

Model-Driven Engineering (MDE)

- **Principles**
 - Models as means for understanding systems
 - Model transformations
 - Metamodels
 - Industry standards
- **Languages for MDE**
 - UML: OMG standard, different types of diagrams, domain-specific profiles
 - SysML: standard UML profile, system-oriented
 - EastADL: automotive domain, aligned on SysML and AUTOSAR standard
 - AADL: SAE standard for embedded safety-critical real-time systems
 - Considered by aerospace and automotive industries
 - Architectural refinement during development cycle
 - Analyses of quality attributes (properties and annexes)

BACKGROUND

MODELING

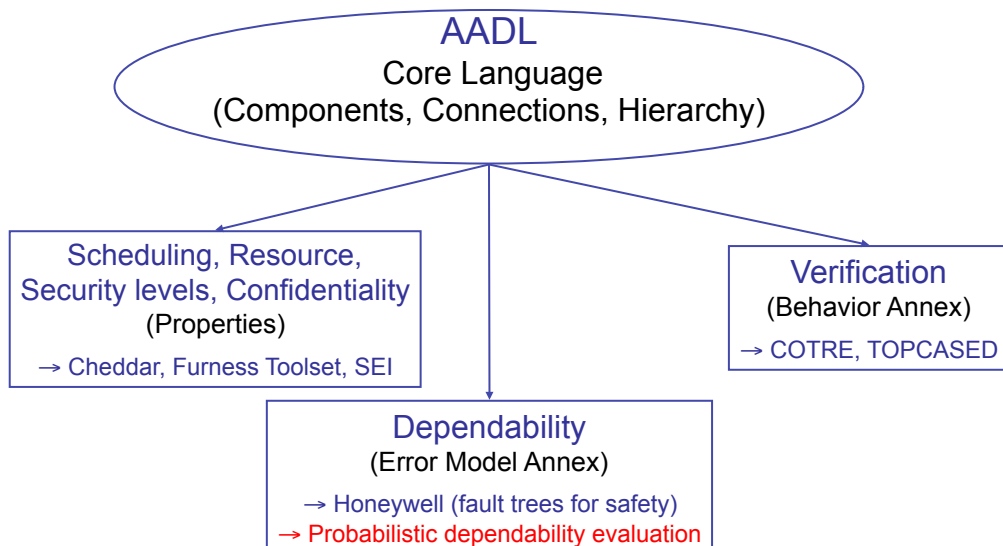
TRANSFORMATION

APPLICATION

CONCLUSION

3

AADL and Analysis



BACKGROUND

MODELING

TRANSFORMATION

APPLICATION

CONCLUSION

4

Probabilistic Dependability Evaluation

- Useful during design, implementation and integration phases
- Helps making decisions about architecture and fault-tolerance, dependability requirements validation
- Model-based
 - Choice of measures (based on system requirements)
 - Model construction (stochastic properties, dependencies, coverage, ...)
 - Non-state space: fault trees, reliability block diagrams
 - State space: Markov chains, GSPN
 - Model processing (measures, sensitivity analyses)

GSPN

BACKGROUND

MODELING

TRANSFORMATION

APPLICATION

CONCLUSION

5

Dependability Evaluation of Complex Systems

- Complexity due to stochastic dependencies
 - Architectural (structural, functional, fault-tolerance)
 - Maintenance and recovery
- AADL
 - + Explicit assignment of responsibilities and behaviors to components
 - + Several analyses on the same architectural model facilitate tradeoffs
 - No direct means for obtaining dependability measures
- GSPN
 - + Modularity and hierarchy
 - + Formal verification
 - + Automatically converted to Markov chains
 - + Supported by many dependability evaluation tools
 - Drawbacks: difficulty of model construction, dedicated model

BACKGROUND

MODELING

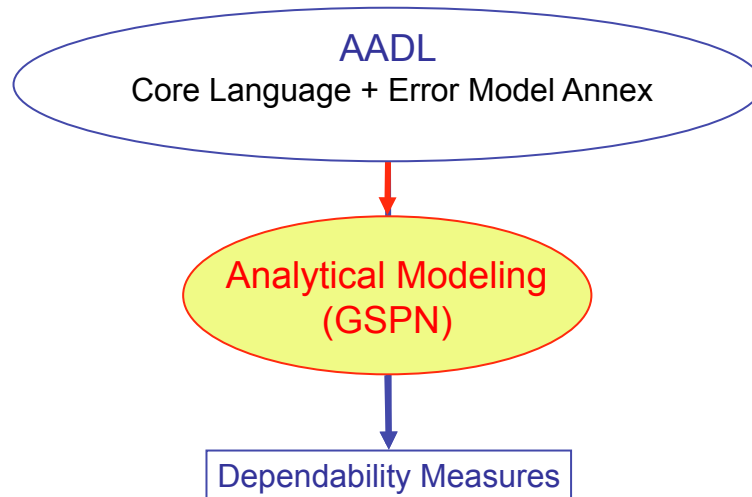
TRANSFORMATION

APPLICATION

CONCLUSION

6

Our Objective



BACKGROUND

MODELING

TRANSFORMATION

APPLICATION

CONCLUSION

7

Contributions

1. AADL Dependability Modeling

→ Proposal of modeling **guidelines** and adaptive modeling **patterns** for classical fault-tolerant architectures

Facilitate model reuse
Master complexity due to dependencies

2. Model Transformation

→ exhaustive set of transformation rules from AADL to GSPN
→ proof of feasibility through tool implementation

Enable obtaining dependability measures from AADL model

3. Case Study

→ a subsystem of the French Air Traffic Control System (CAUTRA)

