types



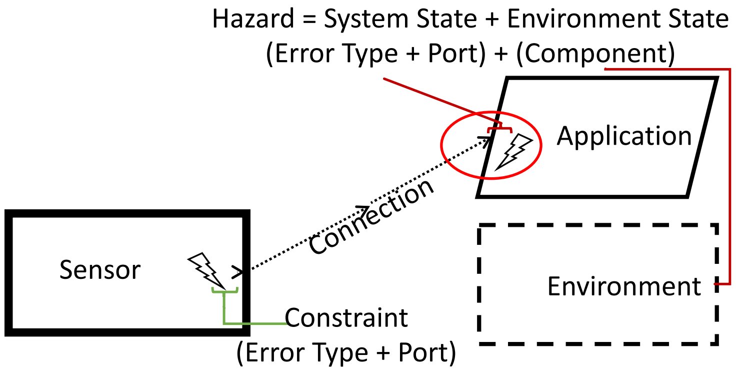




Hazard BadInfoDisplayed

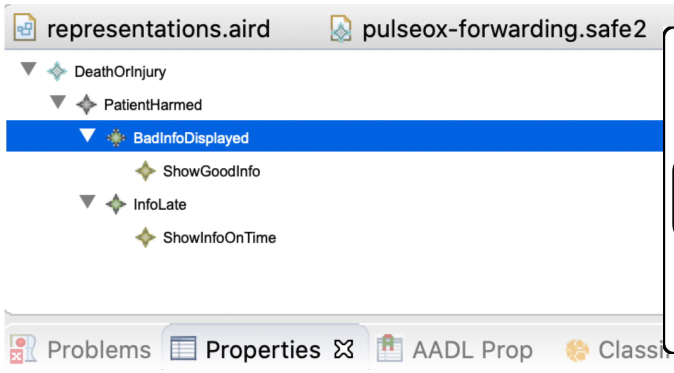
Semantic

Property	Value
Hazard BadInfoDisplayed	
Accident	Accident PatientHarmed
Constraint	Constraint ShowGoodInfo
Description	Incorrect information is sent to the display
Environment Element	Abstract patient
Error Type	Error Type SpO2ValueHigh
Explanations	
Hazardous Factor	SpO2 Information
Name	BadInfoDisplayed
System Element	Event Data Port DispSpO2



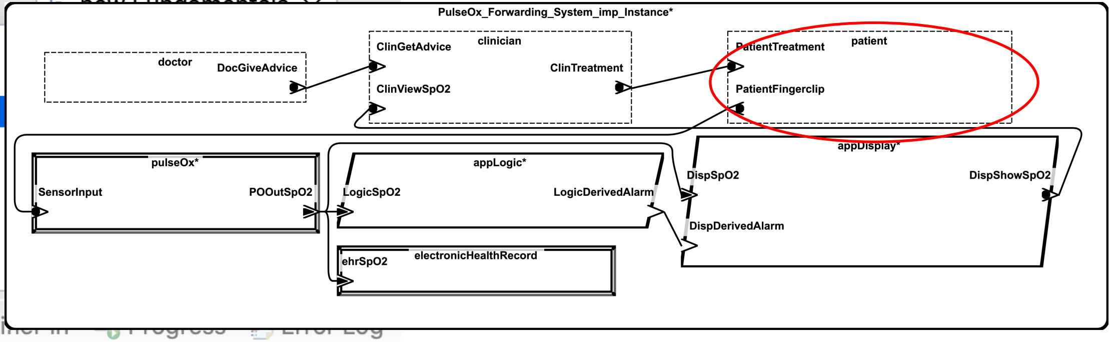
Hazard = System State + Environment State
(Error Type + Port) + (Component)
Application
Connection
Sensor
Environment
Constraint
(Error Type + Port)

