

Automating Timing and Safety Analyses from Architecture Specifications

Steve.Vestal@Honeywell.com

9 February 2005



Honeywell

Outline

- Overview
- Case Studies

AADL and MetaH

AADL is the SAE standard Architecture Analysis & Design Language, including a standard UML profile and XML schema. The standardization committee has fairly wide government and industry representation, e.g.

US Army
US Navy
NATO
ESA

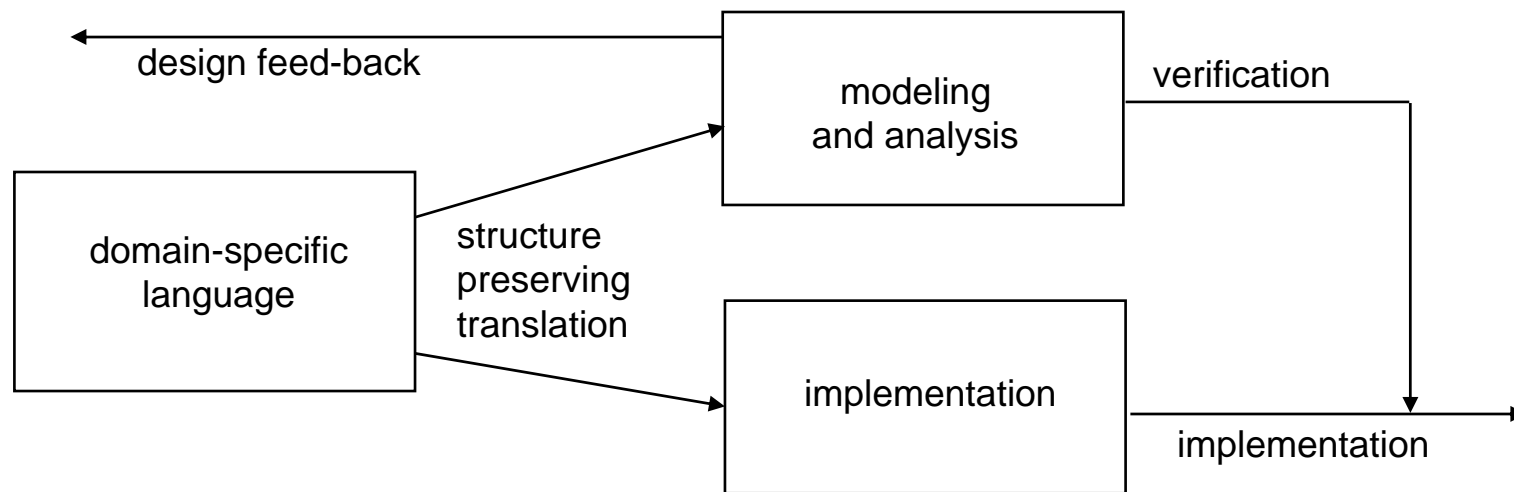
Airbus
Dassault
Lockheed-Martin
Raytheon

Smiths
Rockwell/Collins
Honeywell

MetaH, developed at Honeywell Labs, was the starting point for the AADL standard. The MetaH toolset automates a variety of modeling, analysis, implementation, and verification activities in an integrated and traceable way.

Honeywell

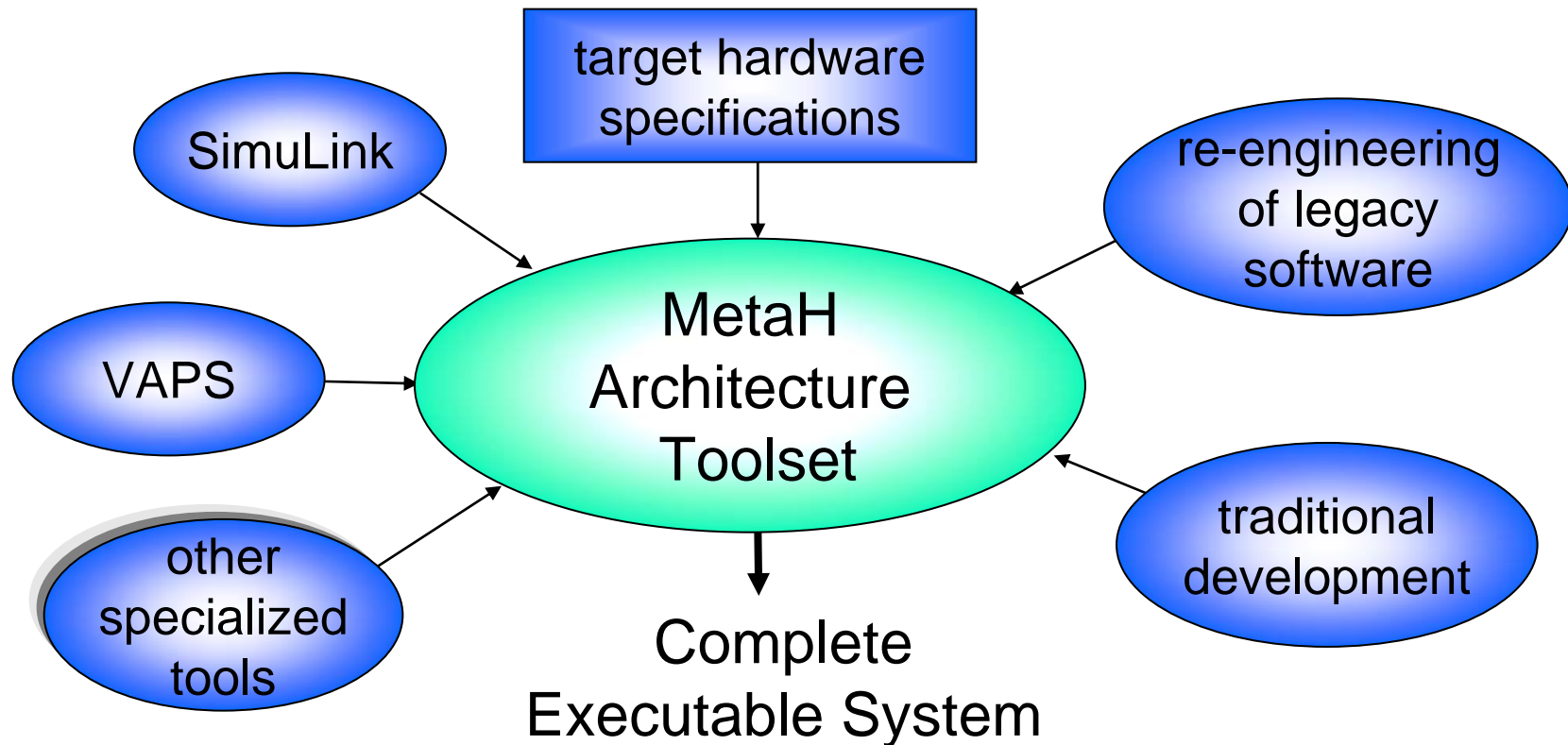
Integrated and Traceable Specification, Analysis, Implementation, Verification



