CPSC 875

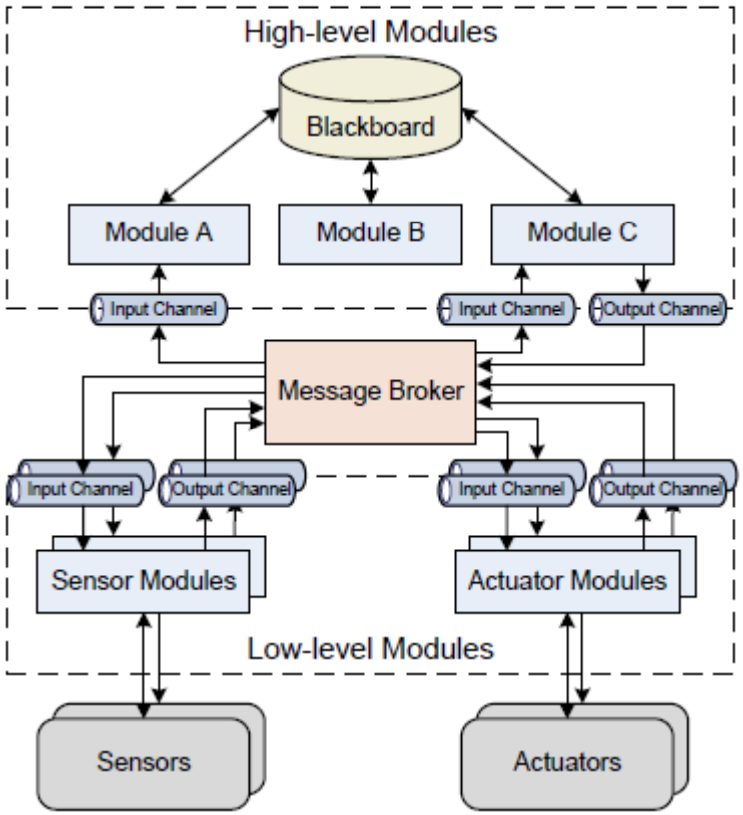
John D. McGregor

C10 – Error Design

Uncertainty

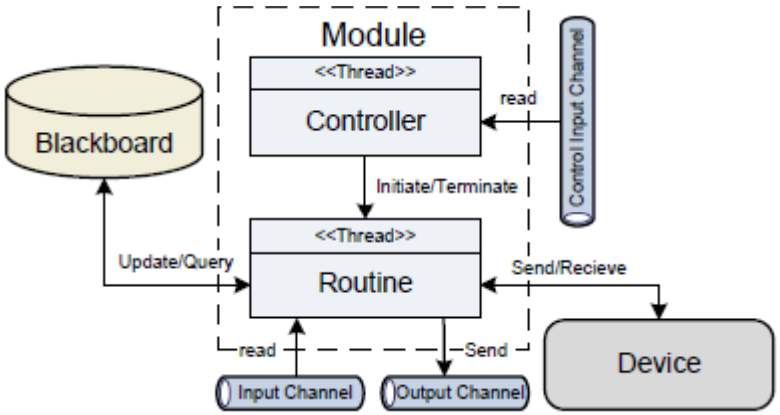
- Make uncertainty a first class entity in design
- Assume things fail
- Watchdog timers check that an operation has not frozen (as opposed to the modal dialog with a cancel button)
- Google File System is designed to recognize a failed disk drive and to work around it