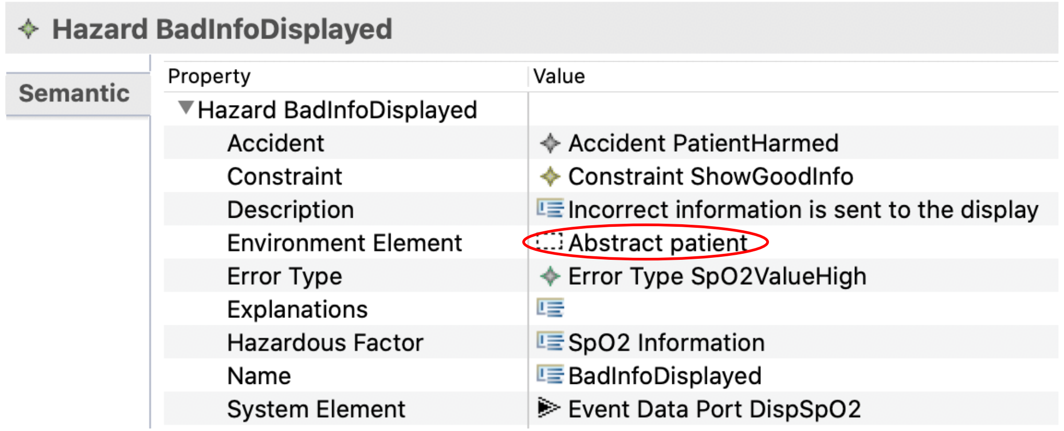
Linking System Safety to Architecture with EMV2 Error Types


representations.aird | pulseox-forwarding.safe2

- DeathOrInjury
  - PatientHarmed
    - BadInfoDisplayed
      - ShowGoodInfo
    - InfoLate
      - ShowInfoOnTime

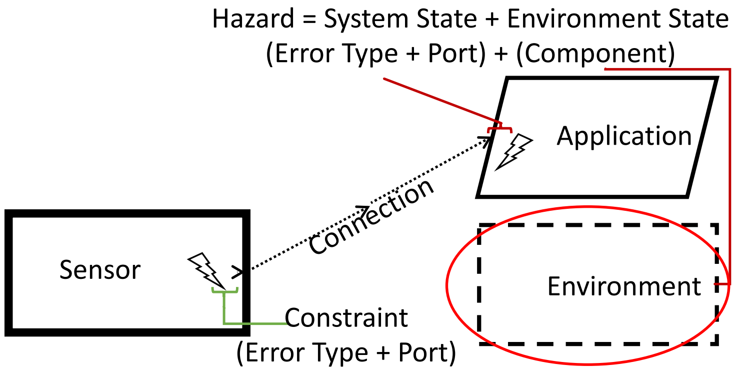
Problems | Properties | AADL Prop | Classi…




