CIS 890: High-Assurance Systems

Hazard Analysis

Lecture: Error Modeling Annex Version 2 - Introduction

Motivation

Why learn EMV2?

- Hands on experience with Hazard Analysis
- Based on existing modeling language (AADL)
- Can discuss various safety issues in Medical devices (Isolette)
- Safety Documents can be generated (FTA, FMEA, STPA)

AADL:EMV2

What is EMV2?

- AADL annex to support hazard analysis

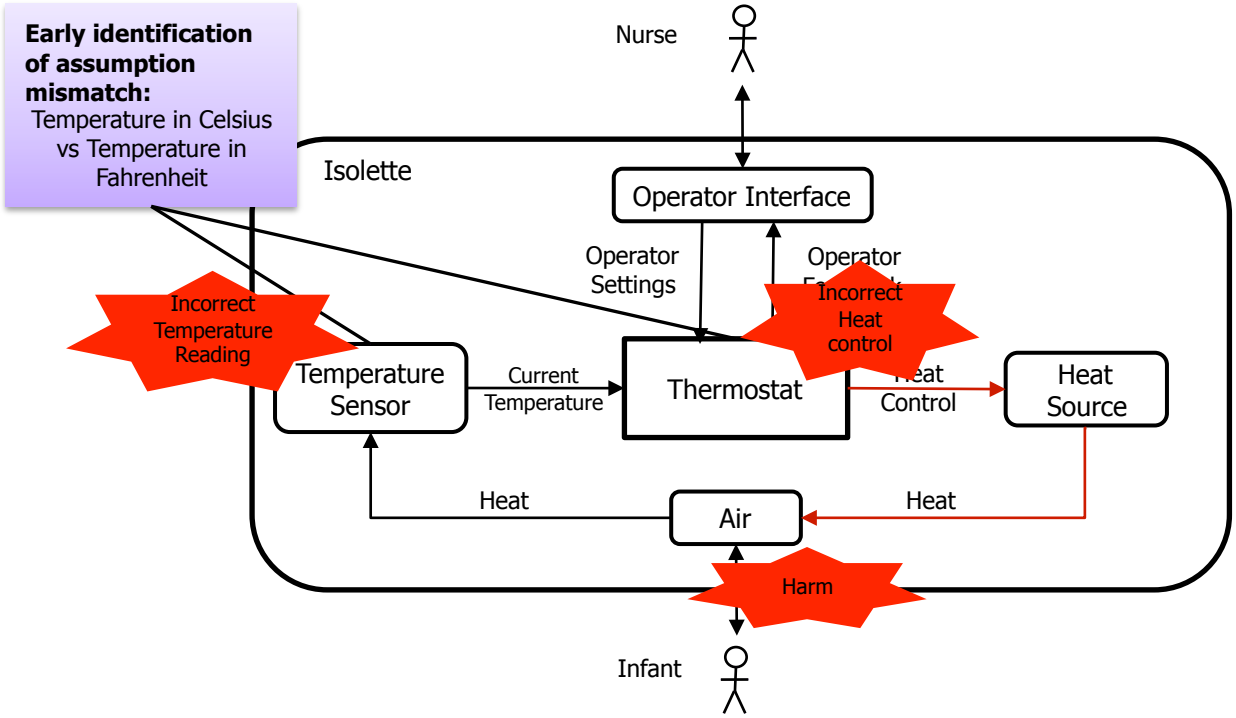
Why to use EMV2?

- Quantitative and qualitative assessment of system dependability
- Architecture fault modeling
- Alternate system architecture can be explored
- Changes are consistently propagated and reflected
- Hazard analysis can be incrementally performed