BACKGROUND

MODELING

TRANSFORMATION

APPLICATION

CONCLUSION

8

Outline

- ❑ Dependability Modeling with AADL
- ❑ Model Transformation
- ❑ Case Study
- ❑ Conclusion

9

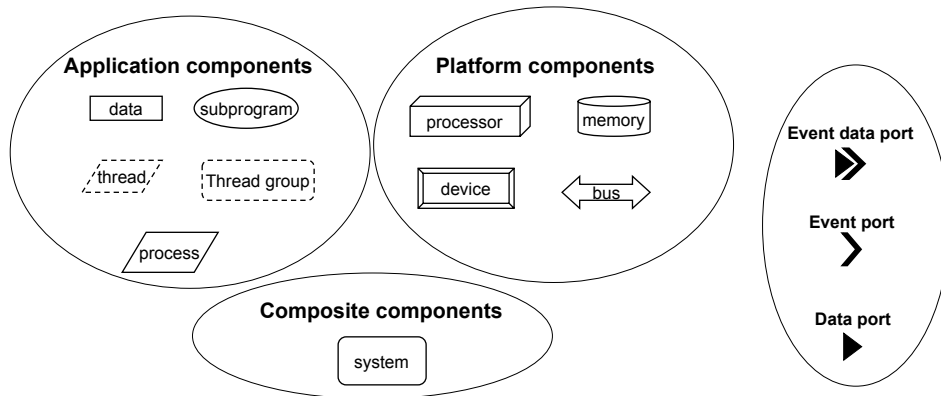
Outline

- ❑ **Dependability Modeling with AADL**
 - Existing AADL
 - Modeling Dependencies
 - Patterns
- ❑ Model Transformation
- ❑ Case Study
- ❑ Conclusion

10

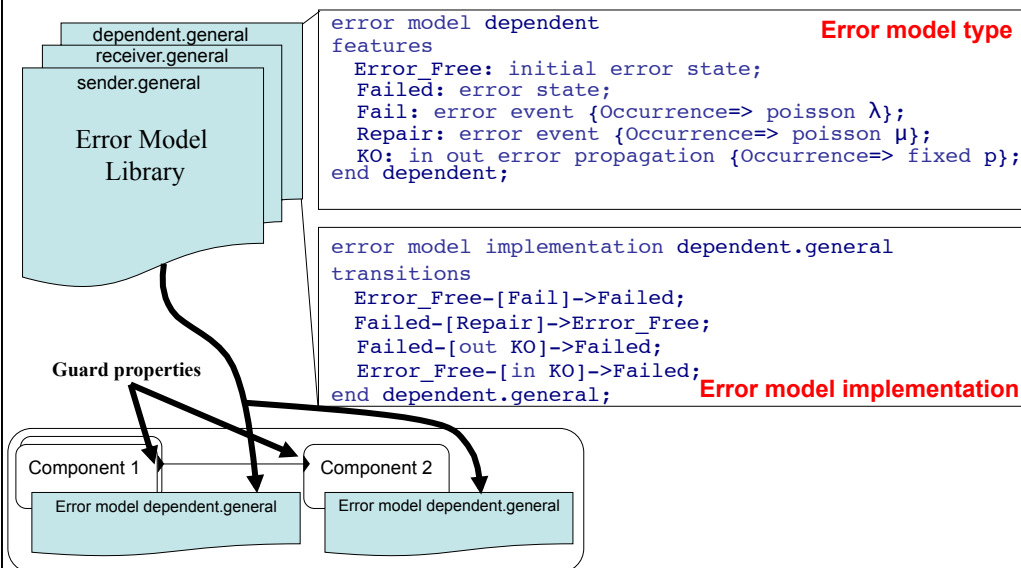
AADL Core Language

- Hierarchical composition of interconnecting components



- Reconfiguration through operational modes

AADL Error Model

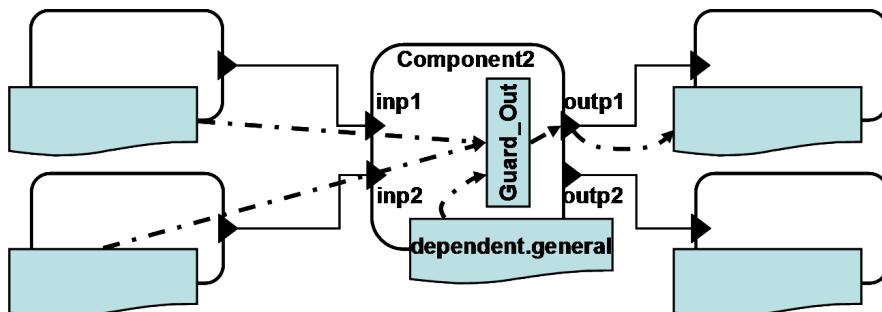


Operational Modes & Error States

- Error states: result from occurrences of error events
- Operational modes: represent operational states of the system that may be totally independent of the occurrence of error events
 - *Phased-mission systems*: configurations representative of different phases in a mission
 - *Mode-specific fault-tolerant system configurations*: the fault tolerance strategy of the system or parts of the system
- Mode transitions: dynamic operational behavior
- A mode transition may be triggered by:
 - out event port of a subcomponent of the component declaring the modes
 - in event port of the component itself
 - a local event port

Connecting Error States to Modes

- Guard_Event property
 - Trigger mode changes depending on error state configurations and error propagations occurrences
 - applies to event ports



