

Error Model Annex Revision

- Kickoff:
- Increased activity in architectural fault modeling
- Error Model Annex needs to be updated for AADL V2
- Looking for improvement input
- You are welcome to join and contribute
- phf@sei.cmu.edu

Error Model Annex Revision

- Update to AADL V2
 - Use of AADL property mechanism
 - Adapt to Virtual Bus, Virtual Processor, Abstract
 - Adapt to Arrays & connection patterns
- Improvements due to experiences
 - Error Model Errata
 - U. Aachen, Thales, ESA: System-software co-engineering & performability
 - Armed Forces U. Munich: Dependability analysis
 - SEI, LIP6: Fault impact analysis
- Document modeling requirements
 - Identify modeling use scenarios
- More formally define the intended semantics
 - Labeled transition systems, Petri nets (colored, timed, stochastic)

Set up Wiki area
Use email list

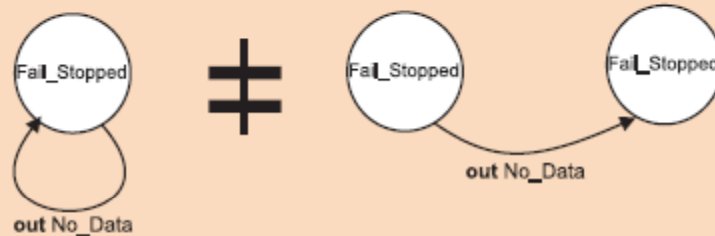
U. Aachen: Co-engineering

- System-Level Integration Modeling (SLIM)
 - System models
 - Performability & dependability
 - Degraded modes of operation
 - Fault detection, isolation, recovery
- AADL extensions
 - Initial values for data, port
 - Resume in current mode on activation (resume not only for threads)
 - Observable boolean ports (maybe in error model)
- Error model extension
 - Error event internal to component: place in EM impl
 - Error states internal to component
 - Error state resumption vs. initial state

U. Munich: Propagation

Propagation should be explicitly specified

- *Normal* (Rate of Exponential Distribution)
- Rugina Semantics (Probability, single propagation)



- Arbitrary (Probability, multiple propagation)

Question 2: State via data channel

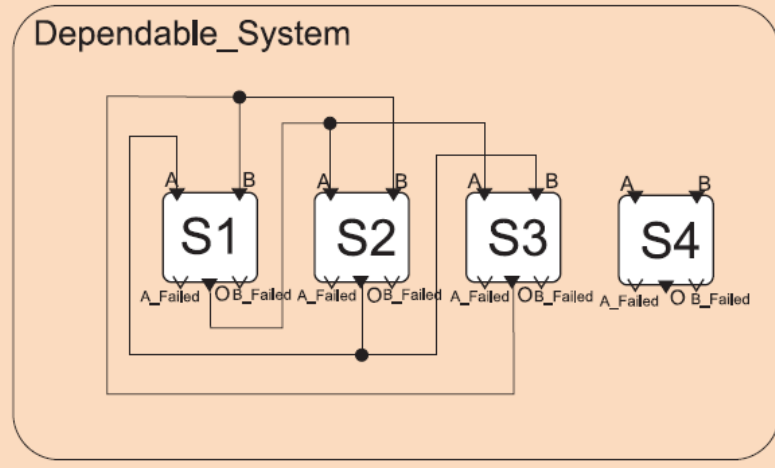
Guard_In => mask when (A[Error_Free] and B[No_Data, Bad_Data]
or (A[No_Data, Bad_Data] and B[Error_Free]),
Bad_Data **when others**
applies to A, B;

Idea

- maybe use clause like (**not** A[No_Data, Bad_Data]) instead of A[Error_Free]

U. Munich: Propagation - 2

3 active components - 1 cold spare



correct or false voter decision possible

Guard_Transition \Rightarrow (S2.A_Failed[Error_Free] and S3.A_Failed[Error_Free])
or (S2.A_Failed[Bad_Data] and S3.A_Failed[Bad_Data])

correct or false voter decision possible

- switching in large example may never occur due to two corrupt voters

U. Munich: Propagation - 3


Issue or feature?