AADL:EMV2 Motivating Scenario

System as composition of components

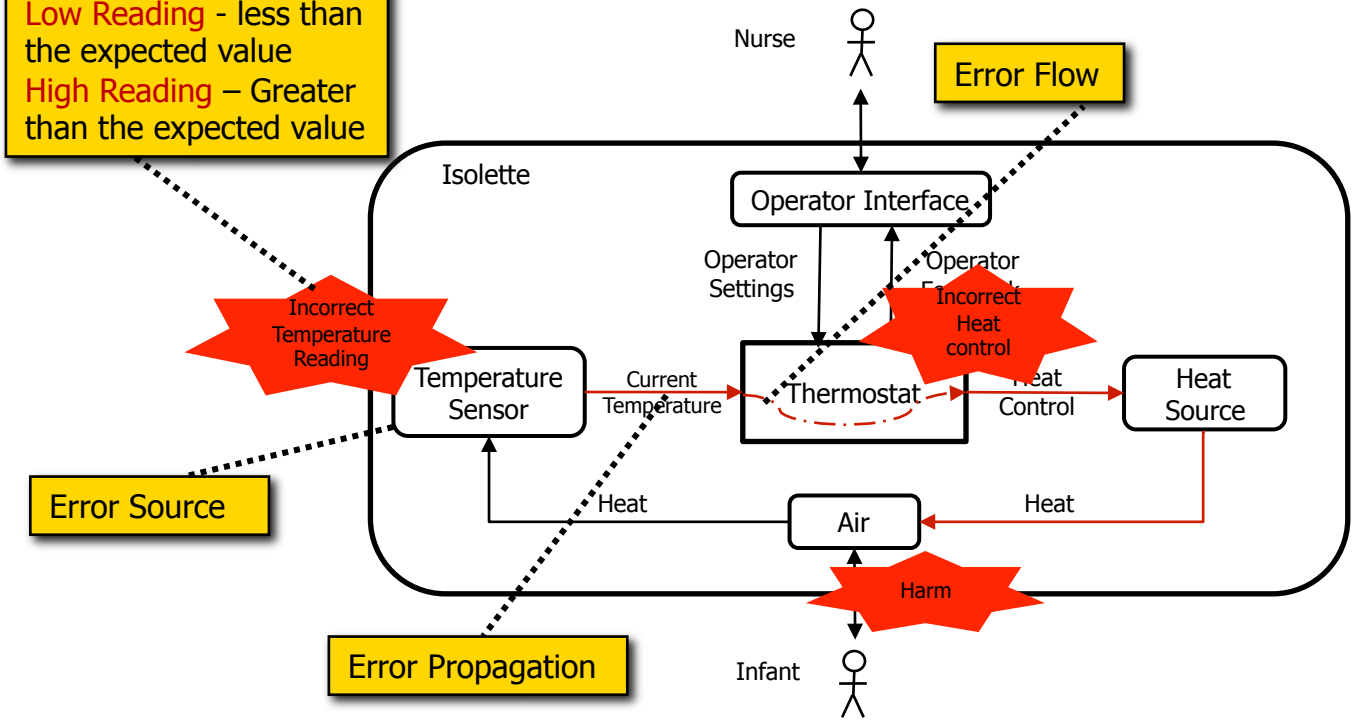
- Virtual integration



AADL:EMV2

Terminologies

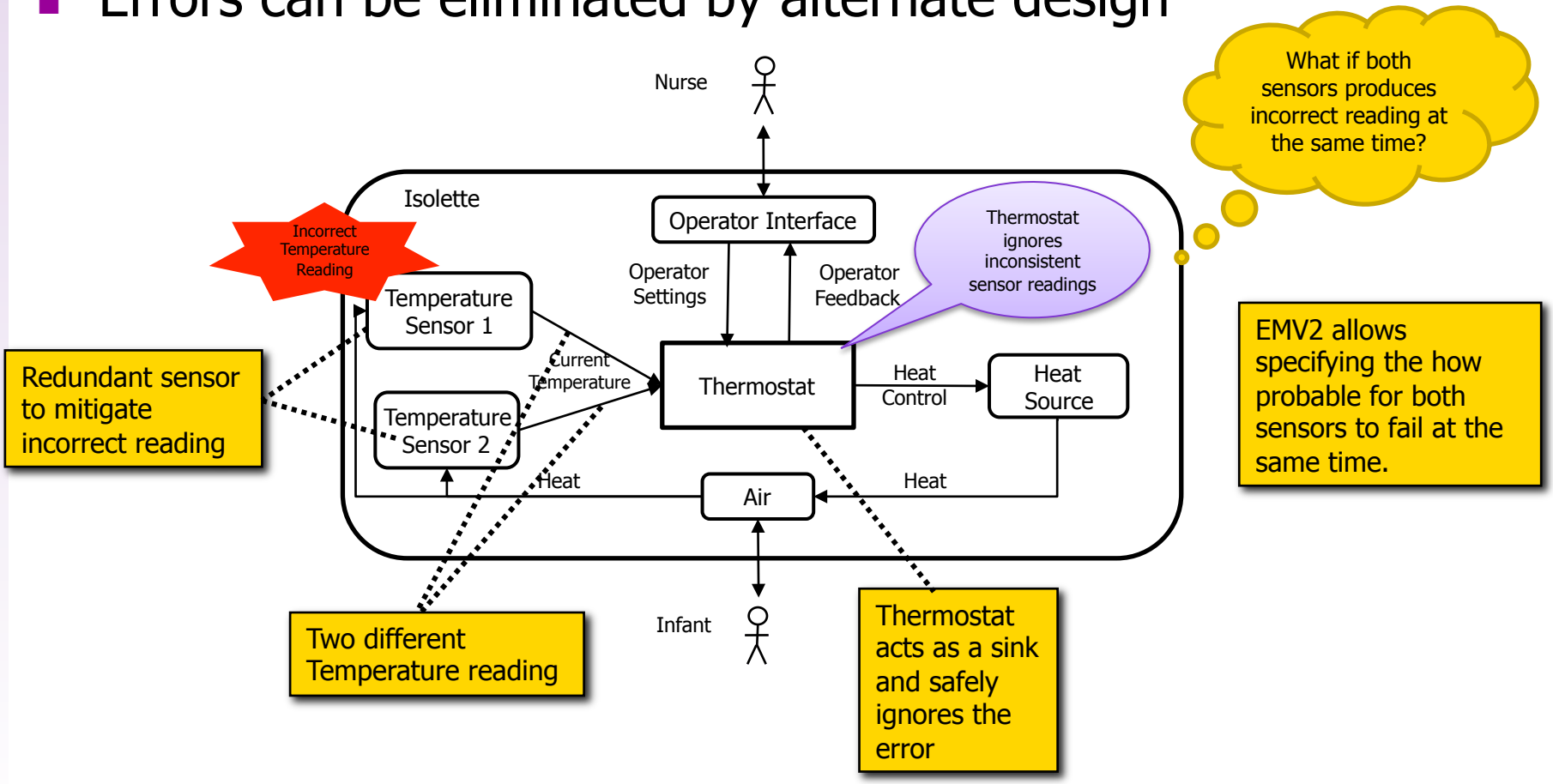
Abstract Error Type
Refined types may be:
Low Reading - less than the expected value
High Reading – Greater than the expected value



AADL:EMV2

Design change to mitigate error

- Errors can be eliminated by alternate design



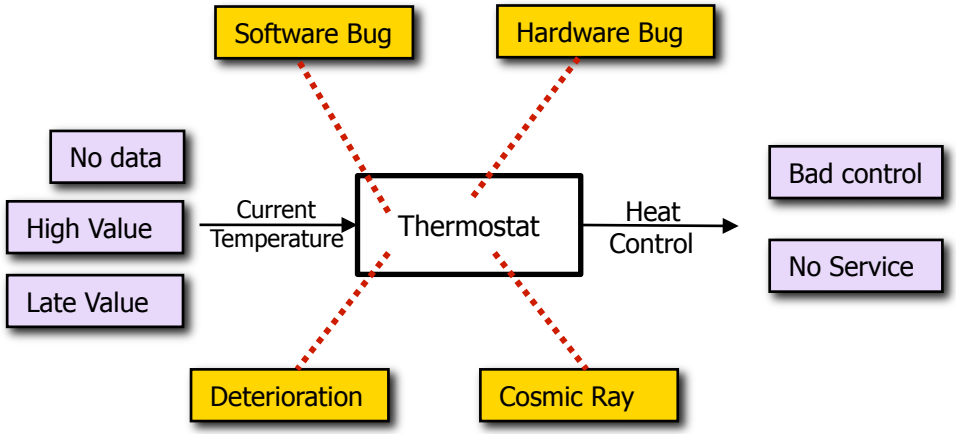
AADL:EMV2

Language Concepts in EMV2

- Error Type system
  - Flexible error type system
  - Pre-declared error type hierarchies
- Three levels of architecture fault modeling
  - Fault Propagation
  - Component Error Behavior
  - Composite Error Behavior

AADL:EMV2

Error Type : Identification

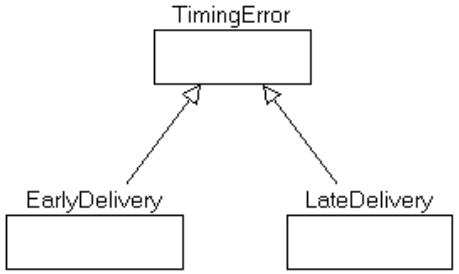


- Different ways in which a component may fail
- How faults in one component can impact other components

AADL:EMV2

Error Type

- An error type represents
  - Category of faults arising in a certain component
  - Category of error being propagated
  - Category of error represented by error behavior state
- Ability to organize these errors
  - Error type library
  - Error Hierarchy
- Predefined set of error types as starting point




```
TimingError: type;
EarlyDelivery: type extends TimingError;
LateDelivery: type extends TimingError;
```