- **Improve quality** of system design through more accurate and rapid design-time evaluations and trade-offs
- **Increase assurance** the implementation will behave the way the analyses say it will behave due to co-generation of models
- **Decrease effort** for modeling, implementation, debugging and verification through automation and defect reduction

Honeywell

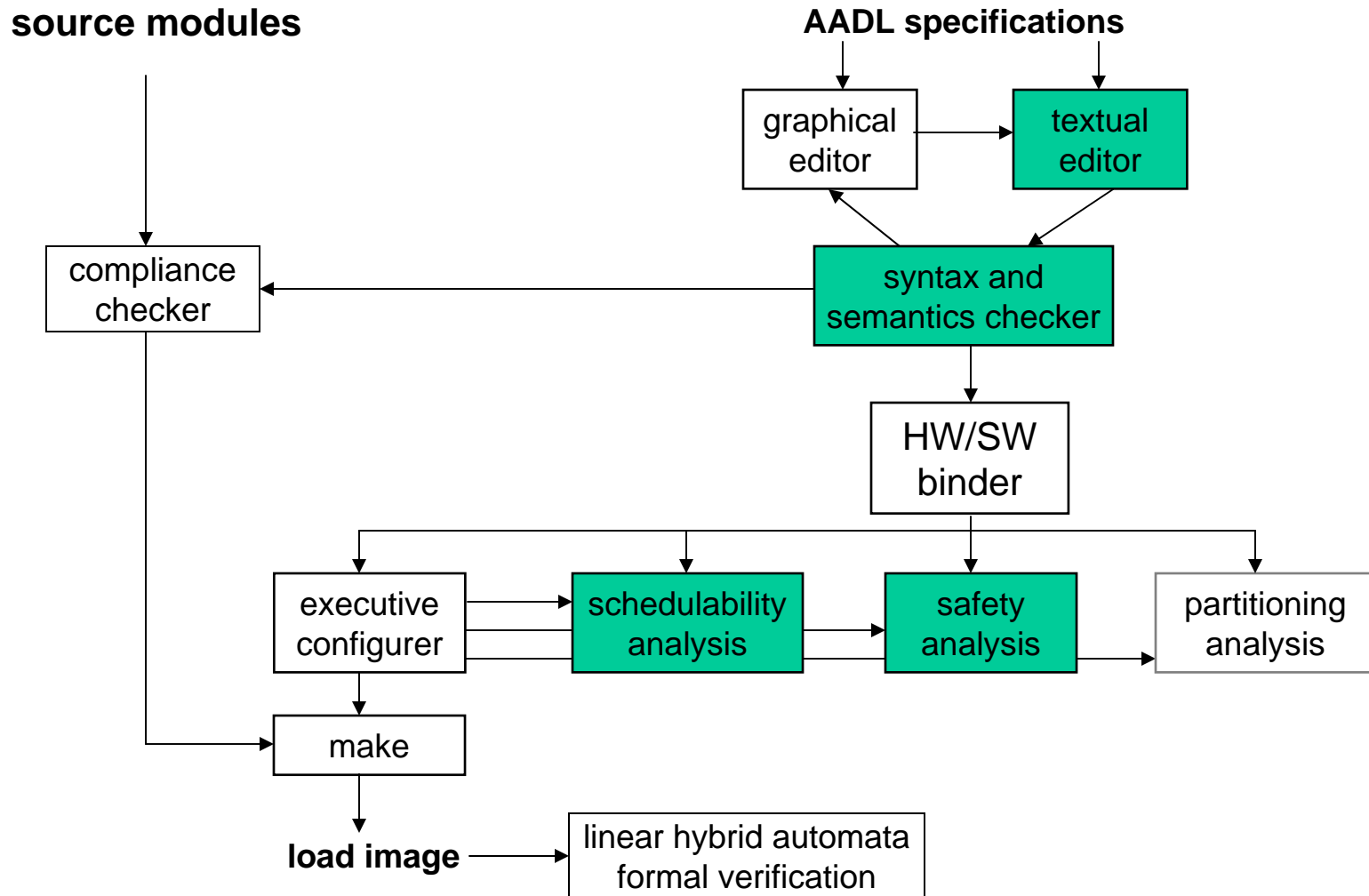
Modular Avionics Architecting & Integration



A Mathworks-like toolset for embedded computer system architects and integrators.

