Automatic Generation of Static Fault Trees from AADL Models

Anjali Joshi, Steve Vestal, Pam Binns

University of Minnesota and Honeywell Laboratories
Traditional Safety Analysis

System Safety Analysis is
- Based on Informal Specifications
- Highly Dependent on Skill of the Analyst
Model-Based Safety Analysis

Hardware

Data Bus
<<BusAccess>>

P1
Data Bus
<<BusAccess>>

P2
Data Bus

Processors

Software

P2
Input_1
<<Data>>
Output_1

Input_2
Output_2
<<Data>>

P1
Input_1
<<Data>>
Output_1

Input_2
Output_2
<<Data>>

Processes

System

June 27, 2007
Architecture Analysis & Design Language (AADL) Overview

- **Component**
  - **Type**
    - features (interaction points), properties, etc.
  - **Implementation**
    - subcomponents, connections, properties, etc.

- **Component Categories**
  - software, execution platform, composite

- **OSATE toolset**
  - Parsing, semantic checking
  - System Instance Model generation
    - Binding properties
Simple Dual Redundant System in AADL

```plaintext
system Duplex
end Duplex;

system implementation Duplex.Basic
subcomponents
  P1: processor Proc.Basic;
  P2: processor Proc.Basic;
  Data_Bus: bus PCI.Basic;
connections
  bus access Data_Bus -> P1.Data_Bus;
  bus access Data_Bus -> P2.Data_Bus;
end Duplex.Basic;

system Dual
end Dual;

system implementation Dual.Basic
subcomponents
  P1: process SW_PROC.Basic;
  P2: process SW_PROC.Basic;
connections
  C12: data port P1.Output_1 -> P2.Input_1;
  C21: data port P2.Output_1 -> P1.Input_1;
  C12_2: data port P1.Output_2 -> P2.Input_2;
  C21_2: data port P2.Output_2 -> P1.Input_2;
end Dual.Basic;
```

June 27, 2007
Example Error Model Specification

**error model Basic**

**features**
- **error_free:** *initial error state*
- **loss_of_availability, loss_of_integrity:** *error state*
- **fail_stop, fail_babble:** *error event* 
- **loss_of_data, corrupted_data:** *in out error propagation*

*end Basic;*

**error model implementation Basic.Hardware**

**transitions**
- **error_free** -[**fail_stop, in loss_of_data**]--> **loss_of_availability**
- **error_free** -[**fail_babble, in corrupted_data**]--> **loss_of_integrity**
- **loss_of_availability** -[**fail_babble**]--> **loss_of_integrity**
- **loss_of_integrity** -[**in corrupted_data, out corrupted_data**]--> **loss_of_integrity**

*end Basic.Hardware;*
Error Model Association via Annex Clause & Properties

process implementation SW_Proc.Basic
annex error_model {**
    model => Basic.Software;
    occurrence => fixed 1E-4 applies to error fail_stop;
guard in =>
    mask when Input_1[error_free] or Input_2[error_free],
corrupted_data when Input_1[loss_of_integrity] and Input_2[loss_of_integrity],
    loss_of_data when others applies to Input_1, Input_2;
**};
end SW_Proc.Basic;

system implementation Dual.Basic
subcomponents
    P1: process SW_Proc.Basic;
...
annex error_model {**
    model => Basic.Software;
    model_hierarchy => Derived;
derived_state_mapping =>
    error_free when P1[error_free] or P2[error_free],
    loss_of_availability when P1[loss_of_availability] or P2[loss_of_availability],
    loss_of_integrity when others;
    report => loss_of_availability, loss_of_integrity;
**};
end Dual.Basic;
Error Propagation Rules

- **Connection to** In Ports **of a Component**
- **Bus** to all Connections bound to it
- **Processor** to required bus
CAFTA Fault Tree Generation from AADL Models
System Instance Error Model Extraction – Directed Graph Representation
High-level Fault Tree Generation Approach

- Identify all the “Report” properties defined
- For each Error State or Out Error Propagation defined in the Report property
  - Generate fault tree by traversing the DG based on
    - Error model (hierarchy, guard, etc.) properties
    - Error propagation sources
  - Break cycles where necessary
  - Share sub-trees where possible

- Optimizations
  - Removing redundant gates and fault tree nodes
  - Sharing sub-trees
Generate Output CAFTA Fault Tree
Summary

- Automatic generation of CAFTA fault trees

Advantages –
- Consistent
- Mapping between fault trees & architecture
- Identify common modes of failure

Challenges
- Aggressive Optimizations needed
  - Sharing of sub-trees
  - Pruning of redundant fault trees
- Guard against missing out error annotations
Contact Information

Anjali Joshi
University of Minnesota, Minneapolis
ajoshi@cs.umn.edu

Steve Vestal, Pam Binns
Honeywell Laboratories, Minneapolis
{Steve.Vestal, Pam.Binns}@honeywell.com
System Instance Error Model Extraction – Directed Graph Representation

Component Node
- **SW.P1**
  - Basic.Software
  - ErrorModelProperties...
  - ErrorPropSrcs: SW.C21, SW.C21_2, HW.P1

Connection Node
- **SW.C21**
  - Basic.Software
  - ErrorModelProperties...
  - ErrorPropSrcs: SW.P2, HW.Data_Bus

Component Node
- **HW.P1**
  - Basic.Hardware
  - ErrorModelProperties...
  - ErrorPropSrcs: HW.Data_Bus

Component Node
- **HW.Data_Bus**
  - Basic.Hardware
  - ErrorModelProperties...
  - ErrorPropSrcs: HW.P1, HW.P2

DualSystem.Basic

HW
- HW.Data_Bus
- HW.P1
- HW.Data_Bus

SW
- C21
- SW.P1
- SW.Data_Bus
- SW.Data_Bus
AADL Error Model Annex Overview

- Error Model Annex
  - Specification of error models and their association to AADL components via AADL annex sub-clause
  - Defines Error Model properties
    - Filtering and masking of error propagations
      - Guard_In, Guard_Out properties
    - Hierarchical composition of sub-component error models
      - Model_Hierarchy, Derived_State_Mapping properties
    - Occurrence properties
  - Defines error propagation rules
    - Error propagation path from bus to connected processor
    - Error propagation paths between components sharing a port connection