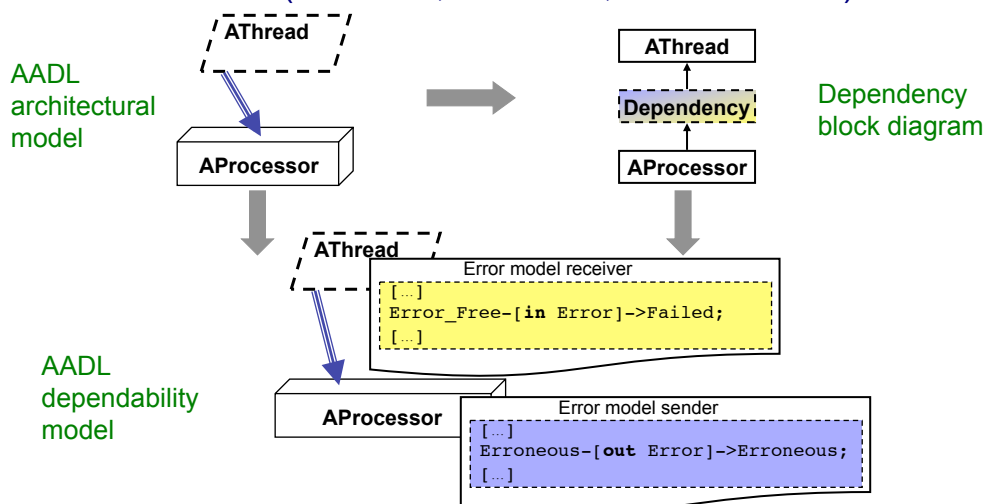
Outline

- ❑ **Dependability Modeling with AADL**
 - Existing AADL
 - **Modeling Dependencies**
 - Patterns
- ❑ Model Transformation
- ❑ Case Study
- ❑ Conclusion

15

Modeling Dependencies

- Architectural (structural, functional, fault-tolerance)



BACKGROUND

MODELING

TRANSFORMATION

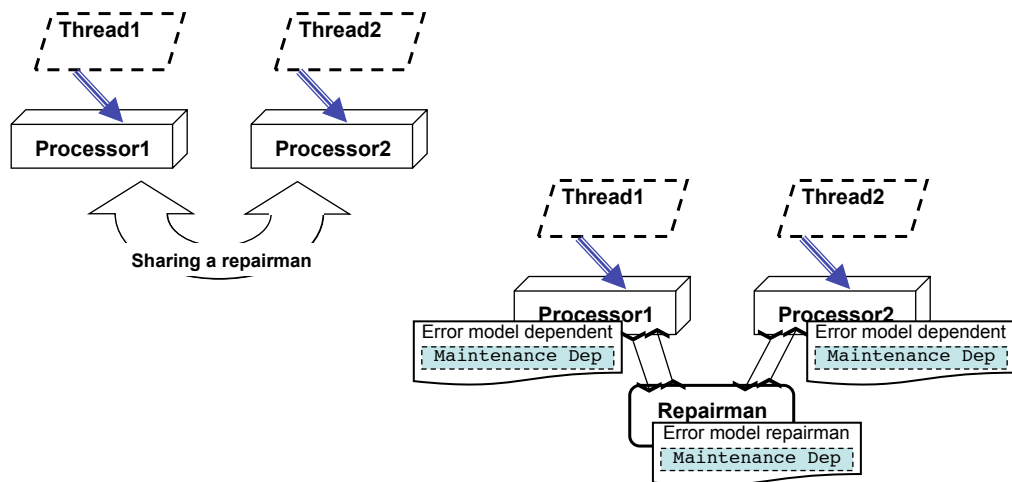
APPLICATION

CONCLUSION

16

Modeling Dependencies

- Maintenance and recovery



BACKGROUND

MODELING

TRANSFORMATION

APPLICATION

CONCLUSION

17

Outline

- **Dependability Modeling with AADL**
 - Existing AADL
 - Modeling Dependencies
 - **Patterns**
- Model Transformation
- Case Study
- Conclusion

18

Reusability and Patterns

- Several levels of reusability in AADL
 - Architectural models
 - Error models
 - Dependability models
- Patterns for fault-tolerant architectures

HW	SW
<ul style="list-style-type: none"> • Cold standby • Warm standby • Hot standby 	<ul style="list-style-type: none"> • Recovery block
<ul style="list-style-type: none"> • Active dynamic redundancy with self-checking components 	<ul style="list-style-type: none"> • 2 self-checking programming
<ul style="list-style-type: none"> • 3-modular redundancy 	<ul style="list-style-type: none"> • 3-version programming