Autonomous Robot

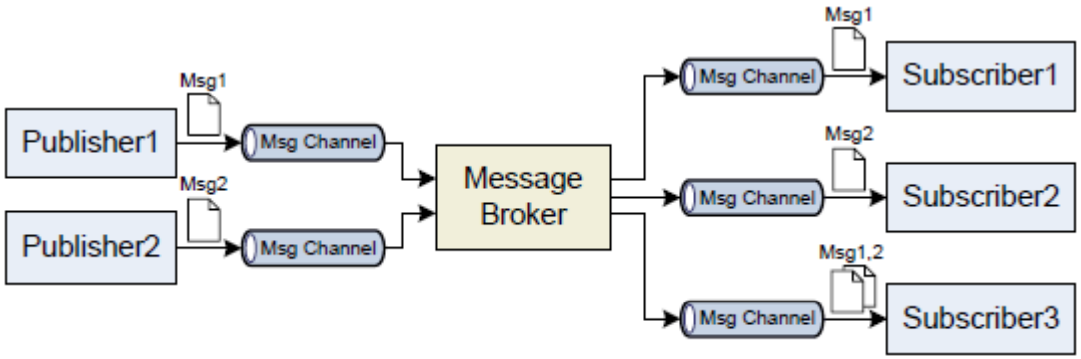


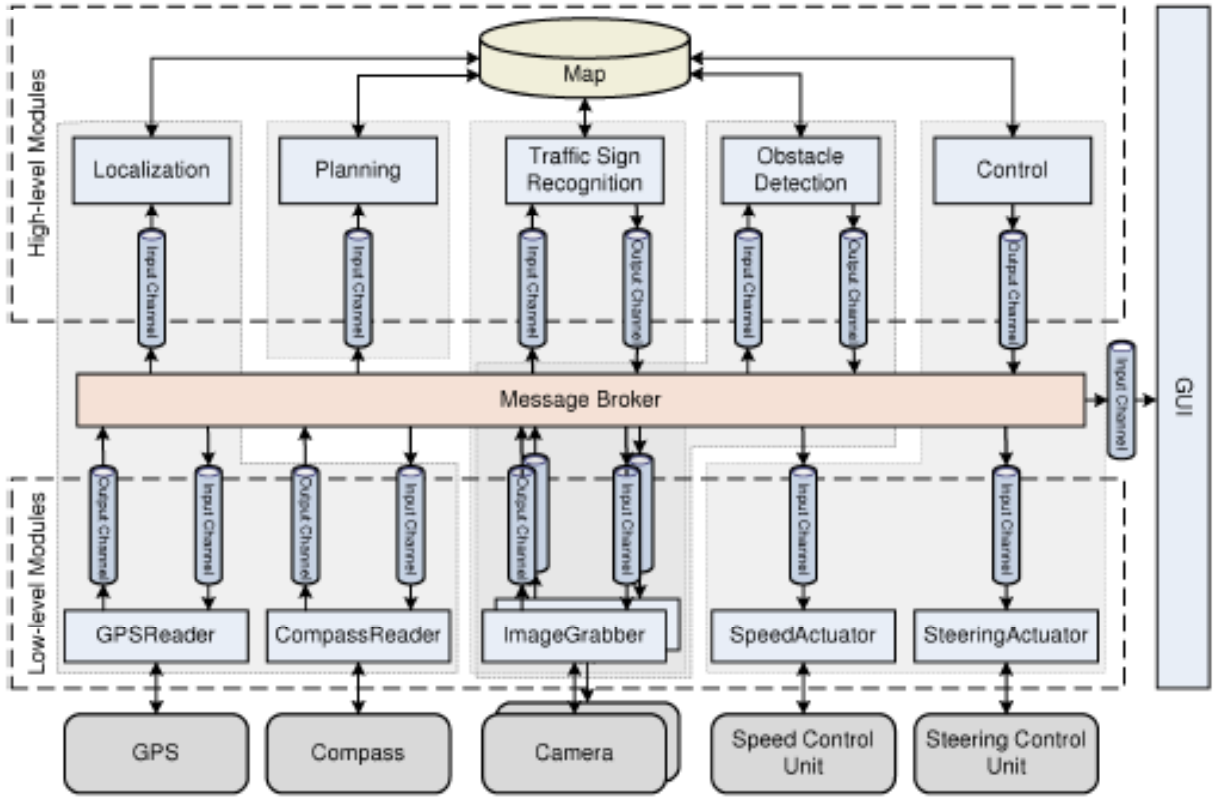
http://www.cs.ait.ac.th/~mdailey/papers/Limsoonthrakul-Arch.pdf

Module structure



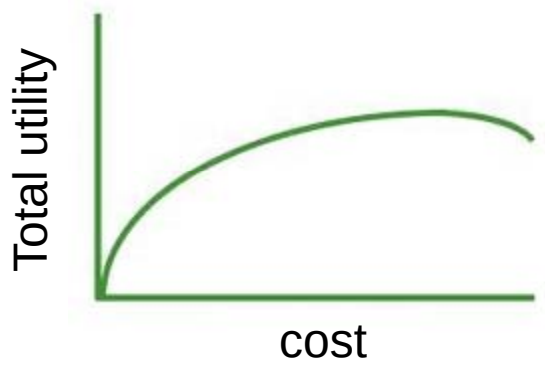
Publish/Subscribe Style




High-level Modules
Map
Localization
Planning
Traffic Sign Recognition
Obstacle Detection
Control
Input Channel
Output Channel
Message Broker
GUI
Low-level Modules
GPSReader
CompassReader
ImageGrabber
SpeedActuator
SteeringActuator
GPS
Compass
Camera
Speed Control Unit
Steering Control Unit