Similar to object-oriented languages with inheritance

AADL:EMV2

Error Type : Custom Error Library

- Create custom error type to a particular domain / application
  - Completely new hierarchy
  - Extend existing hierarchy
  - Rename pre-defined types

EMV2 Custom Error definitions for Isolette

```
annex EMV2
{**
error types
  HeatControlError: type;
  AlarmError : type;
  FalseAlarm : type extends AlarmError;
  MissedAlarm : type extends AlarmError;
  StatusError : type;
  RegulatorStatusError : type extends
    StatusError;
  RegulatorModeError : type extends
    StatusError;
  MonitorStatusError : type extends
    StatusError;
  MonitorModeError : type extends
    StatusError;
  ThreadFault renames type
    ErrorLibrary::EarlyServiceTermination;
  InternalError : type;
  DetectedFault : type;
  UndetectedFault : type;
end types;
**};
```

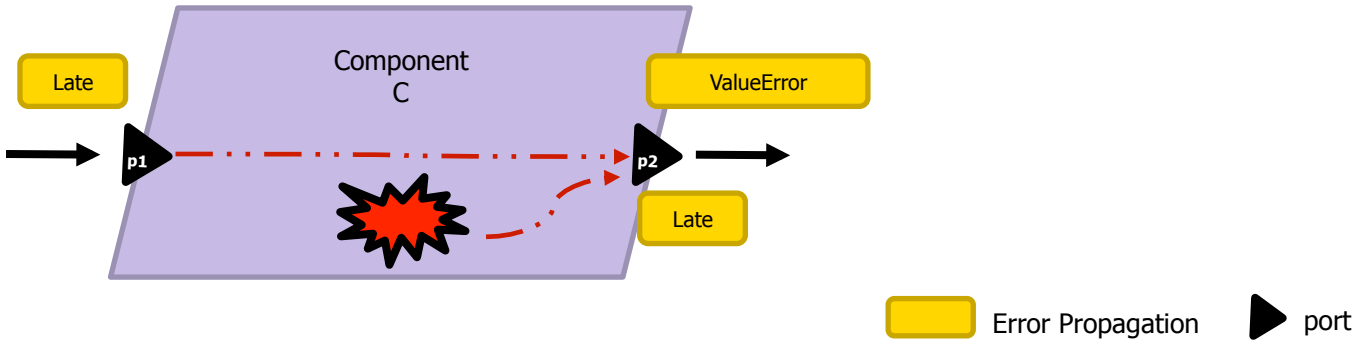
New Error type

Extending existing error type

Renames a error type to Isolette

AADL:EMV2

Annotation Model with Error Types

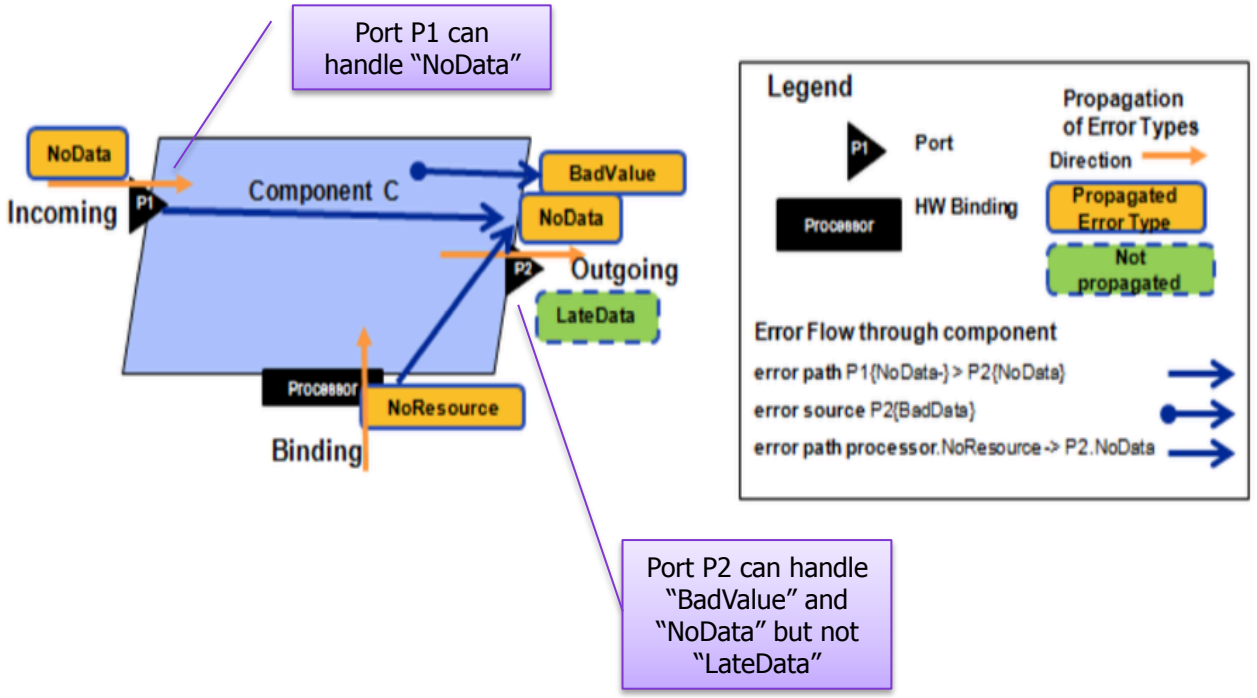


- Error propagation and containment are associated with interaction points eg: ports, bus, binding points etc.
- Error source: component in which error originates
- Error path and sink specifies how components respond to incoming errors

AADL:EMV2

Error Propagation

- Error propagation is like a error contract for a component
- It specifies the set of errors a particular port can handle



AADL:EMV2

Error Propagation : Isolette

```
thread manage_heat_source_mhs
features
  heat_control : out data port;
  current_temperature : in data port;
  desired_range : in data port;
  regulator_mode : in data port;

annex EMV2
  {**
    use types ErrorLibrary,Isolette_Errors;
    use behavior Isolette_Errors::FailStop;
  error propagations
  heat_control: out propagation {HeatControlError};
  current_temperature: in propagation {UndetectableValueError,OutOfRange};
  regulator_mode: in propagation {RegulatorModeError};
  --no errors from desired_range
  flows  --model error in detection and reporting
    mrmoor: error sink current_temperature{OutOfRange};
    mrmsve: error path current_temperature{UndetectableValueError}
      -> heat_control{HeatControlError};  --bad temp causes bad heat control
    mrmif: error path regulator_mode{RegulatorModeError}
      -> heat_control{HeatControlError};  --bad mode causes bad heat control
  end propagations;

**};
```