BACKGROUND

MODELING

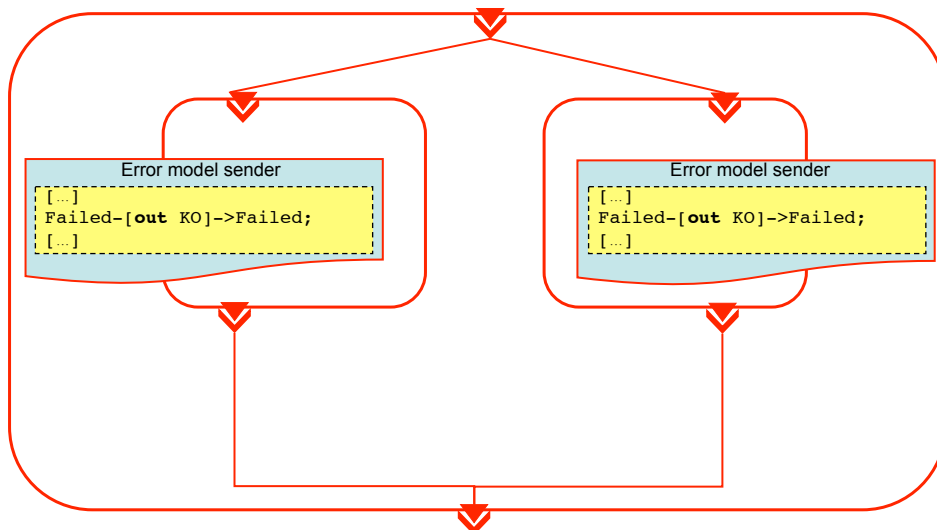
TRANSFORMATION

APPLICATION

CONCLUSION

19

Generic Pattern for Duplex System



BACKGROUND

MODELING

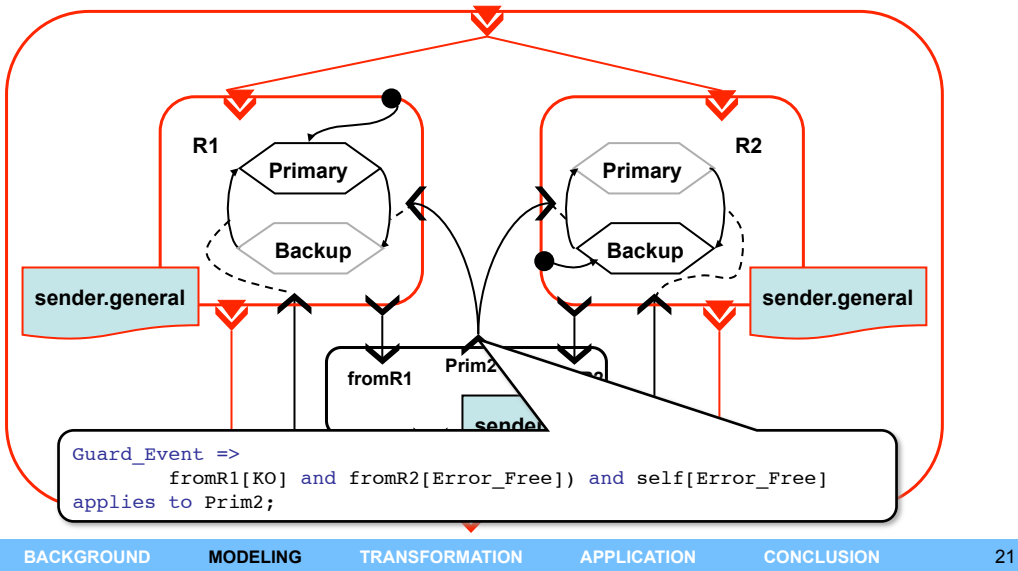
TRANSFORMATION

APPLICATION

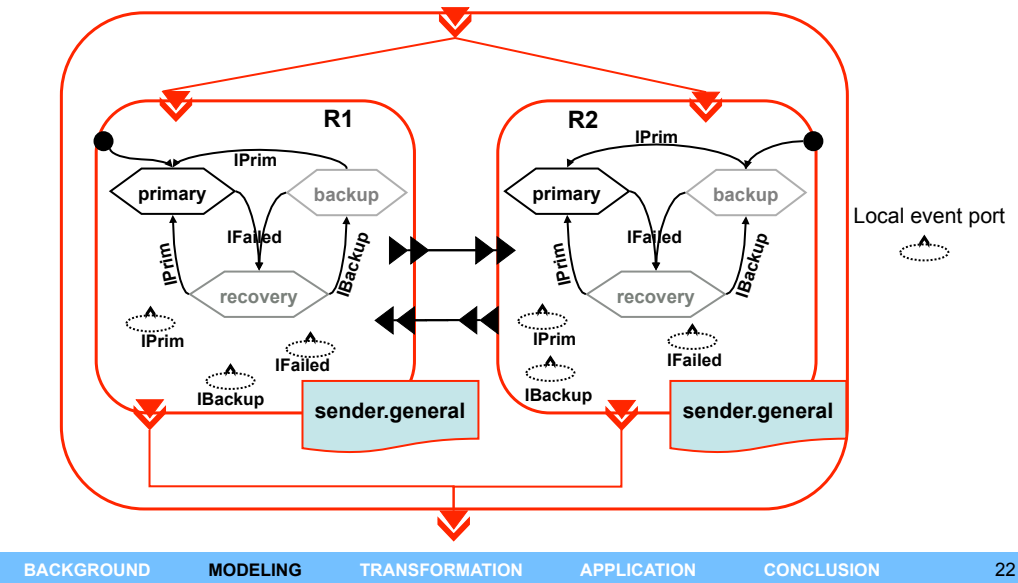
CONCLUSION

20

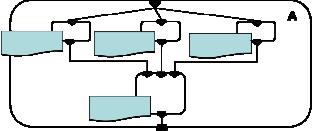
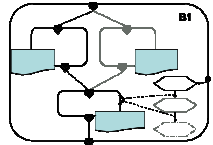
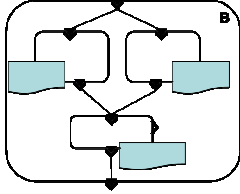
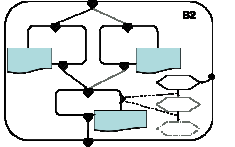
Pattern Refinement 1: HW Hot Standby Sparring



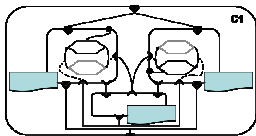
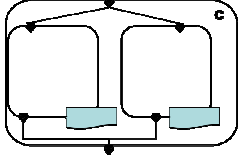
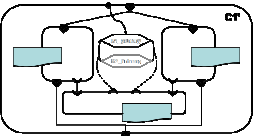
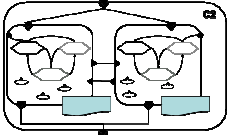
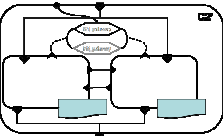
Pattern Refinement 2: Active Dynamic Redundancy with Self-checking Replicas