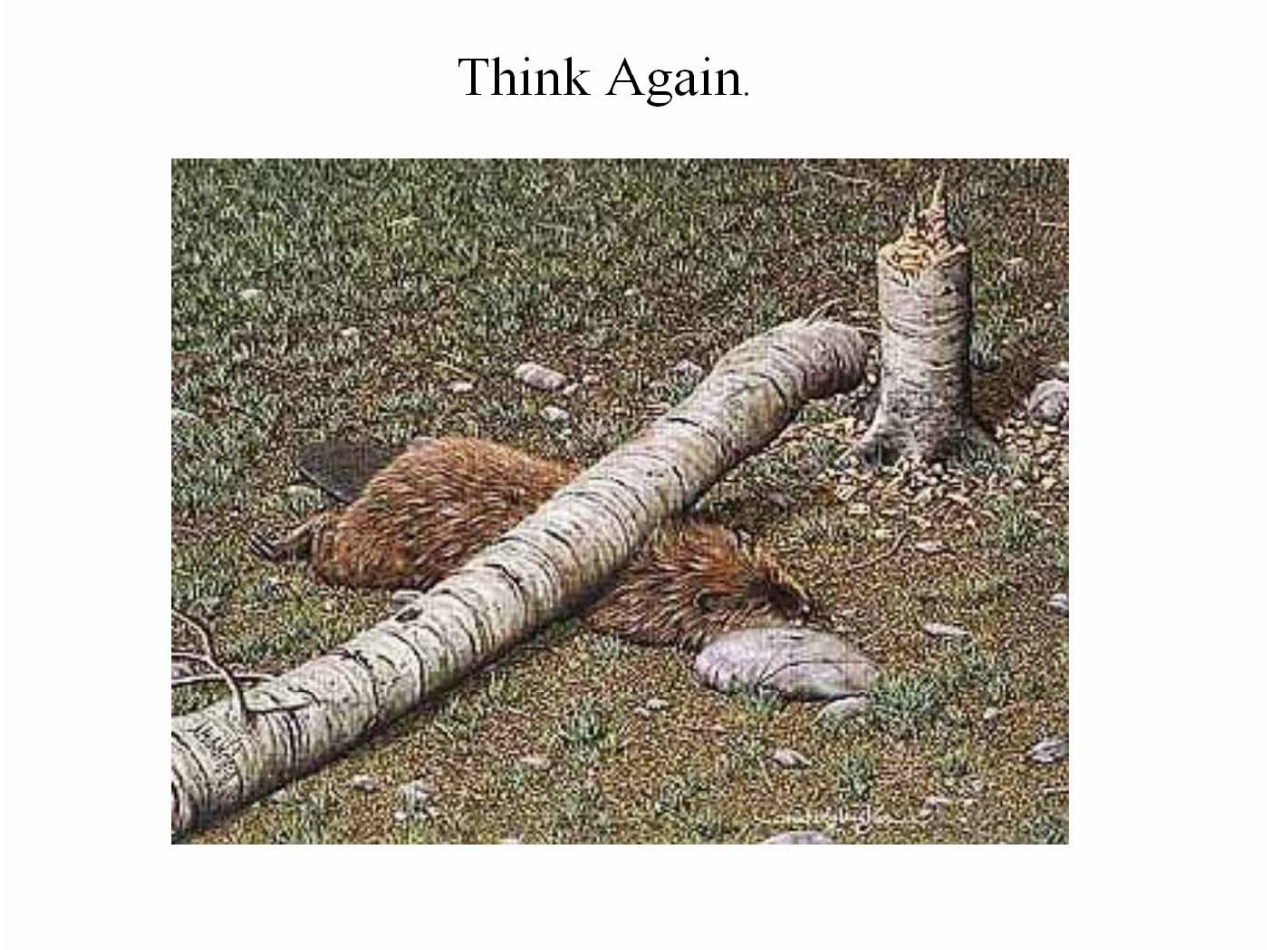
Utility

- Usefulness
- We assume that as a design satisfies more and more of the desired qualities its usefulness is increased
- But it costs more and more so at some point the increase in utility is not worth the increase in cost