Honeywell

MetaH Toolset Functions



Honeywell

Compositional Specification and Modeling

Develop and use compositional modeling technologies:

- Specify models for individual components
- Automatically generate system models from the specification of how components are assembled to form an overall architecture.

Do this in a way that provides:

- Reusable component models
- Easily reconfigured architectures
- Structured traceability between specifications and models
- More tractable analysis using decomposition approaches
- Abstraction and mixed fidelity modeling and analysis

Select and integrate a set of modeling methods suited to the product family architectures and requirements.

Honeywell

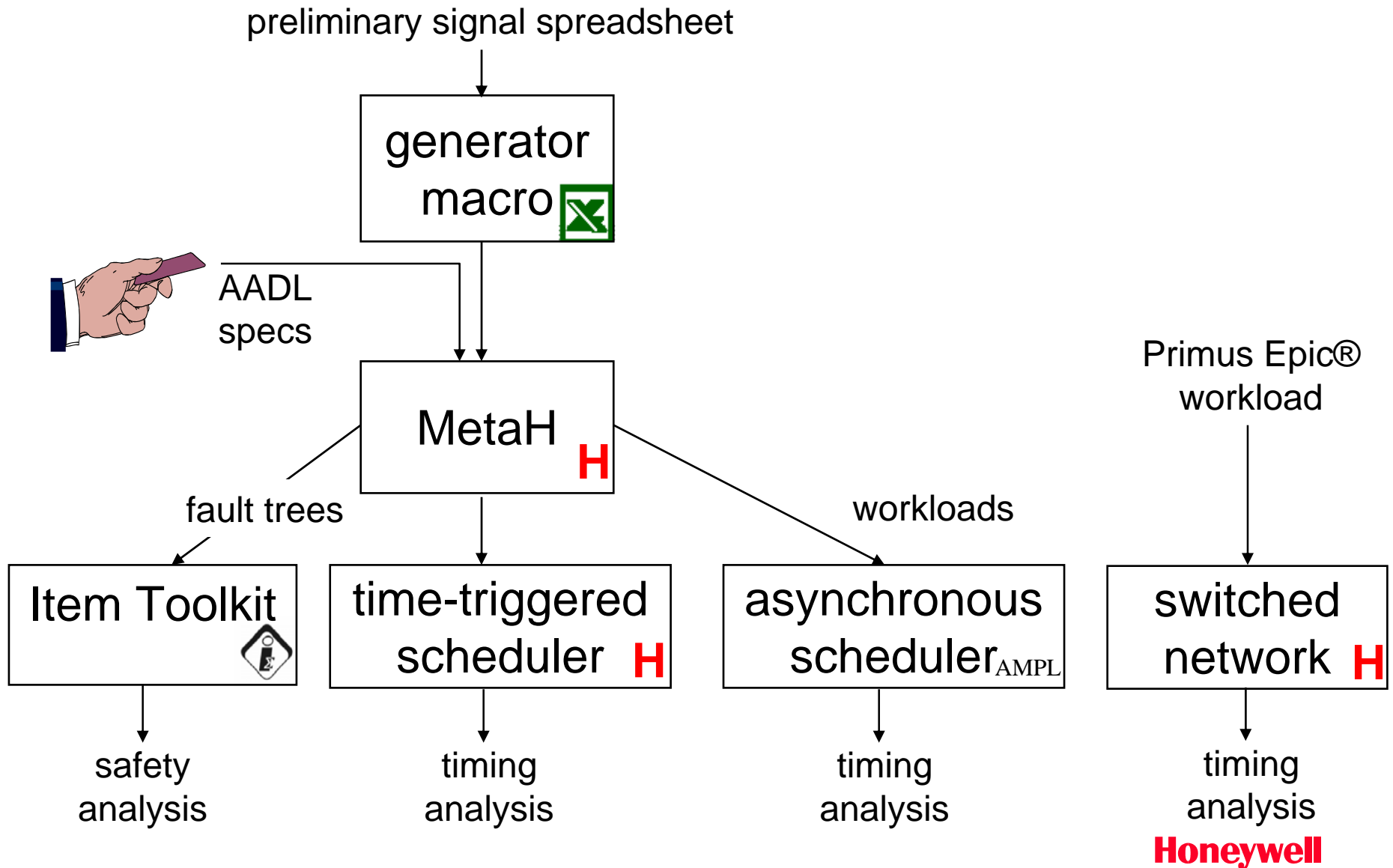
Goals of 7E7 AADL Project

In 1Q 2003 a project was undertaken to specify, model and analyze some then-candidate architectures for the 7E7 Common Computing System. The goals were to

- assess whether the SAE Architecture Analysis & Design Language can capture sufficient detail to automatically generate timing and safety models for a system of real-world complexity.
- assess whether automatic generation and analysis of timing and safety models is feasible and computationally tractable for a system of real-world size.
- provide useful preliminary feed-back to the baseline architecture team.