Hazard BadInfoDisplayed

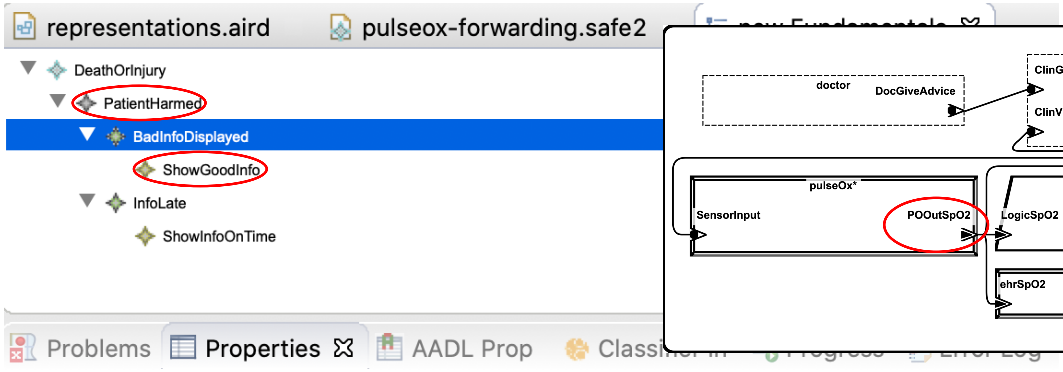
Semantic

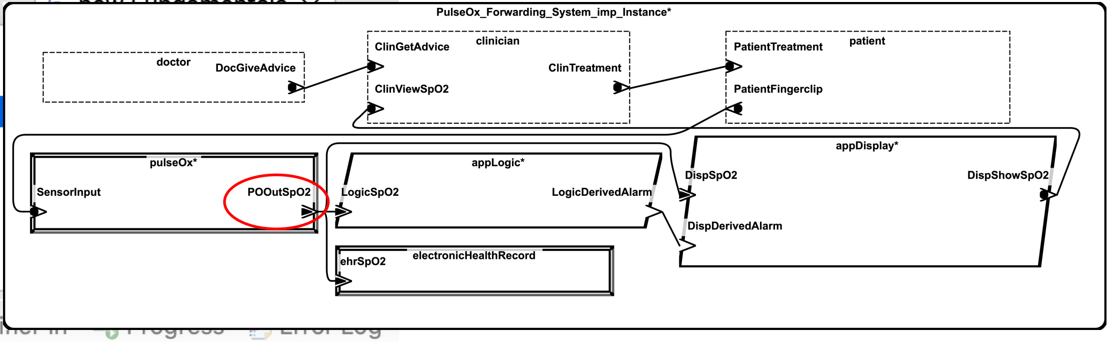
Property	Value
Hazard BadInfoDisplayed	
Accident	Accident PatientHarmed
Constraint	Constraint ShowGoodInfo
Description	Incorrect information is sent to the display
Environment Element	Abstract patient
Error Type	Error Type SpO2ValueHigh
Explanations	
Hazardous Factor	SpO2 Information
Name	BadInfoDisplayed
System Element	Event Data Port DispSpO2

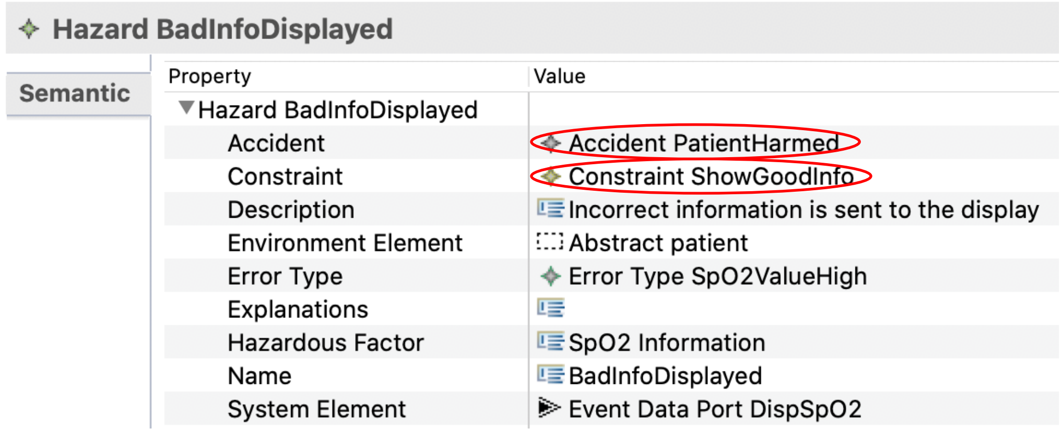


Hazard = System State + Environment State
(Error Type + Port) + (Component)
Application
Connection
Sensor
Environment
Constraint
(Error Type + Port)