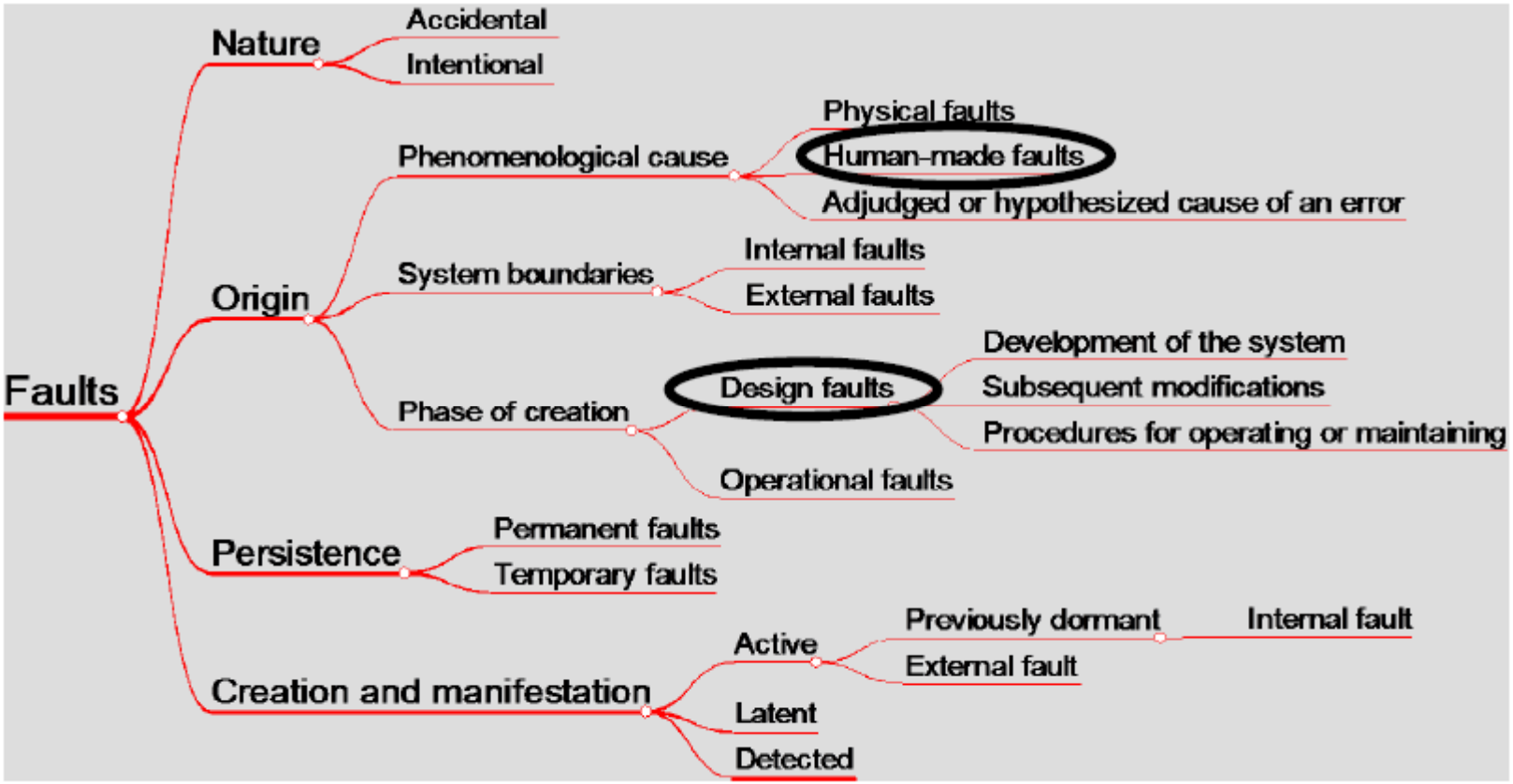


Design for Errors

Nothing can go wrong



From: http://academic.csuohio.edu/duffy_s/Section_03.pdf

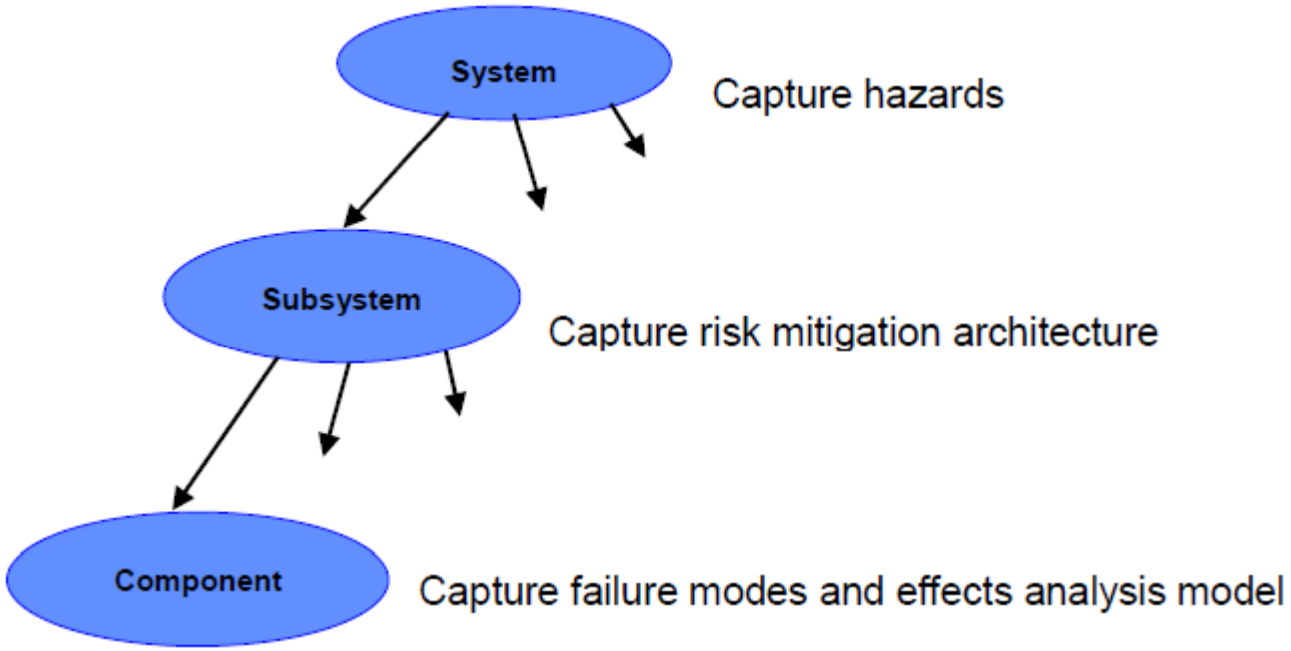


www.artemis-ia.eu/publication/download/?publication=98

AADL Error Annex

- https://wiki.sei.cmu.edu/aadl/images/4/42/ErrorModelOverview-04182012.pdf

Error design



Exception handling

- **Always clean up after yourself**
- **Never use exceptions for flow control**
- **Do not suppress or ignore exceptions**
- **Do not catch top-level exceptions**
- **Log exceptions just once**

PrimaryBackupPattern

```
system implementation PrimaryBackupPattern.impl
    subcomponents
      primary: system sys in modes (Primarymode);
      backup: system sys in modes (Backupmode);
    connections
      inprimary: data port insignal ->
      primary.insignal in modes (Primarymode);