- Two different meanings of a propagation name
- Context-sensitive


Example clause

Guard_Out => Beta **when self [Alpha] applies to** Some_Out_Port

Propagation



short-hand notation of
some error states



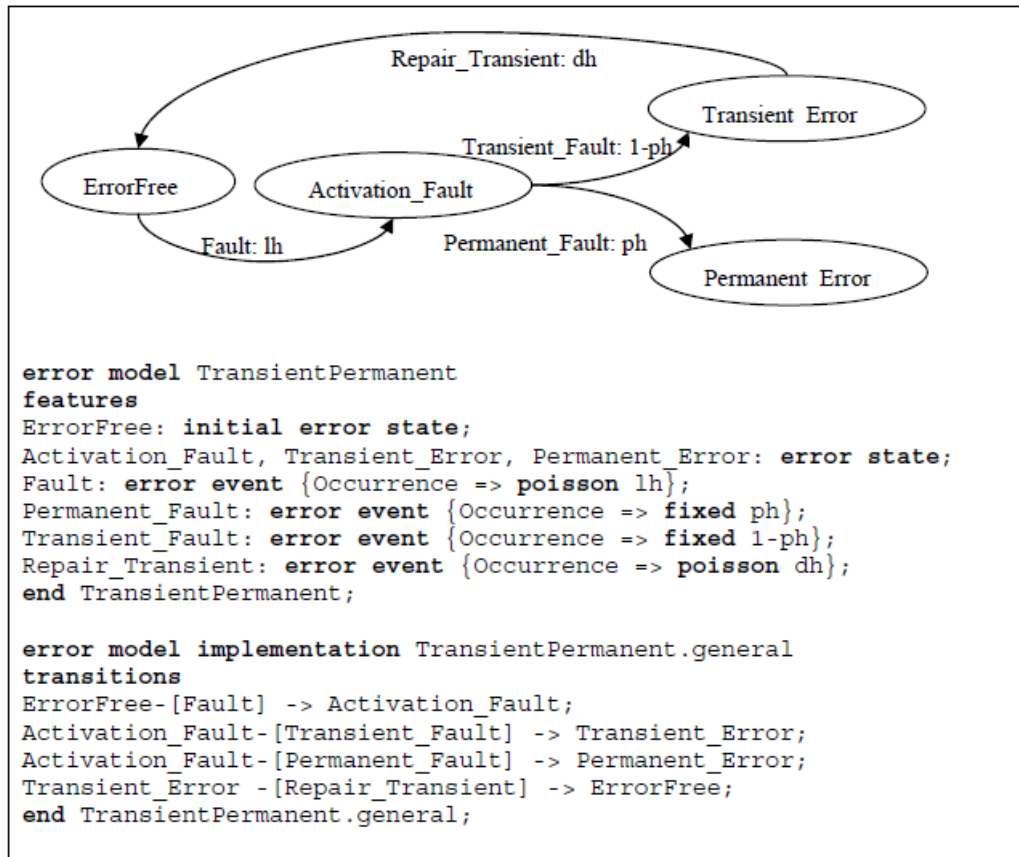
Fault Impact Modeling

- External error propagation view & internal error behavior
 - Propagated errors in error type
 - Error states & events in implementation
- Error propagations
 - Error propagation names & error categories
 - Global dictionary vs. independent specification
 - Equivalence declaration
 - Incoming & outgoing error categories
 - For component
 - Per interaction point (port, binding, access)
 - Interaction match
 - Error propagation flows
 - Error propagation source (fault injection), sink (masking), pass thru, transformation
 - Conditional propagation
- EM refinement
 - General: extension of error classes
 - Component-specific: interaction point specific, propagation flows

Error Behavior Modeling

- Error state machine in EM implementation
 - Propagated errors in error type
 - Error states & error events in implementation
- Error events
 - Errors, repairs, hazards
 - Directly affect only error states
- Error behavior
 - Fault behaviors
 - Permanent, transient
 - HW & SW specific behavior
 - Repair behaviors
 - Typing of “error events”

Permanent & Transient Errors



Error Behavior Modeling

- Error states & out propagations
 - Propagation of state
 - Alternative to loop-back transition:
 - `Guard_Out => outprop when self[errorstatetoprop];`
 - Propagation of transition
 - Are we propagating an error event singleton?
- Error state transitions
 - Incoming propagated errors without affecting state
 - Singletons or discretized propagation of component error state
 - One-step or two-step transaction?
 - Incoming propagations can affect error state and out propagations
 - Old or new error state can affect out propagation
- Error behavior refinement
 - Generic: handling of incoming error categories & error events
 - Component-specific: port-specific transition conditions
 - Reusable generic conditions
 - e.g., 3-way voting on 3 generic incoming port names

Other Error Model Observations

- Out propagations cannot have port specific probabilities
- Interplay between out propagation in loop-back transition & Guard_out for out propagation
 - Overwrite semantics?
- Difference between Guard_In masking and Guard_Out masking?
- Guard conditions
 - Refer to ports without error category: initial error state – interpreted as error free, i.e., no error propagation
 - Do not refer to some ports: any propagation & non propagation condition
 - **Not** means all error categories including NoError except for the specified ones. NOTE: the standard does not provide an explicit explanation, esp. regarding the inclusion of NoError (error free).
 - It may be good to limit the list of operands for the **ormore** and **orless** operators to error source names, i.e., ports and error propagations rather than full Boolean expressions.
 - Allow short-hand for error category on any incoming port?
Any[BadData]

Other EM Observations

- Error events
 - Distinguishable error vs. repair event
 - Property, keyword
 - We may want to explicitly specify that an error event is masked, i.e., does not affect the error state, not is it propagated.
 - The Error Model Annex (V1) only supports error states to be named as part of the **self** reference. This requires us to introduce an error state for each error event to be propagated and then name the error state.
 - Allow **self**.errorevent in guard condition?
- Guard_in plays multiple roles
 - Name alias
 - Association of incoming out propagations with specific ports
 - Acts as error state transition name or named transition condition:
 - based on incoming out propagations or connected component error states
 - must not contain own error state
- Error categories/types
 - Reflected in error propagation name, error event name, error state name
 - Cannot have same name for error event, error state, and error propagation
 - Explicit typing & type hierarchy?
 - Transient/permanent a property of an error category/error event?

Coordination with Other Efforts

- MARTE dependability extension
- AutoSAR/EAST approach to dependability modeling

From MARTE DAM Paper

Conceptual Model consists of System Model, Threat Model, Maintenance Model

Threat model

- The *Threats model* includes the threats that may affect the system, namely the *faults, errors, failures and hazards*.
- We have introduced an abstract concept of *impairment*, that can be specialized depending of the type of analysis domain, i.e., *failure* for reliability/availability analysis and *hazard* for safety.
- The model represents also the cause-effect relationships between the threats and the relationships between the threats and the system core concepts.