Honeywell

Tools and Data Flows



Architecture Specification Files

Used textual representation (graphical for slides only)

- easier to generate
- one less tool to modify

Specification divided into 5 files:

File Name	Lines	Produced By	Description
types.mh	75	hand	signal types & attributes
errmod.mh	75	hand	error models
hw.mh	275	hand	hardware architecture
sw.mh	38000	macro	functions (sw) architecture
ccs.mh	35	hand	required "main" declaration

Honeywell

Hardware architecture specification hand-written, based on spreadsheet and slides

- 8 central processors (CPs)

- 18 I/O processors (IOPs)

hw1.mh specified a switched network architecture

- 12 switches

- 62 point-to-point cables, mix of 10Mbps and 100Mbps

- Used for safety analysis

- Used for globally asynchronous end-to-end timing analysis

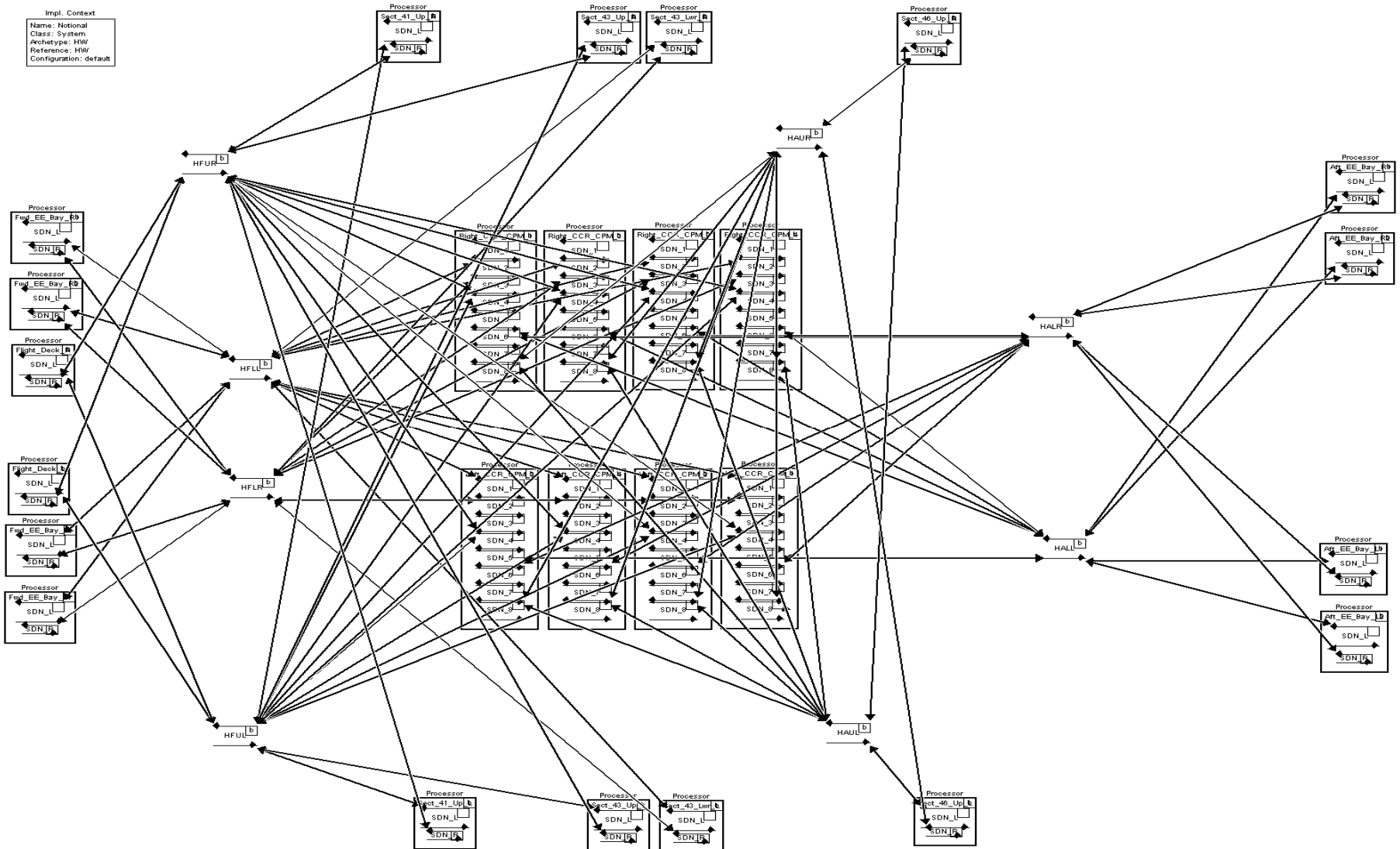
hw2.mh specified a time-triggered architecture

- 8 multi-drop time-triggered busses, 10Mbps

- Used for globally time-triggered end-to-end timing analysis

- Used for globally asynchronous end-to-end timing analysis

Example Graphical MetaH Hardware Specification



Honeywell

Function/application specifications were generated from a preliminary spreadsheet listing signal data.

Total of 1322 signals to/from 40 functions (includes redundant sensors/signals).

Much data we needed was missing, we wrote the generator macro to fill this in speculatively, e.g.