      inbackup: data port insignal -> backup.insignal
    in modes (Backupmode);
      outprimary: data port primary.outsignal ->
      outsignal in modes (Primarymode);
      outbackup: data port backup.outsignal ->
      outsignal in modes (Backupmode);
    modes
      Primarymode: initial mode;
      Backupmode: mode;
end PrimaryBackupPattern.impl;
```

Error design

error model Example1

features

ErrorFree: **initial error state;**

Failed: **error state;**

Fail, Repair: **error event;**

CorruptedData: **out error propagation**

{Occurrence => fixed 0.8};

end Example1;

error model implementation Example1.basic

transitions

ErrorFree-[Fail]->Failed;

Failed-[**out CorruptedData]->Failed;**

Failed-[Repair]->ErrorFree;

properties

Occurrence => **poisson 1.0e-3 applies to Fault;**

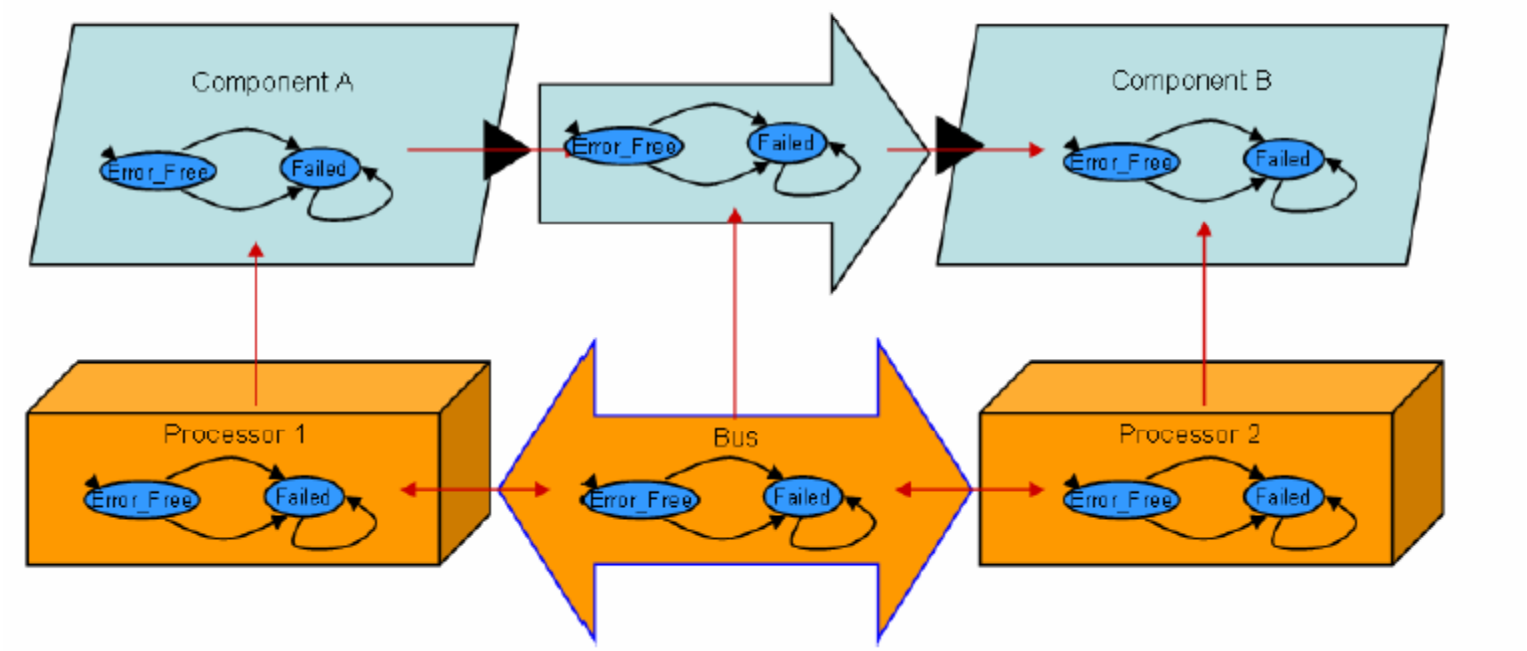
Occurrence => **poisson 1.0e-4 applies to Repair;**