Overview of the Fault Tolerance patterns

HW FT	SW FT	Pattern	Ancestor pattern
N-Modular Redundancy (3 replicas)	N-Version Programming (3 replicas)		
Cold Standby Sparing	Recovery Block		
Warm Standby Sparing			

23

Hot Standby Sparing			
			
Active Dynamic with Self-Checking Components	N Self-Checking Programming		
			

24

Outline

- Dependability Modeling with AADL
- **Model Transformation**
- Case Study
- Conclusion

25

Overview

- Complete set of transformation rules: systematic, formalized

Category	AADL Constructs
Independent components	states and events
Basic elements for dependencies	out, in, in-out propagations
Propagation filtering mechanisms	Guard_In, Guard_Out
Connecting error states to modes	Guard_Event, Guard_Transition, activate/deactivate
Hierarchy	Derived error models
Architecture configurations	Modal architectural elements

- Principle: transformation guided by dependencies
- Modular GSPN: Independent component → block, Dependency → block

BACKGROUND

MODELING

TRANSFORMATION

APPLICATION

CONCLUSION

26

Overview

- Complete set of transformation rules: systematic, formalized

Category	AADL Constructs
Independent components	states and events
Basic elements for dependencies	out, in, in-out propagations
Propagation filtering mechanisms	Guard_In, Guard_Out
Connecting error states to modes	Guard_Event, Guard_Transition, activate/deactivate
Hierarchy	Derived error models
Architecture configurations	Modal architectural elements

- Principle: transformation guided by dependencies
- Modular GSPN: Independent component → block, Dependency → block