- redundancy management templates

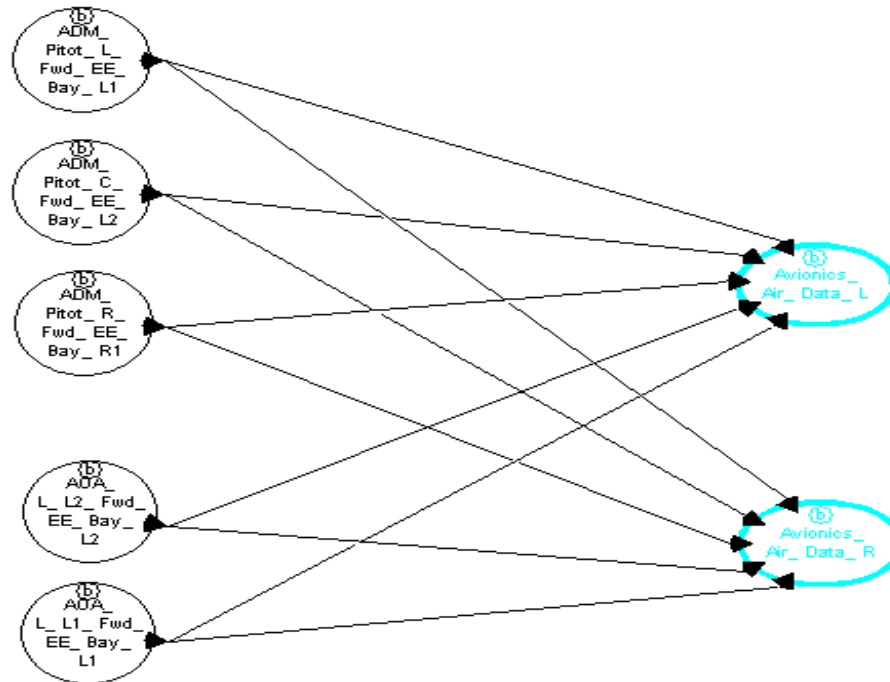
- process rates and execution times (using existing product data)

- partition bindings

Using a dual redundancy template for all functions, total of 2644 signals to/from 80 partitions.

Example Graphical MetaH Function Specification

Impl. Context
Name: Notional
Class: Macro
Archetype: Avionics_Air_Data
Reference: Avionics_Air_Data
Configuration: default



Honeywell

AADL Timing Specifications

AADL specifications capture timing and resource requirements, e.g.

- Periods
- Deadlines
- Execution Times
- Transmission Times

Additional data for a particular tool or design choice may be specified using additional non-core properties (e.g. eventual AADL ARINC 653 annex).

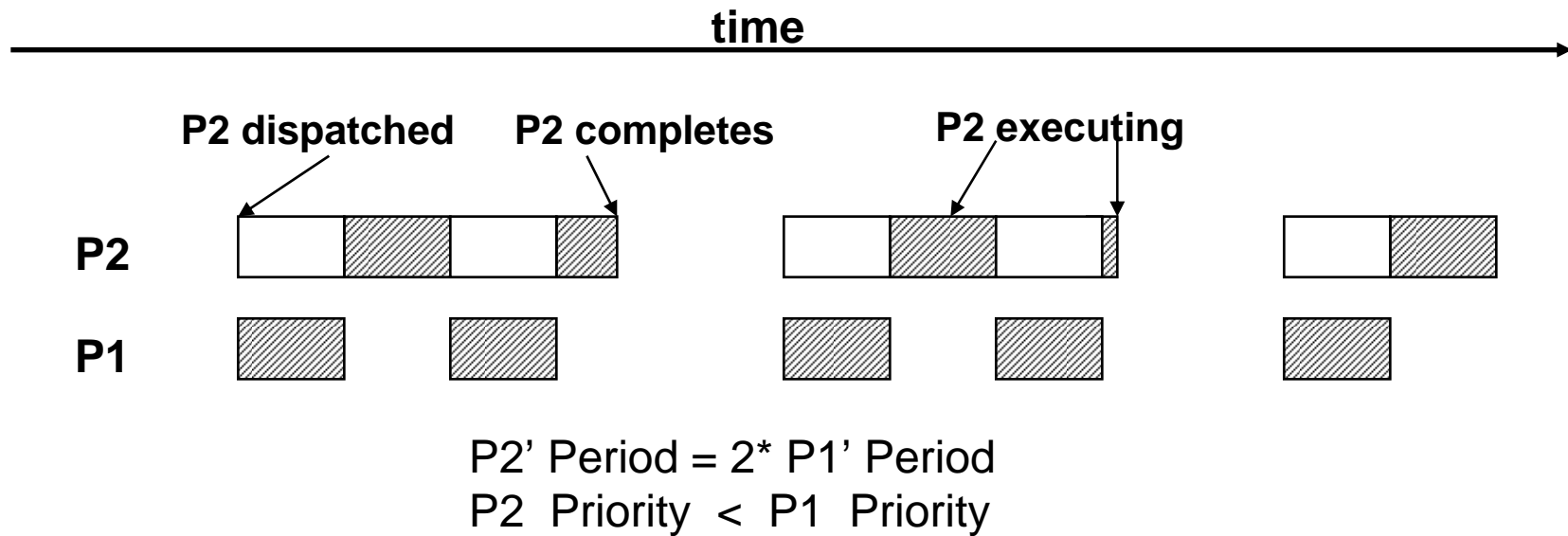
Assumed all processes and messages were periodic.

Used a previous product workload (periods, execution times) as a template.

Assumed end-to-end signal deadlines were 3X the specified process period (standard includes a flow declaration feature).

Honeywell

Processor Scheduling Model



Used MetaH Preemptive Fixed Priority (PFP) analysis tool.