Port with in/out specifies the direction of information flow

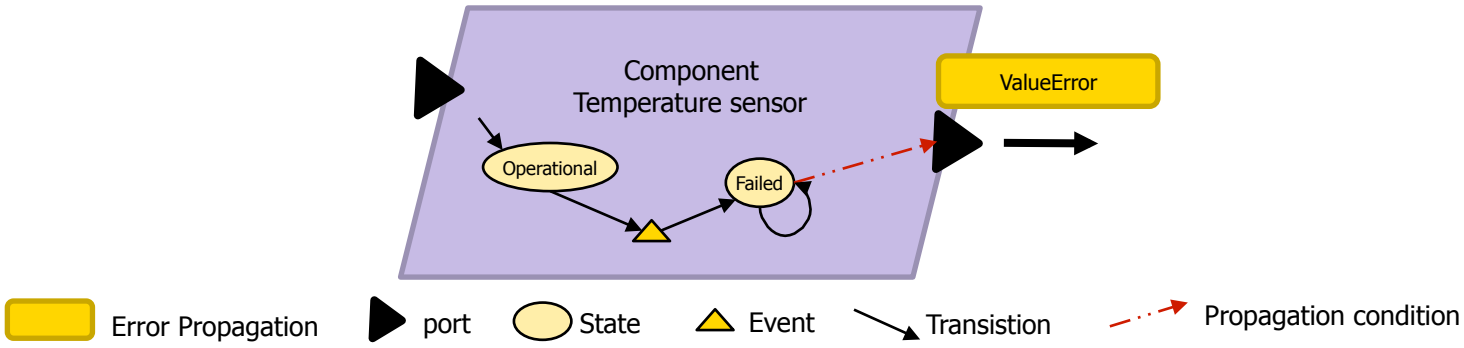
Error Propagation:
Specifies the set of errors a port can handle

Error Sink:
Specifies OutOfRange error is detected and suppressed ie. Not further propagated

Error Path:
Specifies how an incoming error is transformed into outgoing error

AADL:EMV2

Component Fault and Recovery Behavior



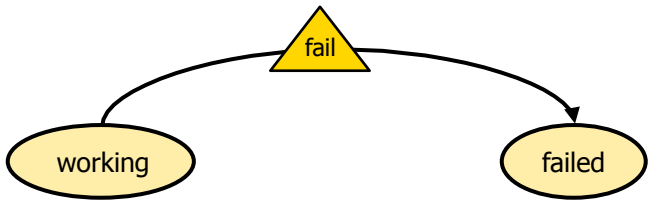
- Fault occurrences and failure modes within a component
- Event or incoming error may trigger transitions
- Three types of Events
  - Error event
  - Recovery event
  - Repair event
- Propagation condition specifies at which mode a component may produce error or transform error

AADL:EMV2

Error State Machines

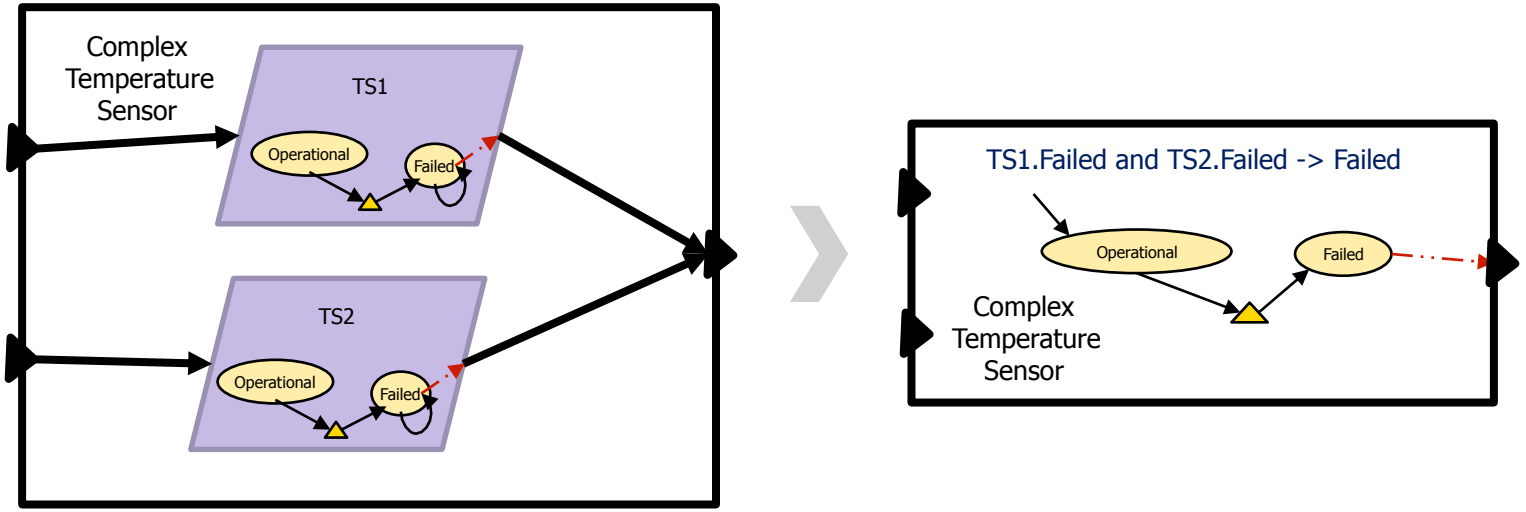
- Define error state machines in Error Library
- Use them in components
  - `use behavior Isolette_Errors::FailStop;`

```
--error state machine for components that
--have out-of range values when failed
  error behavior FailStop
    use types isolette;
    events fail: error event;
    states
      working: initial state;
      failed : state;
    transitions
      working -[fail]-> failed;
  end behavior;
```



AADL:EMV2

Composite Behavior



- Error behavior of a system in terms of its subsystem behavior
- Abstract the sub-component behavior

AADL:EMV2

Composite Behavior

- Similar to component error states, define states in error library
- Use it to define the error behavior of a system/component
- In terms of its sub-components

```
error behavior CompositeFailure
  use types Isolette_Errors;
  states
    Operational: initial state;
    ReportedFailure: state {DetectedFault};
    MissedFailure: state {MissedAlarm};
    FalseAlarm: state {FalseAlarm};
end behavior;
```

Subcomponents of temperature sensor

```
composite error behavior
states
[regulate_temperature.ReportedFailure or monitor_temperature.ReportedFailure
]->ReportedFailure;
[regulate_temperature.MissedFailure]->MissedFailure;
end composite;
```