end Example1.basic;

Using error model

```
system computer
end computer;
system implementation computer.personal
    subcomponents
      CPU: processor Intel.DualCore;
      RAM: memory SDRAM;
      FSB: bus FrontSideBus;
    annex Error_Model {**
      Model => My_ErrorModels::Example1.basic applies to CPU;
      Occurrence => fixed 0.9 applies to error CPU.CorruptedData;
    **};
end computer.personal;
```

Propagation



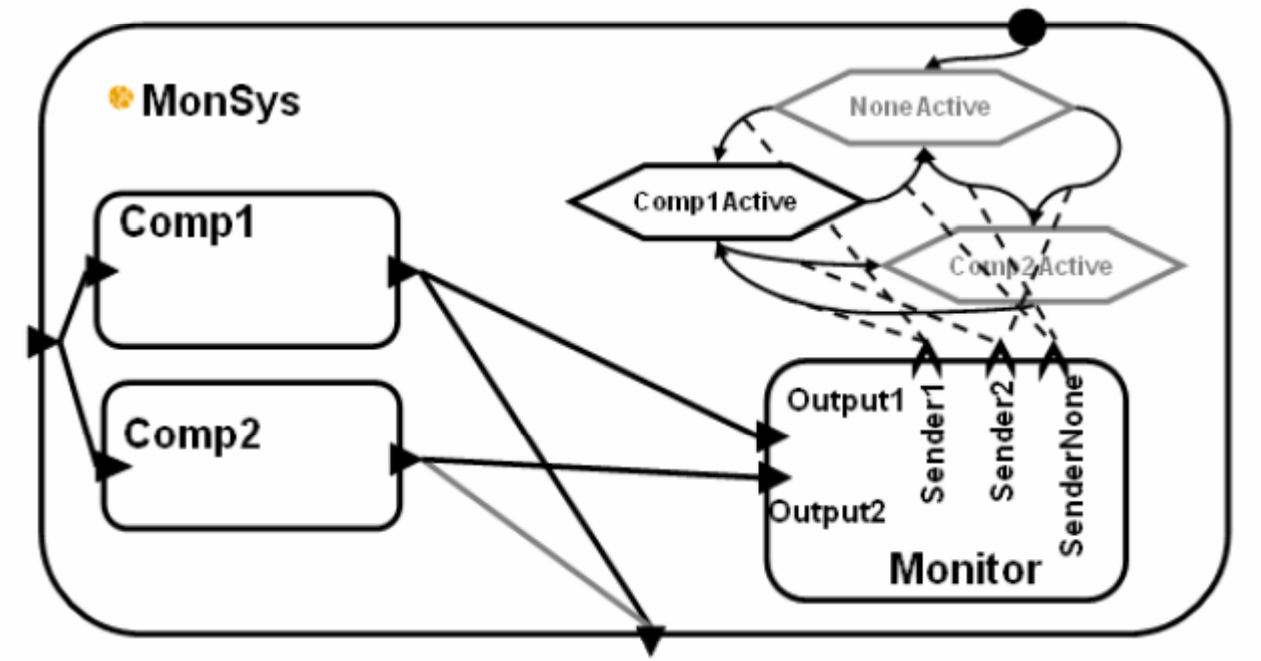
Error Propagation

Rule No.	Propagations may occur from	Propagations may occur to
D-1	Processor component	Every thread bound to that processor
D-2	Processor component	Every connection routed through that processor
D-3	Memory component	Every software component bound to that memory
D-4	Memory component	Every connection routed through that memory
D-5	Bus component	Every connection routed through that bus
D-6	Device component	Every connection routed through that device