Honeywell

Global Timing Models

Time-Triggered

periodic dispatching on CPs, IOPs is synchronized and aligned
release times and deadlines statically scheduled
data exchanges at interfaces have deterministic timing

Asynchronously Sampled Interfaces

periodic dispatching on CPs, IOPs is not synchronized, phase may drift
message arrivals not synchronized with receiving CP or IOP clocks
varying delay across asynchronous interfaces between CPs, IOPs, switches

Precedence-Constrained Sequencing

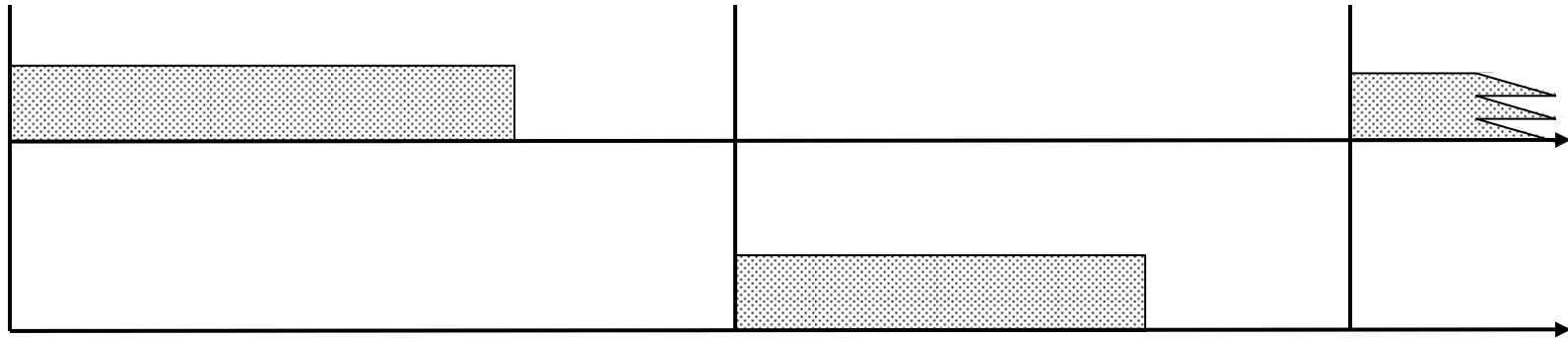
Periodic dispatching on CPs, IOPs is not synchronized, phase may drift
Release times occur when preceding task or message completion occurs
Queuing occurs for arrivals at every resource along a sequence

Many systems combine more than one.

We assume individual resource schedulers and analyzers are used, we are concerned with setting local parameters to guarantee global (end-to-end) latency bounds.

Honeywell

Time-Triggered Model



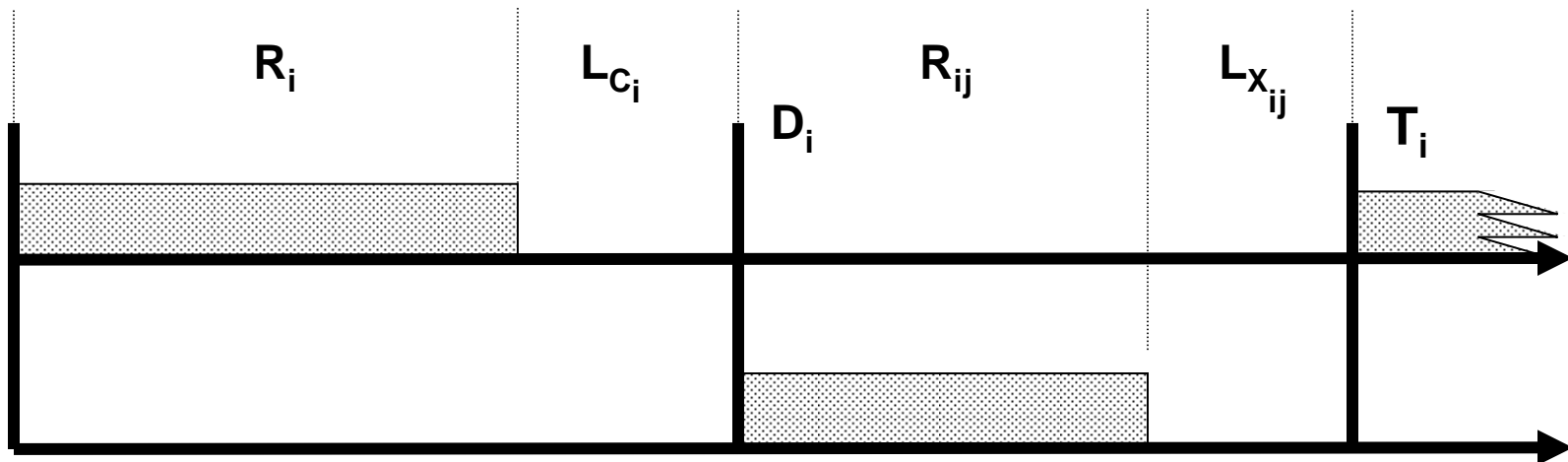
Globally synchronized clocks are maintained on every processor.

Release times and deadlines are statically scheduled within the hyperperiod.

- + deterministic and anomaly-free
- strictly periodic workloads

Honeywell

Time-Triggered Decomposition Scheduling



$$D_{i,0} = \frac{T_i}{2}$$

$$D_{i,t+1} = F(L_{ci}, L_{xij})$$

Decomposition scheduling iteratively computes a sequence of intermediate process deadline/ message release time points in a way that balances loading, using laxity results from individual resource scheduling and schedulability analysis algorithm.