BACKGROUND

MODELING

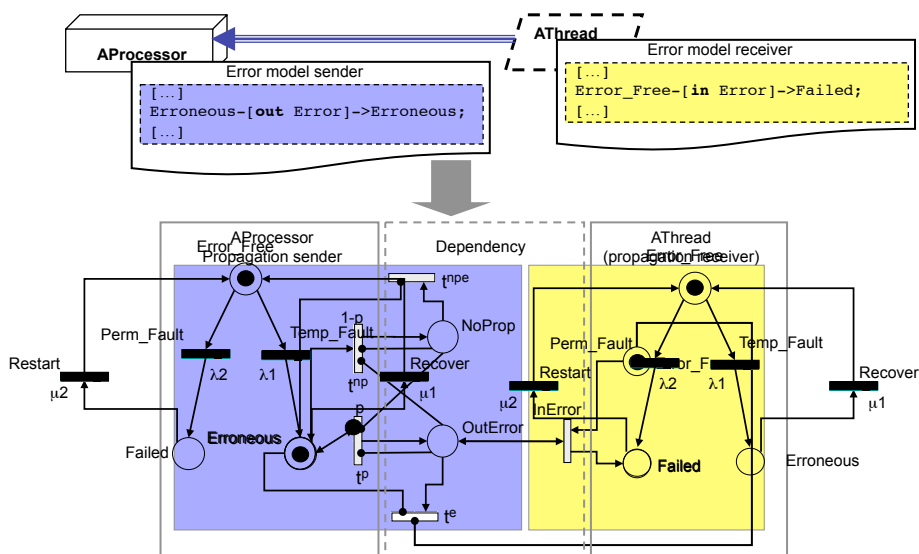
TRANSFORMATION

APPLICATION

CONCLUSION

27

In - Out Propagations



BACKGROUND

MODELING

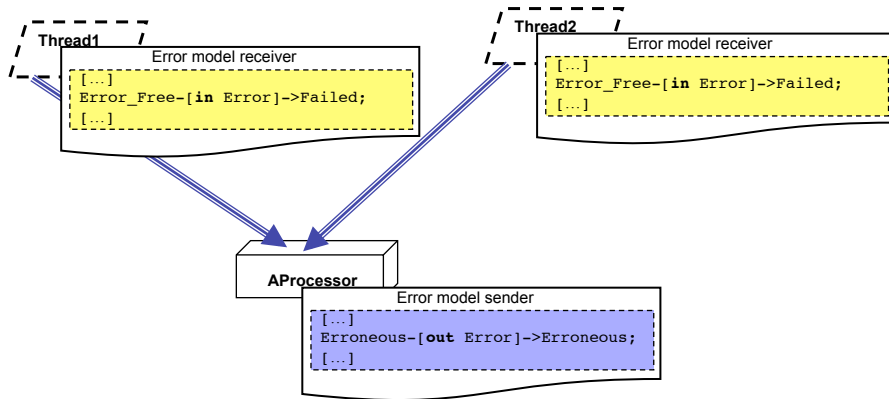
TRANSFORMATION

APPLICATION

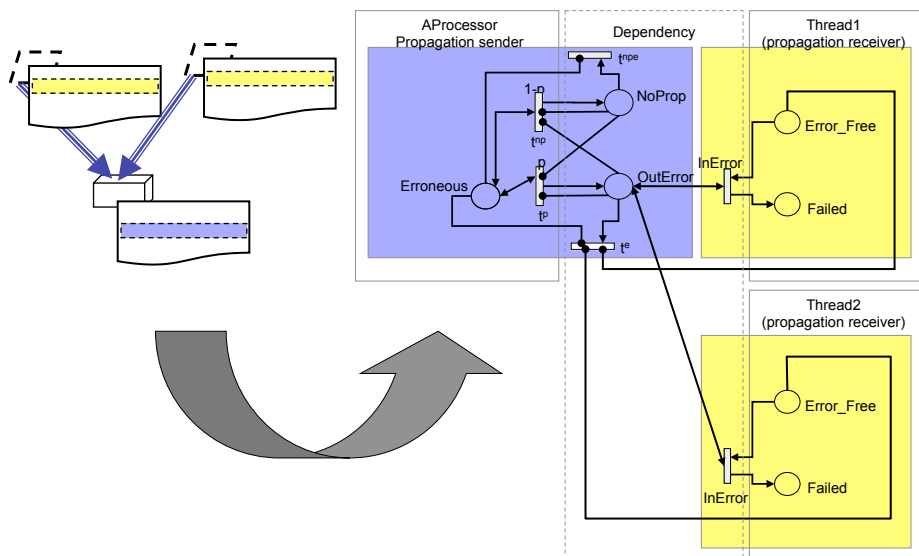
CONCLUSION

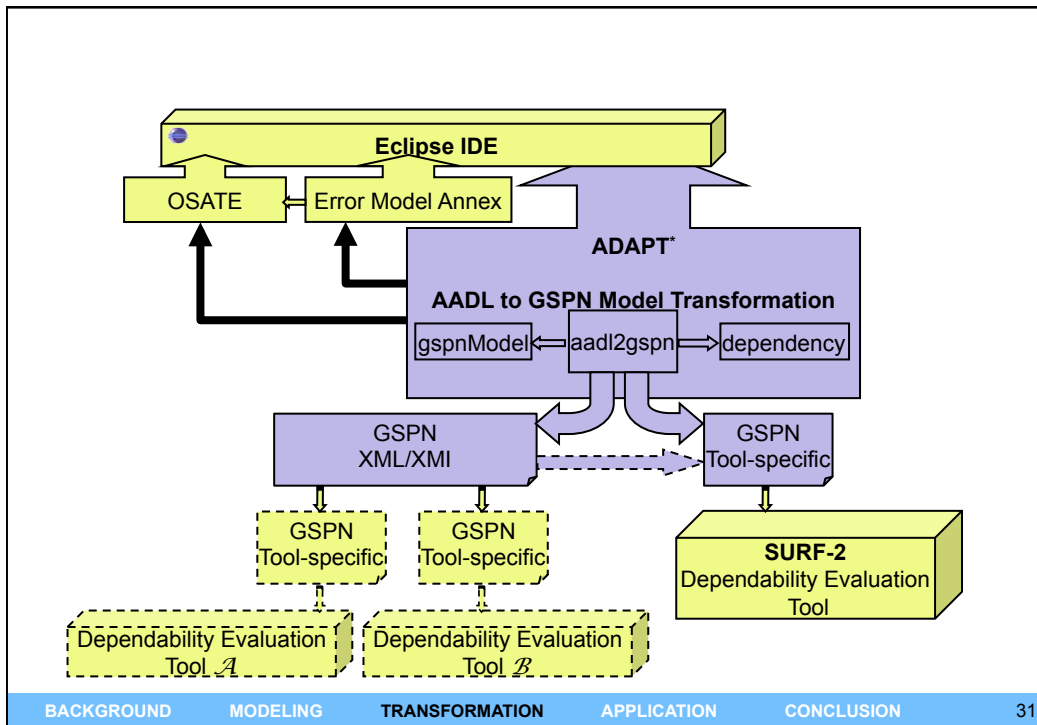
28

Several dependencies



Modularity and Reuse



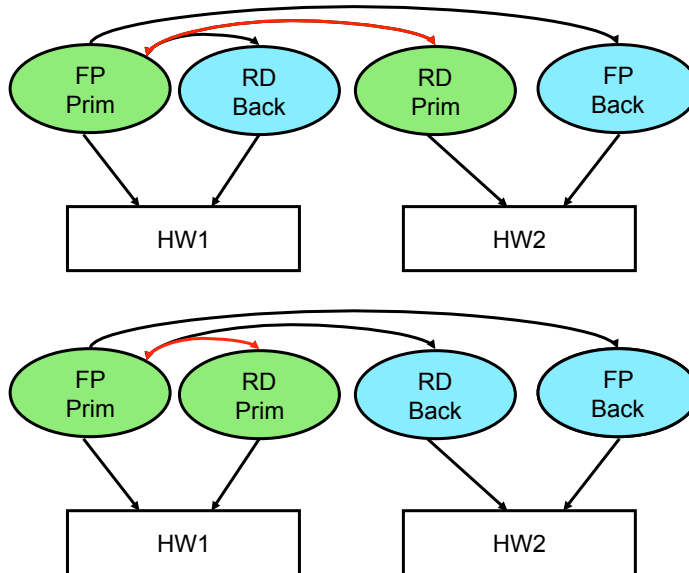


Outline

- Background
- Dependability Modeling with AADL
- Model Transformation
- **Case Study**
- Conclusion

Specifications - Subsystem of CAUTRA

- Measure of interest: availability
- Two dual-redundant fault tolerant software units:
 - FP: flight plans
 - RD: radar data
- Two processors
- Two architectural configurations