Propagations

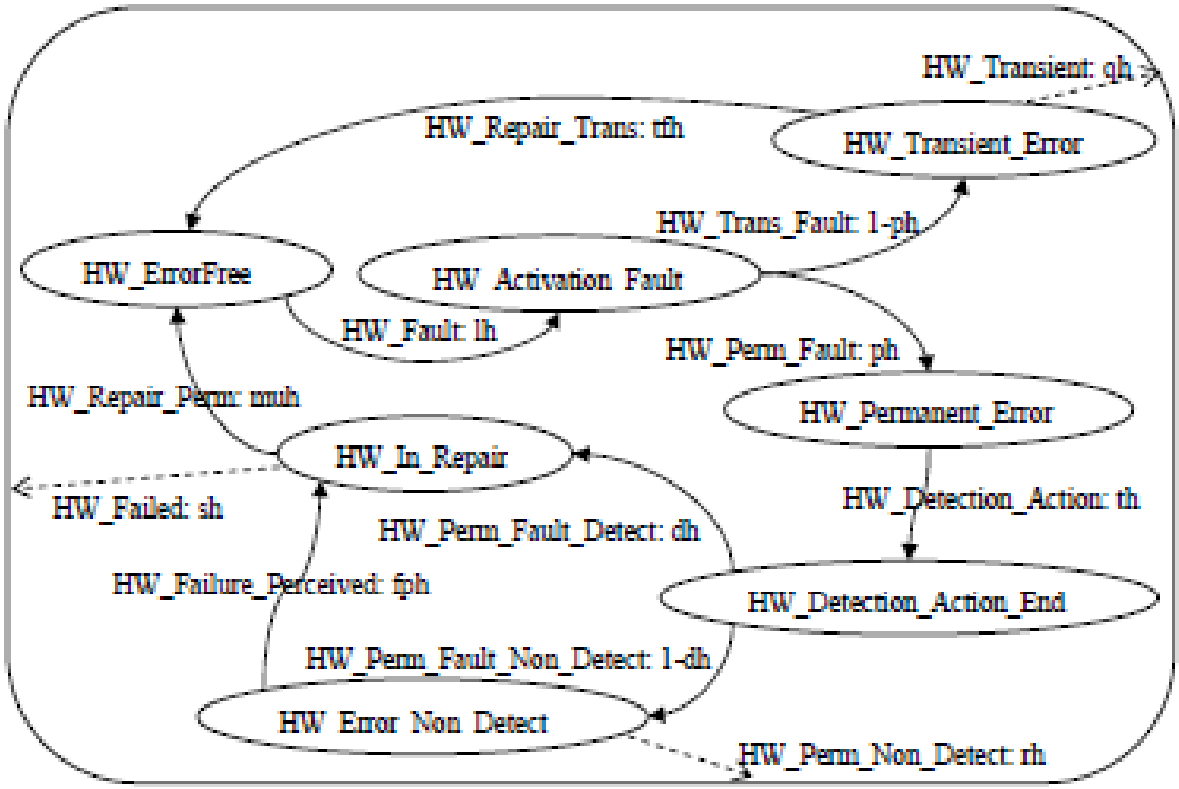
Rule No.	Propagations may occur from	Propagations may occur to
D-7	Application component	Each of the data components it has access to through **`provides`** and **`requires`** data access declarations
D-8	Shared component	All components that access it to through **`provides`** and **`requires`** data access declarations[1]
D-9	Application component	Every connection from any of its **`out`** ports
D-10	Connection	Every component having an **`in`** port to which it connects[2]
D-11	Application component	Any component via its outgoing connections
D-12	Client subprogram	Every server subprogram to which a call is bound
D-13	Server subprogram	Every client whose calls are bound to that server