Honeywell

Decomposition Scheduling Results

Messages from same process over same bus merged

(but no other multiplexing or multicasting)

One period deadline on each process/message pair

(pre-period deadlines on arbitrary length chains not implemented yet)

Workload (decomposition scheduler model):

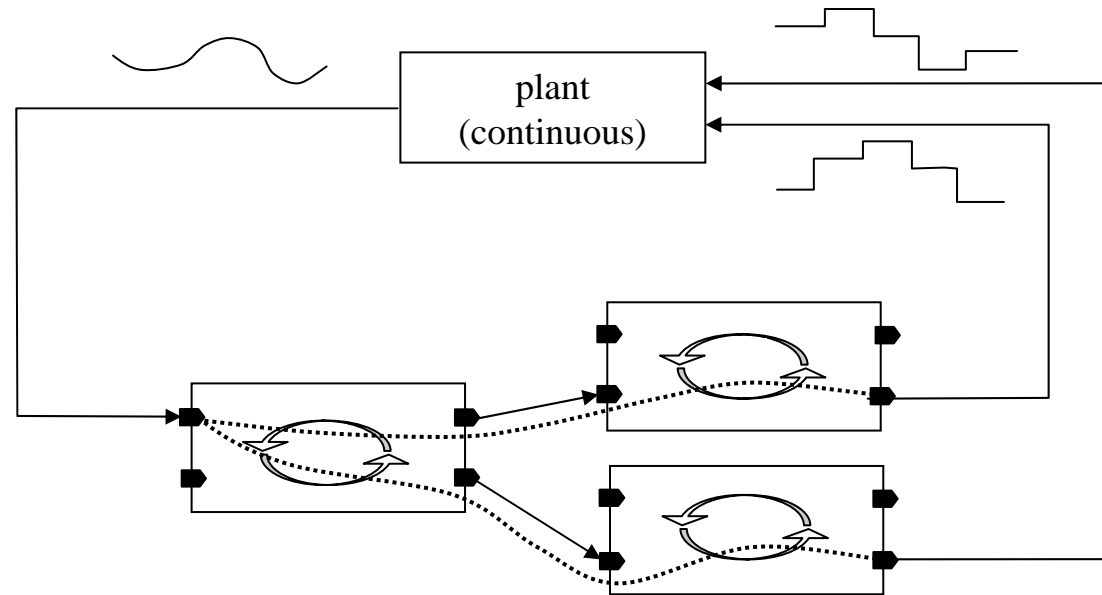
- 1322 messages on 8 busses
- 1402 processes on 26 processors

Model generated from AADL spec in about 45 seconds

Model scheduled and analyzed in about 10 seconds

Honeywell

Asynchronous Sampled Interface Timing Model



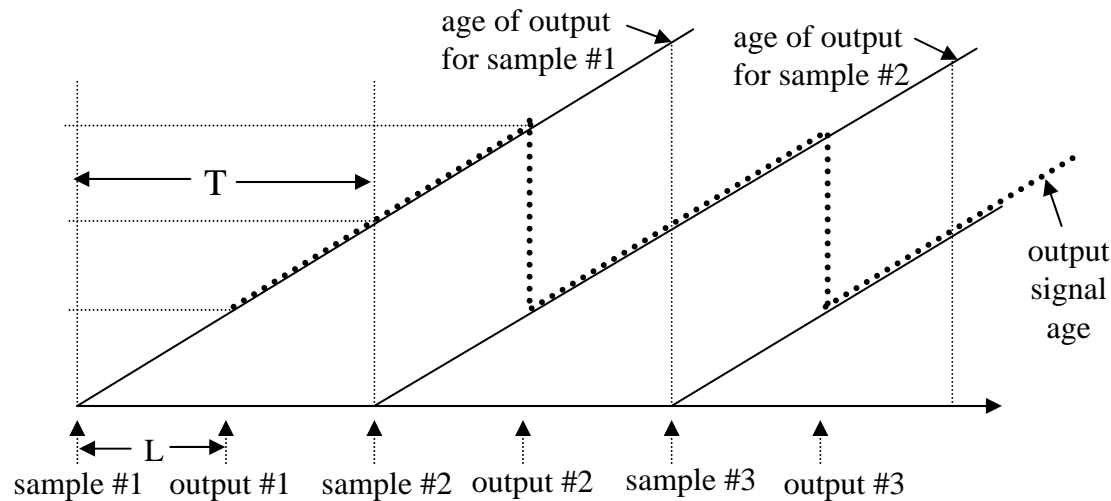
Each task asynchronously samples the outputs of the preceding tasks on a signal flow.

- + minimal synchronization, fan-in/fan-out easily implemented
- inherent inefficiencies
- periodic feed-back control only

Honeywell

Age Scheduling

Definition: The *Age* of an output signal is the time elapsed since the input on which it is based was sampled.



Theorem: The age of an output signal is bounded by $\sum_{i \in \Psi_\phi} (T_t + L_i)$

Developed uni-processor efficiency bounds and age scheduling algorithms.

Global problem expressible as a system of non-linear constraints,

$$\sum_{t \in \Psi_\rho} \frac{C_t}{A_t} \leq U_\rho^* \quad \sum_{t \in \Psi_\phi} A_t \leq A_\phi$$

Honeywell

Age Scheduling Results

Connections were merged (multiplexed) if

- They had the same route between the same processors
- The connected processes had the same periods
- 2644 merged to 610

Processes at same rates on same IOM (but not CP) were merged

- 1472 merged to 203

We modeled every CP, IOM, and bus as a resource

Workload (AMPL model):