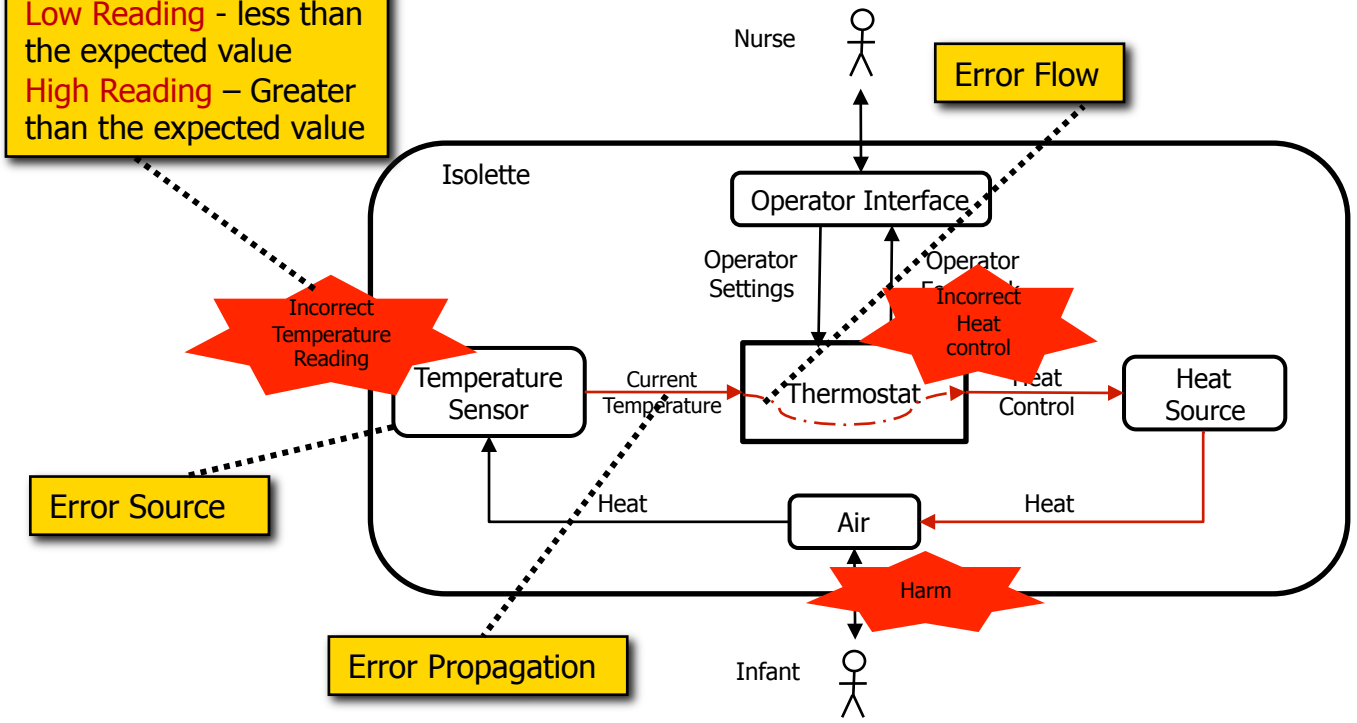
Summary

- Architecture-centric hazard analysis
- Incremental design and evaluation
- Error Types
- Error Propagation
- Component and Composite behavior

AADL:EMV2

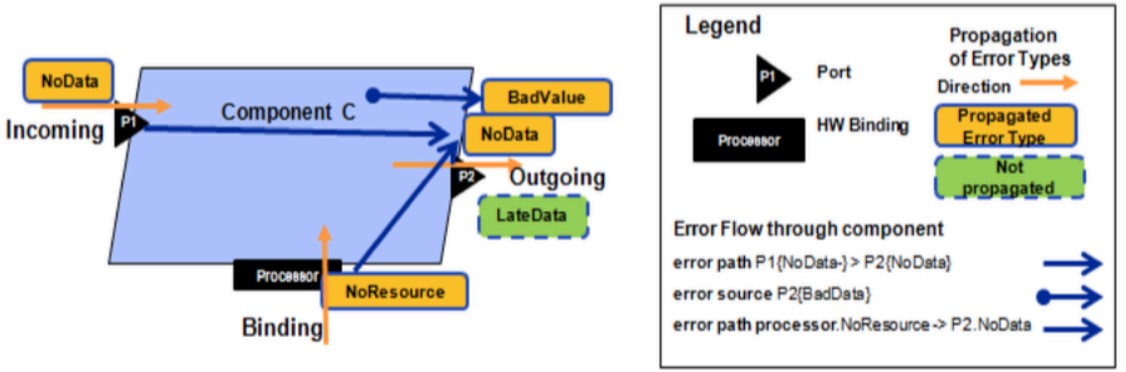
Terminologies

Abstract Error Type
Refined types may be:
Low Reading - less than the expected value
High Reading – Greater than the expected value



AADL:EMV2

Fault Propagation



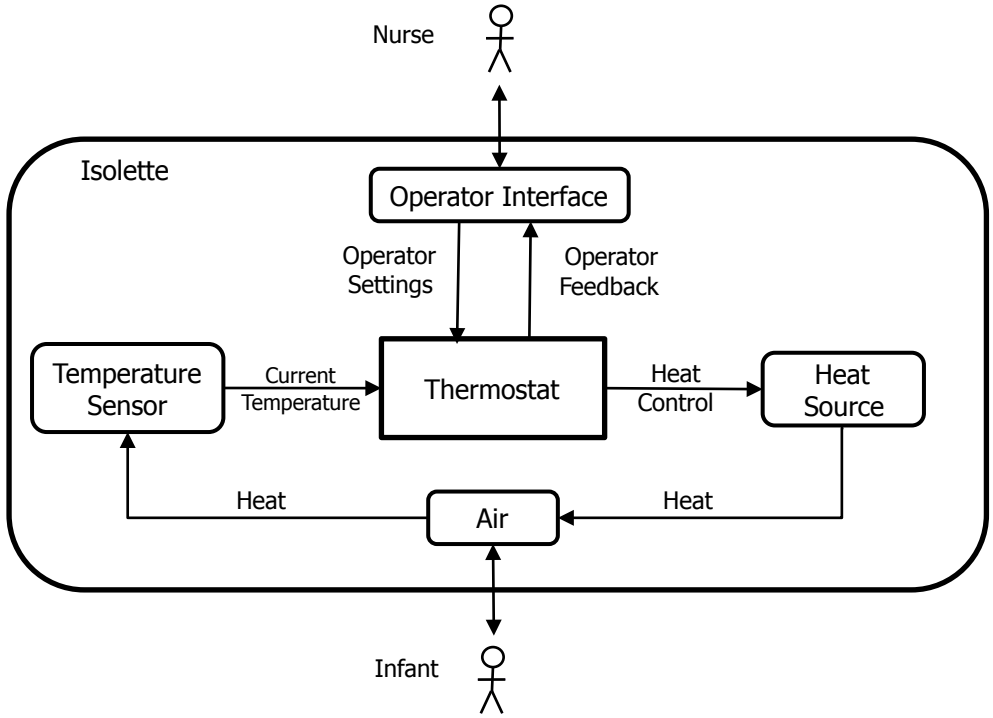
- Error propagation and containment are associated with interaction points eg: ports, bus, binding points etc.
- Error types are the different faults being propagated
- Error source: component in which error originates
- Error path and sink specifies how components respond to incoming errors