Full spec


MonSys
Comp1
Comp2
NoneActive
Comp1Active
Comp2Active
Output1
Output2
Sender1
Sender2
SenderNone
Monitor

```
error model forHardware
features
HW_ErrorFree: initial error state;
HW_Activation_Fault, HW_Transient_Error, HW_Permanent_Error,
HW_Detection_Action_End, HW_Error_Non_Detect, HW_In_Repair: error
state;
HW_Fault: error event {Occurrence => poisson lh};
HW_Perm_Fault: error event {Occurrence => fixed ph};
HW_Trans_Fault: error event {Occurrence => fixed 1-ph};
HW_Detection_Action: error event {Occurrence => poisson th};
HW_Failure_Perceived: error event {Occurrence => poisson fph};
HW_Perm_Fault_Detect: error event {Occurrence => fixed dh};
HW_Perm_Fault_Non_Detect: error event {Occurrence => fixed 1-dh};
HW_Repair_Trans: error event {Occurrence => poisson tfh};
HW_Repair_Perm: error event {Occurrence => poisson muh};
HW_Transient: out error propagation {Occurrence => fixed qh};
HW_Perm_Non_Detect: out error propagation {Occurrence=> fixed rh};
HW_Failed: out error propagation {Occurrence => fixed sh};end
forHardware;

error model implementation forHardware.general
transitions
HW_ErrorFree-[HW_Fault]-> HW_Activation_Fault;
HW_Activation_Fault-[HW_Trans_Fault]-> HW_Transient_Error;
HW_Activation_Fault-[HW_Perm_Fault]-> HW_Permanent_Error;
HW_Transient_Error -[HW_Repair_Trans]-> HW_Err_Free;
HW_Permanent_Error-[HW_Detection_Action]-> HW_Detection_Action_End;
HW_Detection_Action_End -[HW_Perm_Fault_Detect]-> HW_In_Repair;
HW_Detection_Action_End-[HW_Perm_Fault_Non_Detect]->
HW_Error_Non_Detect;
HW_Err_Non_Detect-[HW_Failure_Perceived]-> HW_In_Repair;
HW_In_Repair-[HW_Repair_Perm]-> HW_ErrorFree;
HW_Transient_Error -[out HW_Transient]-> HW_Transient_Error;
HW_Error_Non_Detect-[out HW_Perm_Non_Detect]->  HW_Error_Non_Detect;
HW_In_Repair-[out HW_Failed]-> HW_In_Repair;
end forHardware.general;
```


HW_Transient: qh
HW_Repair_Trans: tfh
HW_Transient_Error
HW_Trans_Fault: 1-ph
HW_ErrorFree
HW_Activation_Fault
HW_Fault: lh
HW_Perm_Fault: ph
HW_Repair_Perm: tmh
HW_Permanent_Error
HW_In_Repair
HW_Failed: sh
HW_Detection_Action: th
HW_Perm_Fault_Detect: dh
HW_Failure_Perceived: fph
HW_Detection_Action_End
HW_Perm_Fault_Non_Detect: 1-dh
HW_Error_Non_Detect
HW_Perm_Non_Detect: rh

```
error model forSoftware
features
SW_ErrorFree: initial error state;
SW_Activation_Fault, SW_Detection_Action_End, SW_Error_Non_Detected,
SW_Error_Detected, SW_Handling_End, SW_In_Restart: error state;
SW_Fault: error event {Occurrence => poisson ls};
SW_Detect_Action: error event {Occurrence => poisson ts};
SW_Detected: error event {Occurrence => fixed ds};
SW_Non_Detected: error event {Occurrence -> fixed 1-ds};
SW_Non_Detected_Disappear: error event {Occurrence => poisson dis};
SW_Non_Detected_Perceived: error event {Occurrence => poisson pcs};
SW_Handling: error event {Occurrence => poisson pis};
SW_Error_Temp: error event {Occurrence => fixed 1-ps};
SW_Error_Perm: error event {Occurrence => fixed ps};
SW_Restart: error event {Occurrence => poisson vs};
SW_Failed: out error propagation {Occurrence => fixed ps};
end forSoftware;

error model implementation forSoftware.general
transitions
SW_ErrorFree-[SW_Fault]-> SW_Activation_Fault;
SW_Activation_Fault-[SW_Detect_Action]-> SW_Detection_Action_End;
SW_Detection_Action_End -[SW_Detected]-> SW_Error_Detected;
SW_Detection_Action_End -[SW_Non_Detected]-> SW_Error_Non_Detected;
SW_Error_Non_Detected-[SW_Non_Detected_Disappear]->SW_ErrorFree;
SW_Error_Non_Detected-[SW_Non_Detected_Perceived]-
>SW_In_Restart;SW_Error_Detected-[SW_Handling]-> SW_Handling_End;
SW_Handling_End -[SW_Error_Temp]->SW_ErrorFree;
SW_Handling_End -[SW_Error_Perm]-> SW_In_Restart;
SW_In_Restart-[SW_Restart]-> SW_ErrorFree;
SW_In_Restart-[out SW_Failed]-> SW_In_Restart;
end forSoftware.general;
```

Next steps

- Read:
  - http://hbswk.hbs.edu/item/5699.html At the bottom of the page there is a place to download "Full Working Paper Text"
  - http://www.sei.cmu.edu/reports/07tn043.pdf
- Continue to expand your AADL model
  - Add at least one state machine
  - Define and bind to a platform
  - Identify at least one type of error and add to your model
- Create the DSMs for your architecture so far