Linking System Safety to Architecture with EMV2 Error Types



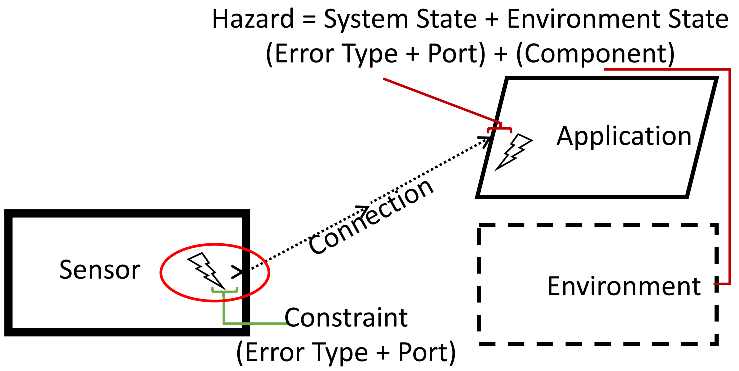




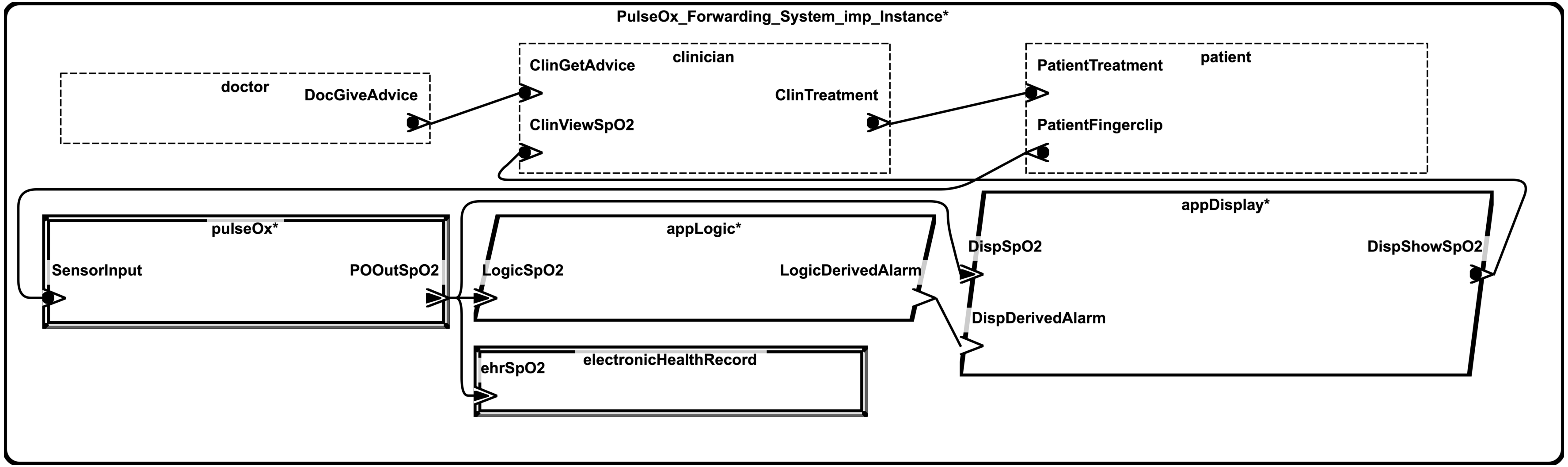
Hazard BadInfoDisplayed

Semantic

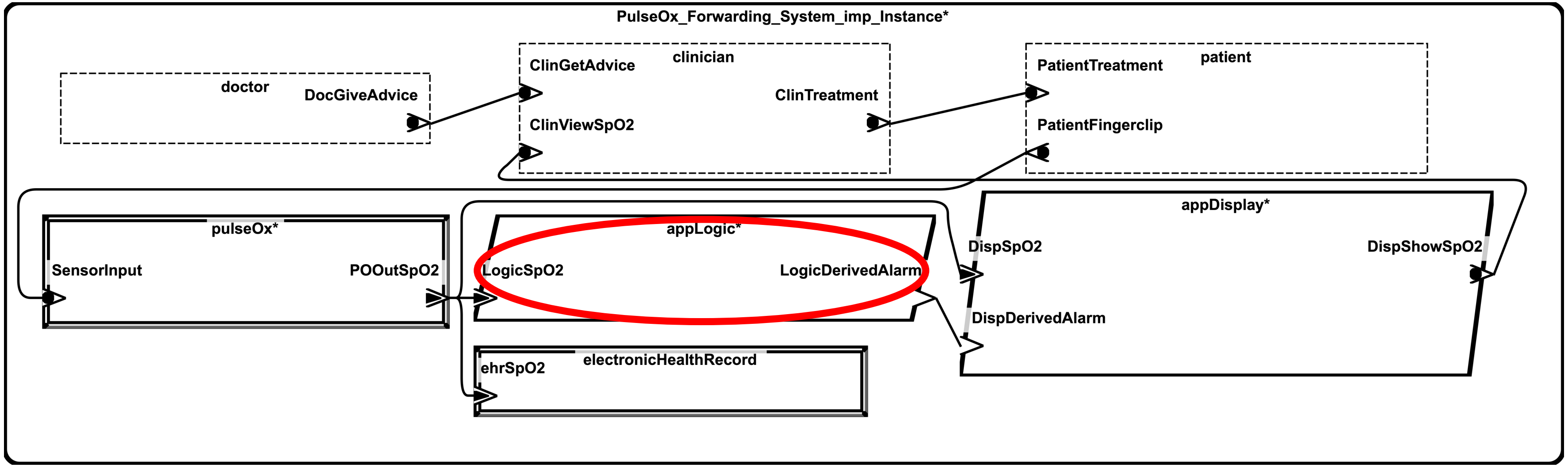
Property	Value
Hazard BadInfoDisplayed	
Accident	Accident PatientHarmed
Constraint	Constraint ShowGoodInfo
Description	Incorrect information is sent to the display
Environment Element	Abstract patient
Error Type	Error Type SpO2ValueHigh
Explanations	
Hazardous Factor	SpO2 Information
Name	BadInfoDisplayed
System Element	Event Data Port DispSpO2