AADL:EMV2

Language Concepts in EMV2

- Error Type system
  - Error types to characterize faults (source of error)
  - Predefined set of error types
  - Extendable and flexible to adapt
- Fault Propagation
  - Propagation of faults and their impact between components, systems and environment
- Component Behavior
  - Error states and transitions
  - Error trigger events
  - Error propagation and transformation
- Consistency between various HAT(FTA, FHA, FMEA)
- Alternate system architecture can be explored
- Changes are consistently propagated and reflected
- Hazard analysis can be incrementally performed

Isolette Model



AADL:EMV2 vs STPA

AADL:EMV2

- AADL Architecture
- Connections
- Error Types

STPA

- Control Structure
- Control Action
- Hazard Causes

EMV2 Definitions

Fault

Actual cause of a hazard

Error

Difference in state from the correct state

Failure

Deviation in behavior from a nominal specification

AADL:EMV2

Error Type System

EMV2 provides a flexible and hierarchical type system

Different hierarchical error types

- Value Errors
- Timing Errors
- Service Errors

AADL:EMV2

Error Type System

- Error is defined as the difference in state from the correct state.
- In EMV2 Error types are treated as a type set with a single element type

Service

- A service S is defined as a sequence of n service items si with n > 0
- A service item si is a pair (vi, di), where
  - vi is the value of the service and
  - di is the delivery time of the service

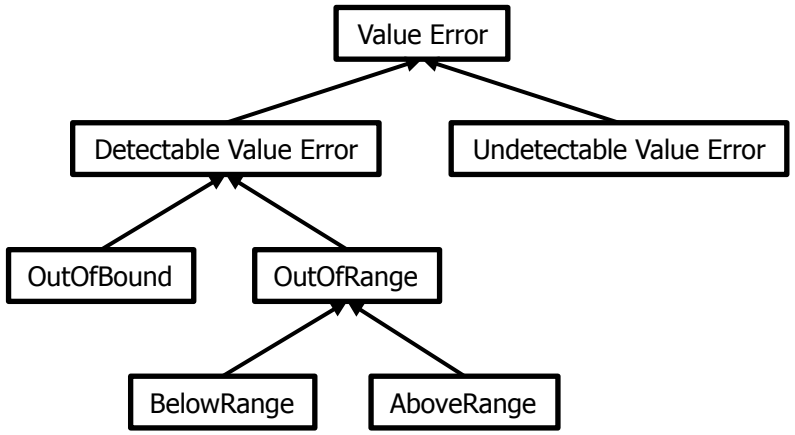
AADL:EMV2

No Error

- A service item(si) is defined as correct when
  - vi ∈ Vi and di ∈ Di, where
  - Vi is the correct range of values for vi, and
  - Di is the correct range of delivery time for di
- Challenge is knowing the correct values for a service item
  - For analysis: Assume there exist an oracle/application domain function providing correct values for si
  - Implementation: Redundant systems to identify correct values
- Note: There is no EMV2 type for No Error, the empty error typeset represents the absence of error

AADL:EMV2

Value Error Hierarchy



- Value Error: $\exists\, s_i \in S \mid v_i \neq V_i$
- OutOfRange: Values that are outside of a component's specification
- OutOfBound: The current item's value is above or below the acceptable range of values

AADL:EMV2

Custom Error Types

- Custom error types can be created in EMV2
  - To provide meaningful errors with respect to a component
  - Eg. HotAir vs AboveRange for Thermometer
- Three ways to create custom error types in EMV2
  - Completely new hierarchy
  - Extending an existing hierarchy
  - Renaming an existing error type

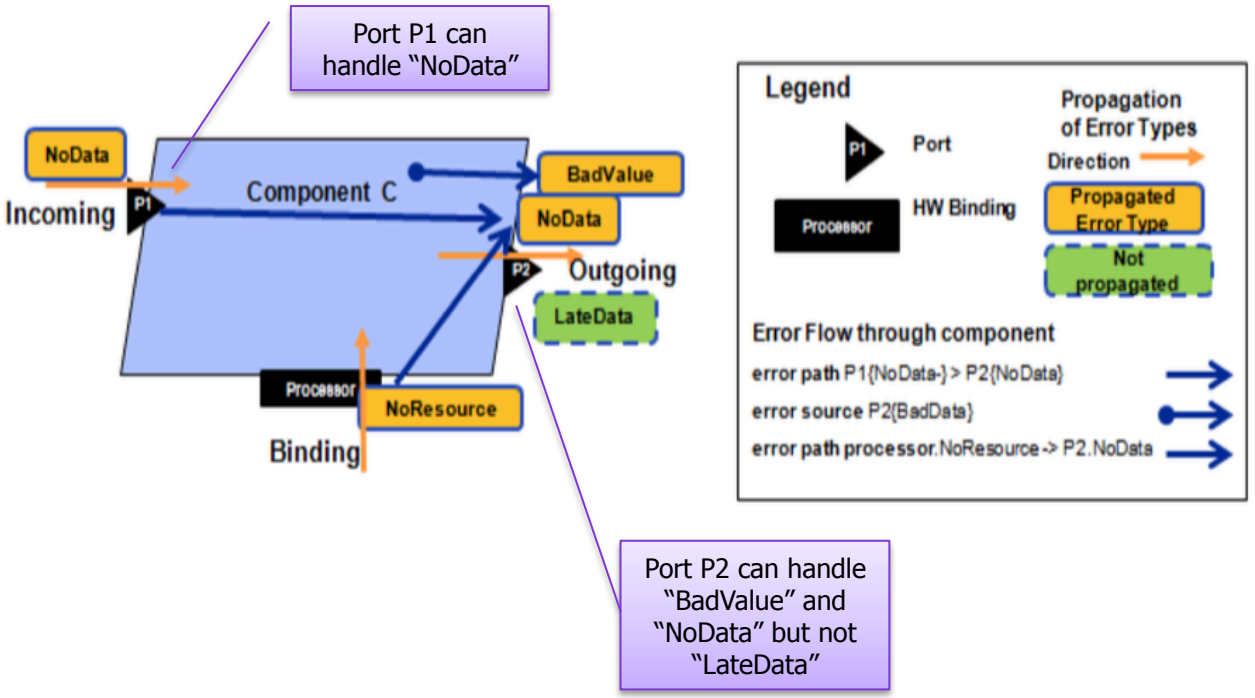
New error types by extending existing type

```
TimingError: type;
EarlyDelivery: type extends TimingError;
LateDelivery: type extends TimingError;
ValueError: type;
UndetectableValueError: type extends
  ValueError;
BenignValueError: type extends ValueError;
OutOfRange: type extends BenignValueError;
OutOfBounds: type extends BenignValueError;
BelowRange: type extends OutOfRange;
AboveRange: type extends OutOfRange;
```