BACKGROUND

MODELING

TRANSFORMATION

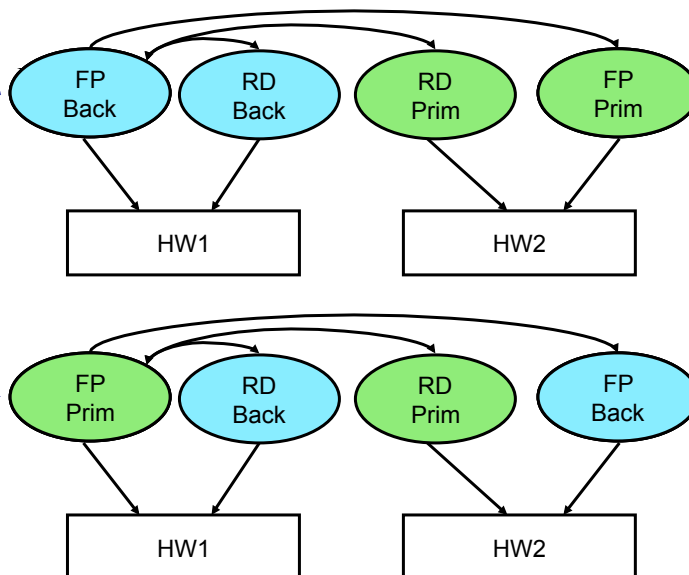
APPLICATION

CONCLUSION

33

Specifications - Subsystem of CAUTRA

- Measure of interest: availability
- Two dual-redundant fault tolerant software units:
 - FP: flight plans
 - RD: radar data
- Two processors
- Two fault tolerance policies for FP
 - FT1: Keep roles
 - FT1': Switch roles



BACKGROUND

MODELING

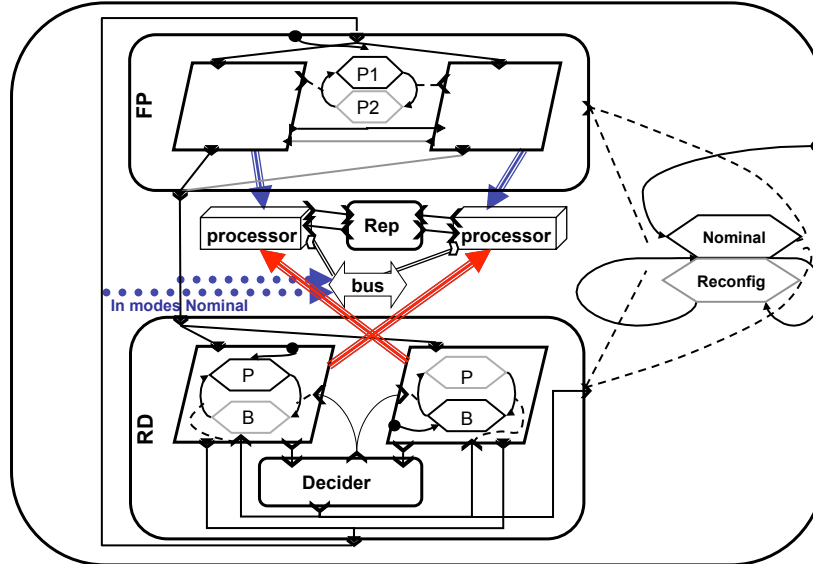
TRANSFORMATION

APPLICATION

CONCLUSION

34

AADL Architectural Models (Configuration 1)



BACKGROUND

MODELING

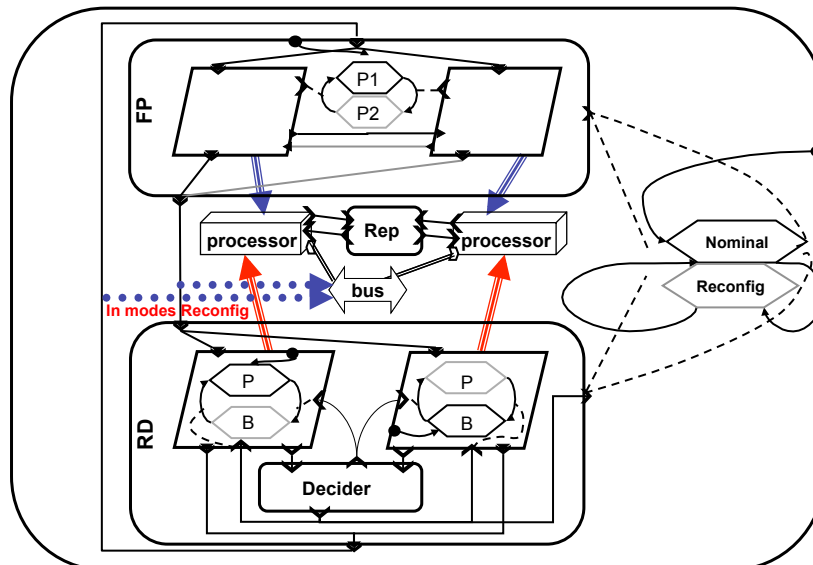
TRANSFORMATION

APPLICATION

CONCLUSION

35

AADL Architectural Models (Configuration 2)