Hazard = System State + Environment State
(Error Type + Port) + (Component)
Application
Connection
Sensor
Environment
Constraint
(Error Type + Port)

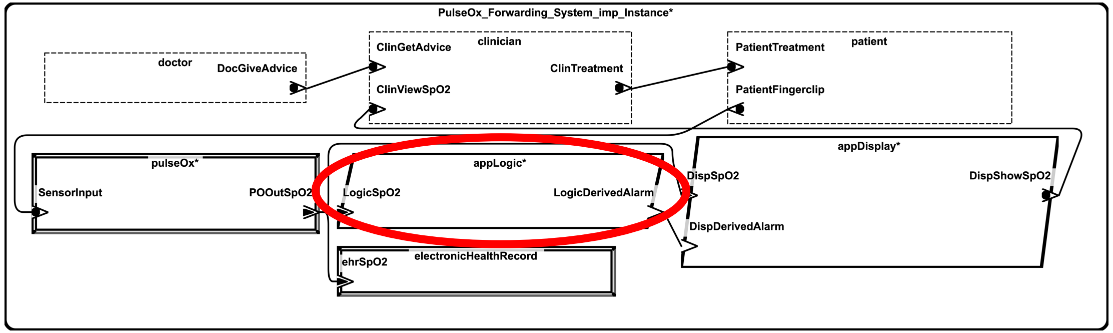
Viewpoint 2: Connected Neighbors

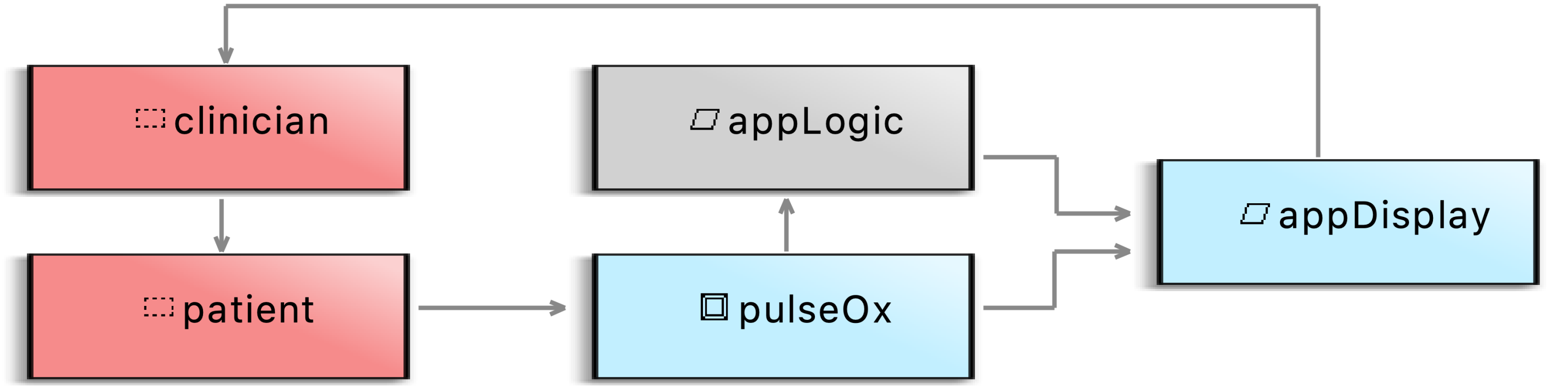


Viewpoint 2: Connected Neighbors

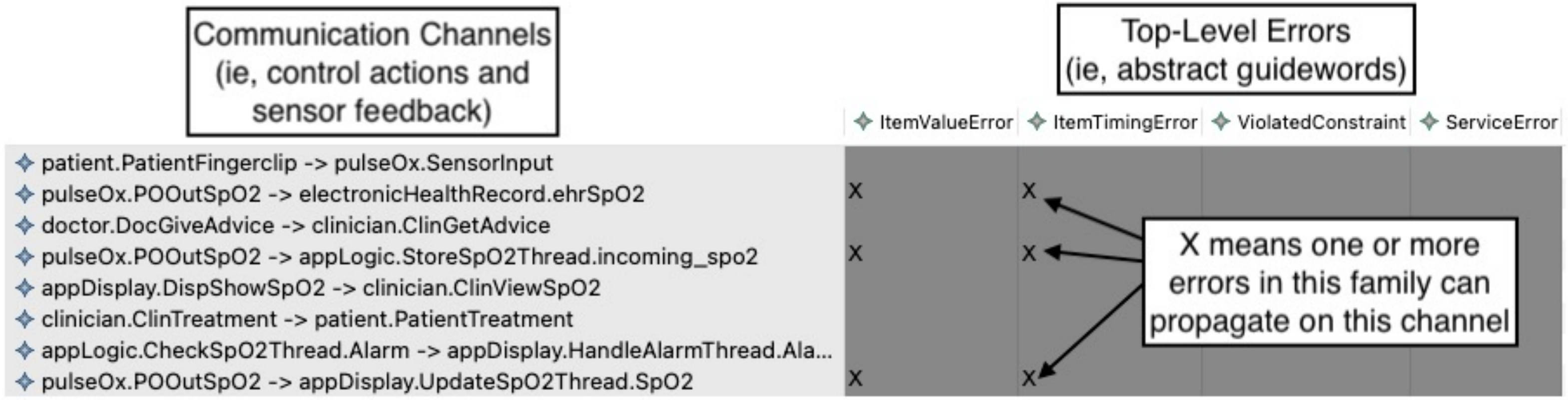


Viewpoint 2: Connected Neighbors



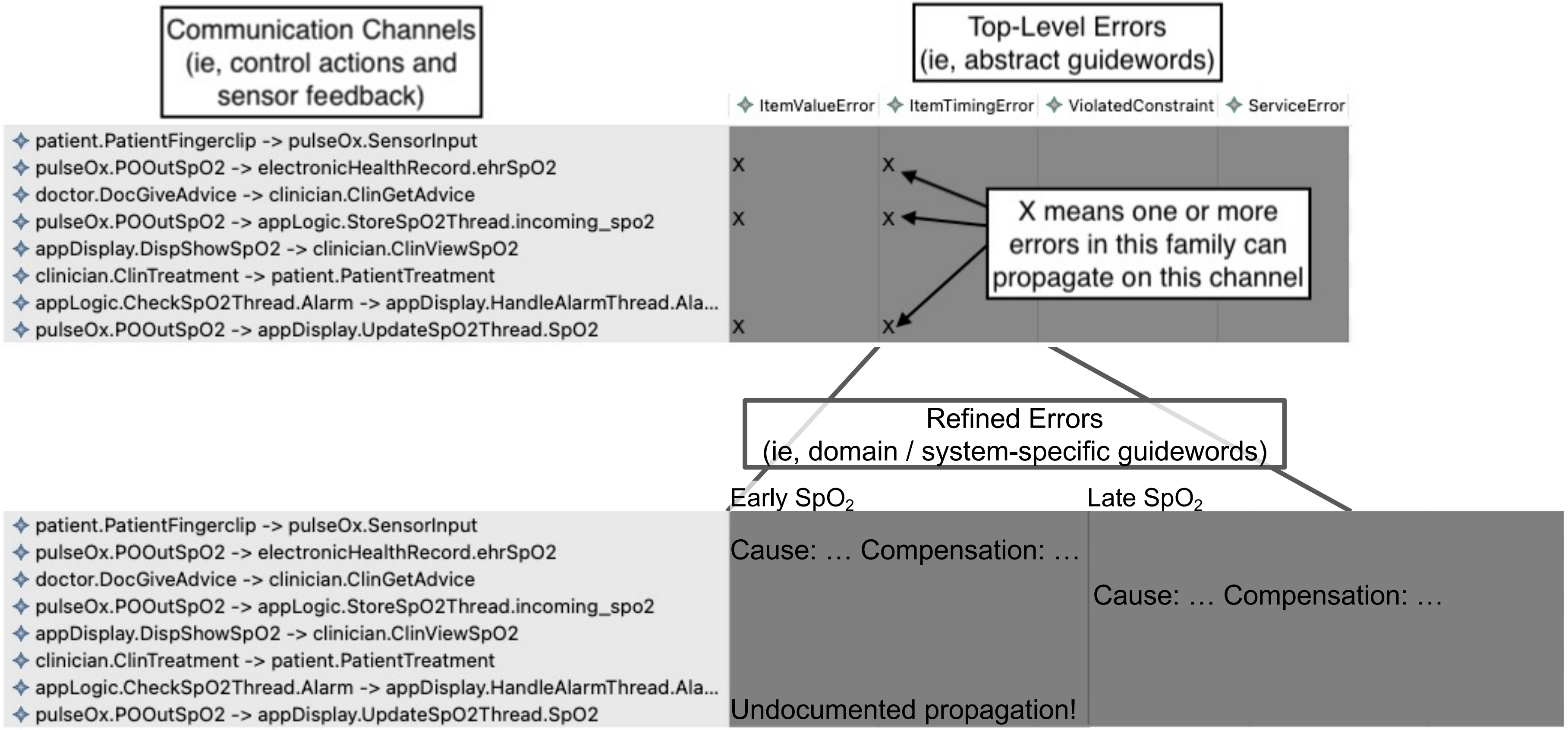


Viewpoint 3: Unsafe Control Actions


Communication Channels (ie, control actions and sensor feedback)
Top-Level Errors (ie, abstract guidewords)
X means one or more errors in this family can propagate on this channel

	ItemValueError	ItemTimingError	ViolatedConstraint	ServiceError
patient.PatientFingerclip -> pulseOx.SensorInput				
pulseOx.POOutSpO2 -> electronicHealthRecord.ehrSpO2	x	x		
doctor.DocGiveAdvice -> clinician.ClinGetAdvice				