AADL:EMV2

Component Error Propagation

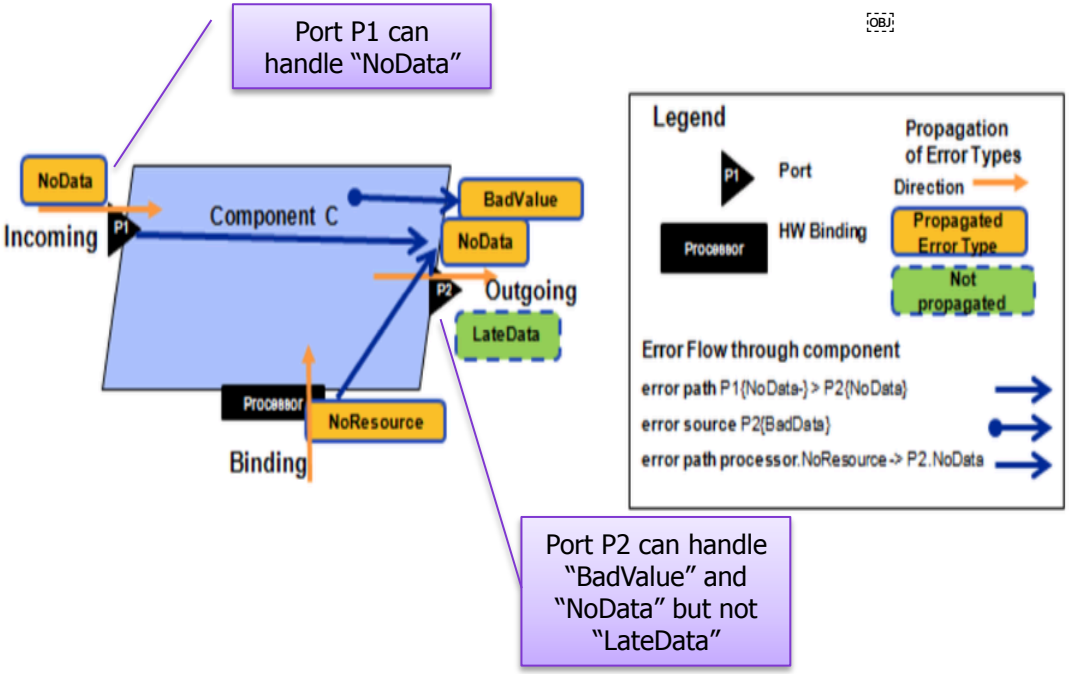
- Error propagation is like a error contract for the component
- It specifies the set of errors a particular port can handle



AADL:EMV2

Component Error Propagation

- Informs the set of possible incoming and outgoing errors
- It doesn't matter where the error originates
- Observing only at the boundary of a component
- In practice the there are derived from device specification

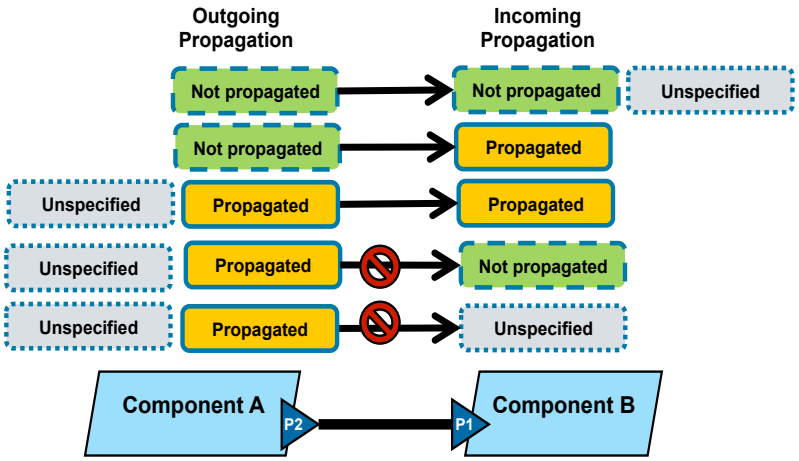


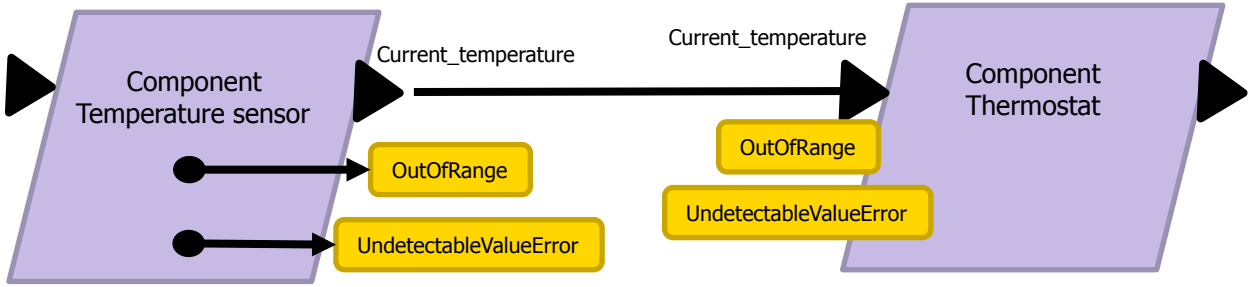

```
annex EMV2 {**
 use types ErrorLibrary;
error propagations
P1: in propagation {NoData} ;
P2: out propagation {NoData, BadValue};
P2: not out propagation {LateData};
end propagations; **};
```

AADL:EMV2

Consistency in Error Propagation

- To integrate two components A and B
  - If there exist a connection from port A.P1 to port B.P2, then
    $\forall e \in ErrorSet(A.P1) \mid e \in ErrorSet(B.P2)$
- Eg: Temperature sensor produces error "OutOfBound" and "UndetectableValueError"
- Also part of in propagation of Thermostat, therefore it is consistent