BACKGROUND

MODELING

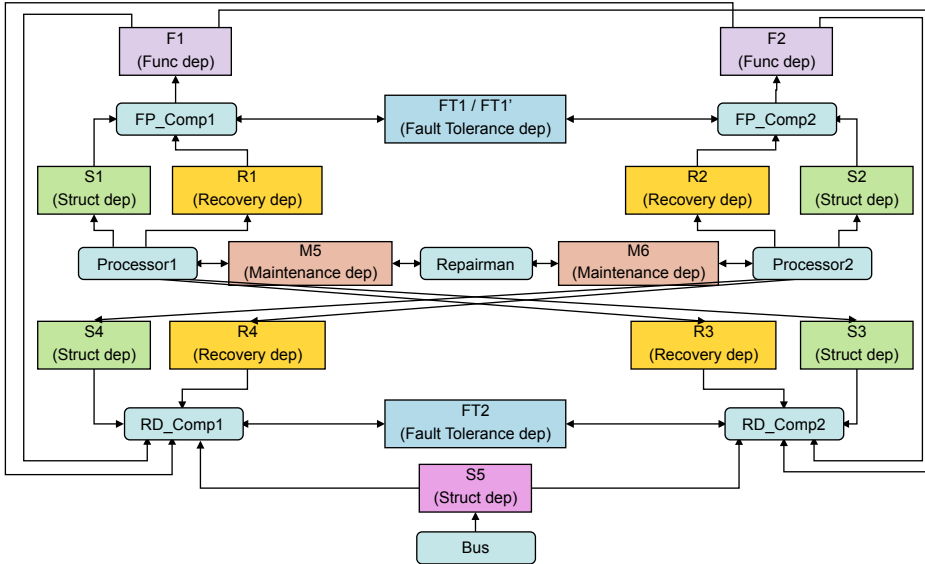
TRANSFORMATION

APPLICATION

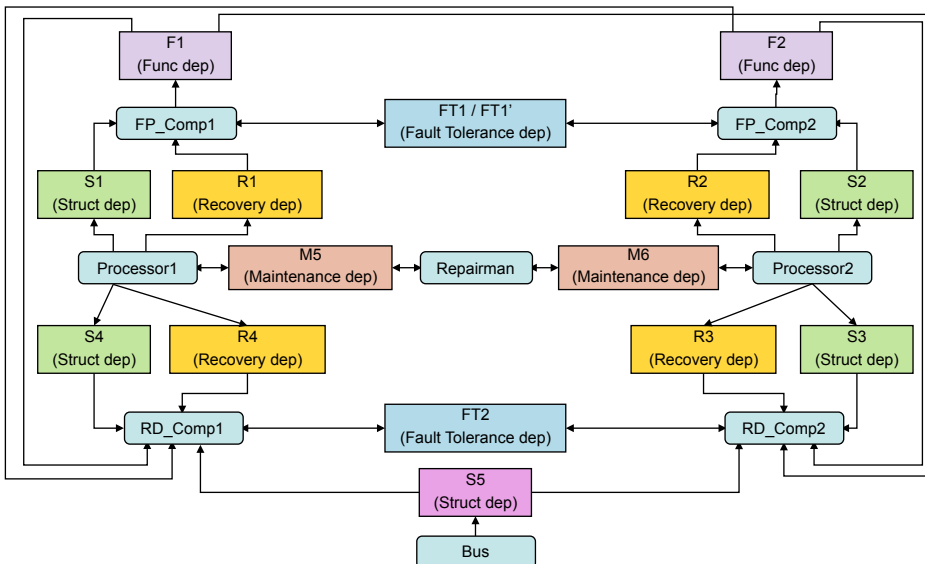
CONCLUSION

36

Dependency Block Diagram (Configuration 1)



Dependency Block Diagram (Configuration 2)



Case Study Summary

- AADL dependability model construction: 10 iterations
- GSPN: 24 subnets
 - 8 component subnets
 - 16 dependency subnets
 - 91 places, 205 transitions
- Model reuse
 - Software replicas
 - Processors
 - Identical dependencies between different components
 - 2 fault tolerance patterns (with / without customization)

BACKGROUND

MODELING

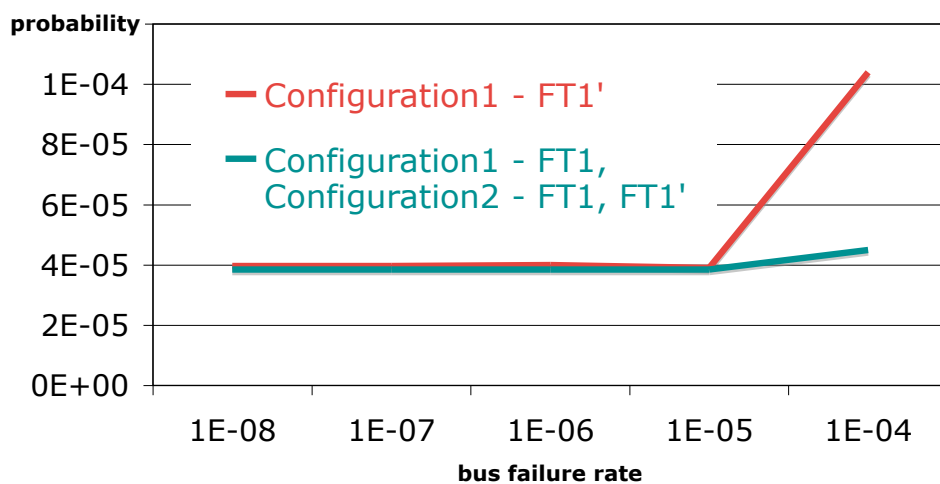
TRANSFORMATION

APPLICATION

CONCLUSION

39

Example of Results (Unavailability)



BACKGROUND

MODELING

TRANSFORMATION

APPLICATION

CONCLUSION

40

Outline

- Background
- Dependability Modeling with AADL
- Model Transformation
- Case Study
- **Conclusion**

41

Conclusion

- **Structured modeling approach guided by dependencies**
 - favors reuse and evolution
- **Fault tolerance modeling patterns**
 - favors reuse
 - enhances readability and understandability
- **Model transformation from AADL to GSPN**
 - designed to be automated, formalized
 - implementation of ADAPT tool
- **Case study issued from real-life application**
- **Proposals of evolution of AADL standard**
 - partly integrated in current standard version

BACKGROUND

MODELING

TRANSFORMATION

APPLICATION

CONCLUSION

42