pulseOx.POOutSpO2 -> appLogic.StoreSpO2Thread.incoming_spo2	x	x		
appDisplay.DispShowSpO2 -> clinician.ClinViewSpO2				
clinician.ClinTreatment -> patient.PatientTreatment				
appLogic.CheckSpO2Thread.Alarm -> appDisplay.HandleAlarmThread.Ala...				
pulseOx.POOutSpO2 -> appDisplay.UpdateSpO2Thread.SpO2	x	x		

Viewpoint 3: Unsafe Control Actions



Communication Channels (ie, control actions and sensor feedback)

Top-Level Errors (ie, abstract guidewords)

	ItemValueError	ItemTimingError	ViolatedConstraint	ServiceError
patient.PatientFingerclip -> pulseOx.SensorInput				
pulseOx.POOutSpO2 -> electronicHealthRecord.ehrSpO2	x	x		
doctor.DocGiveAdvice -> clinician.ClinGetAdvice				
pulseOx.POOutSpO2 -> appLogic.StoreSpO2Thread.incoming_spo2	x	x		
appDisplay.DispShowSpO2 -> clinician.ClinViewSpO2				
clinician.ClinTreatment -> patient.PatientTreatment				
appLogic.CheckSpO2Thread.Alarm -> appDisplay.HandleAlarmThread.Ala...				
pulseOx.POOutSpO2 -> appDisplay.UpdateSpO2Thread.SpO2	x	x		

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

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

	Early SpO_2	Late SpO_2
patient.PatientFingerclip -> pulseOx.SensorInput		
pulseOx.POOutSpO2 -> electronicHealthRecord.ehrSpO2	Cause: … Compensation: …	
doctor.DocGiveAdvice -> clinician.ClinGetAdvice		
pulseOx.POOutSpO2 -> appLogic.StoreSpO2Thread.incoming_spo2		Cause: … Compensation: …
appDisplay.DispShowSpO2 -> clinician.ClinViewSpO2		
clinician.ClinTreatment -> patient.PatientTreatment		
appLogic.CheckSpO2Thread.Alarm -> appDisplay.HandleAlarmThread.Ala...		
pulseOx.POOutSpO2 -> appDisplay.UpdateSpO2Thread.SpO2	Undocumented propagation!	

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 jjhugues@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213