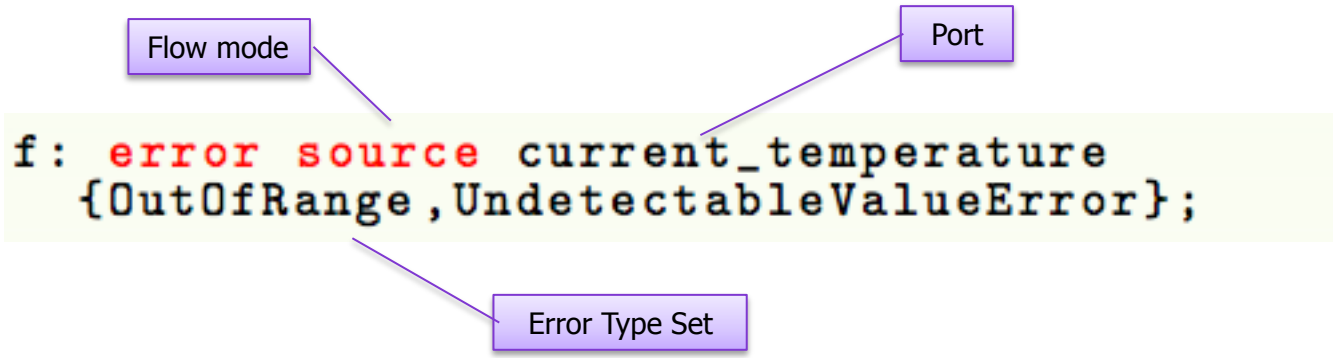




AADL:EMV2

Error Flow

- Specifies the relation between input and output ports
- It captures the error flow within a component
- Error flow can be
  - Source – Origin of an error
  - Sink – Containment / Suspension of error
  - Path – Transmission of error through component



AADL:EMV2

Error Flow : Path

- Describes the flow of error through a component
  - Propagation : Error A from in port is passed to out port
  - Transformation : Error A from in port is modified to error B and passed to out port

```
mmmiff: error path interface_failure
  {ItemOmission,ItemComission}
      -> monitor_mode(MonitorModeError);
mmminf: error path internal_failure
  {ItemOmission,ItemComission}
      -> monitor_mode(MonitorModeError);
mmmct: error path current_temperature
  {UndetectableValueError}
      -> monitor_mode(MonitorModeError);
```

All three path represents transformation

AADL:EMV2

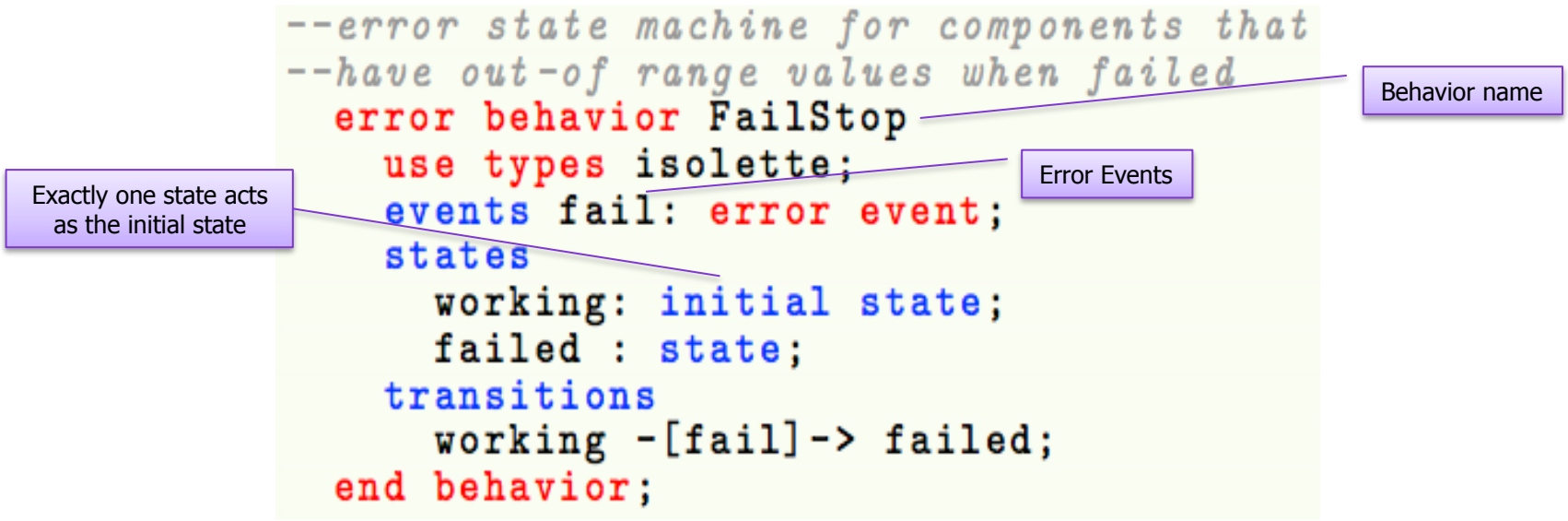
Component Error Behavior

- Error flow of dependents on the state of the component
- Error flow may be constrained based on the component state
- Three types of Behaviors:
  - Error behavior
  - Repair/mitigation behavior
  - Recovery behavior
- State transitions are triggered by error events and incoming error
- Similarly outgoing errors can be specified in terms of current state and incoming error
- Detected errors are mapped to a specific logging port with a error message/code

AADL:EMV2

Component Error Behavior

- Error behavior FailStop used in Isolette error detection mechanism



- FailStop behavior :
  - When the component is in "Working" state the "fail" even may trigger state transition to "failed" state

AADL:EMV2

Component Error Detection

- When an error is detected, the system may raise alarm or log error
- Component error behavior is used to specify detection condition

```
component error behavior
  detections
    failed -[ ]-> internal_failure!;
  --in "failed" state send event out
  --port internal_failure
end component;
```

The above specifies, when the component is in failed state, sends out an event through internal_failure port to raise alarm

AADL:EMV2

Composite Behavior

- Component's error behavior is specified in terms of its subcomponent

Error states of the component Isolette

```
error behavior CompositeFailure
  use types isolette;
  states
    Operational: initial state;
    ReportedFailure: state {DetectedFault};
    MissedFailure: state {MissedAlarm};
    FalseAlarm: state {FalseAlarm};
end behavior;
```

Error Types

Composite behavior: a constraint on the subcomponent's states to decide the component's state

```
composite error behavior
states
[temperature_sensor.failed
  or thermostat.ReportedFailure
  or heat_source.failed]->ReportedFailure;
[temperature_sensor.flakey
 or thermostat.MissedFailure]->MissedFailure;
end composite;
```