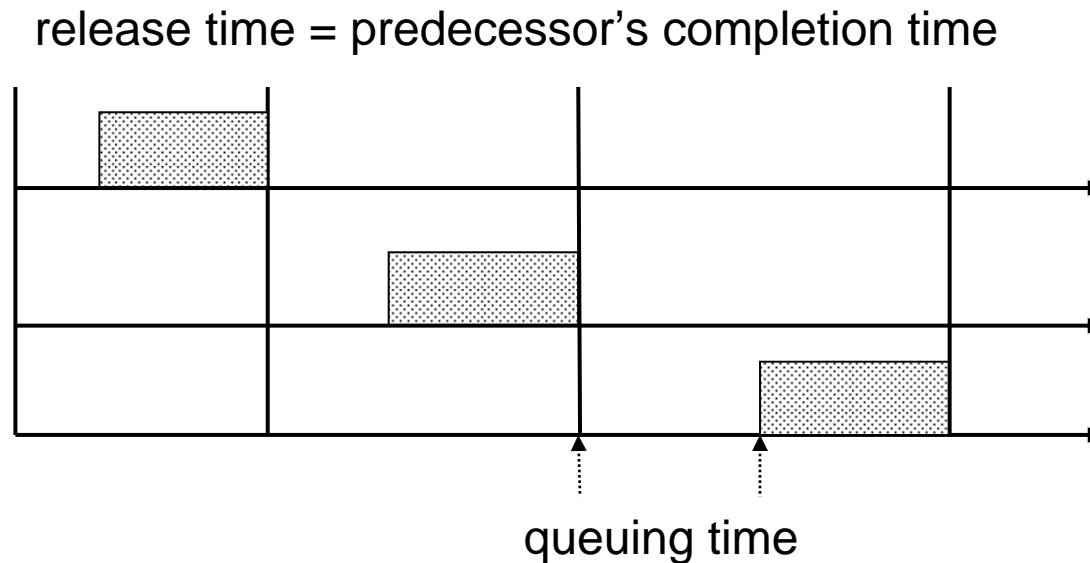
- 1425 variables (one for each process/processor and message/bus pair)
- 1872 constraints (one for each resource and one for each signal)

Model generated from AADL spec in about 45 seconds

Feasible solution found by CONOPT in about 45 seconds

Honeywell

Precedence-Constrained Sequencing



Next task or message is released when predecessor completes.

Every resource has a scheduling queue, adds latency and jitter

- + non-periodic workloads
- non-deterministic and anomalous
- fan-in can require non-trivial synchronization

Honeywell

Precedence-Constrained Sequencing Evaluation

Internal project to co-develop and assess preliminary analytic performance models and AFDX designs.

Performance is what you can certify, not what you can simulate.

Evaluated using a Primus EPIC® periodic bus workload

47 processor and IO modules

179 software applications

367 signal sources, 2444 destinations on bus A

Used postulated ARINC 664 (AFDX) End System and Switch designs, topologies, routing, scheduling.

Honeywell

AFDX Evaluation Results

Initial seemingly reasonable design choices lead to a very inefficient network, initial bottleneck payload utilization $< 1\%$

Details are important to achieve high certifiable performance with an efficient and simple design, e.g.

- Multi-cast multiplexing and routing
- Deterministic and accurately analyzable scheduling
- Simple minimal regulation and jitter management

Final model+design achieved bottleneck breakdown payload utilizations $> 80\%$.

Developed a holistic schedulability model (vs Network Calculus)

Multiplexed, routed, scheduled in about 30 seconds.

Analyzed in about 10 seconds.

Honeywell

Safety & Error Models

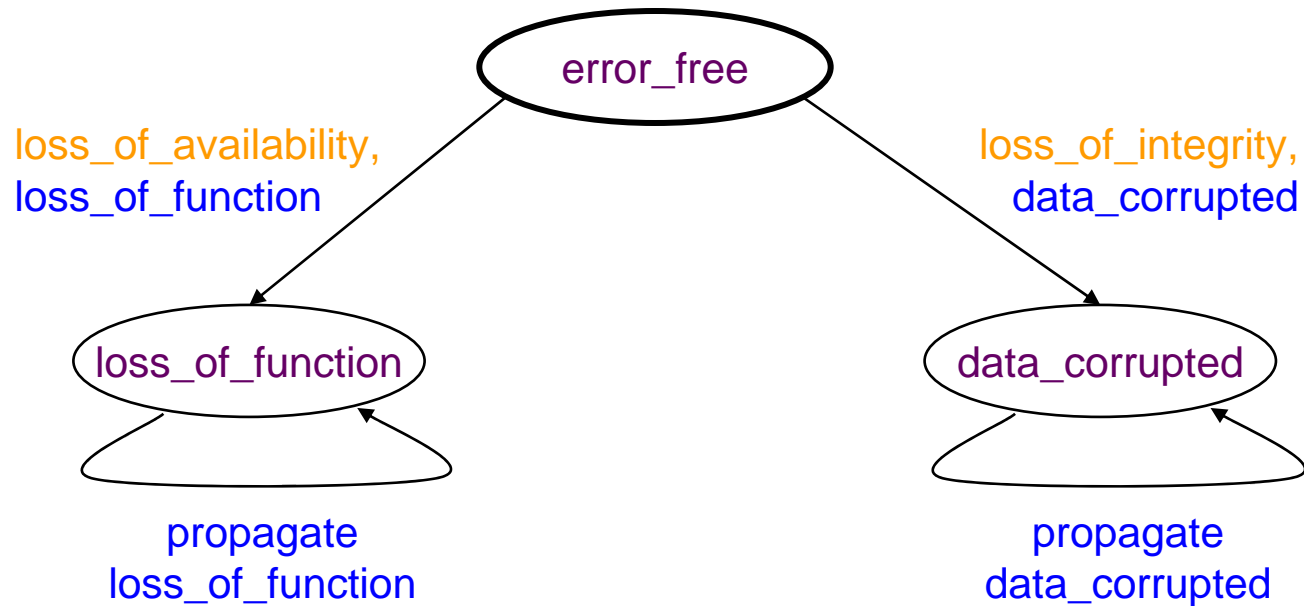
Markov Models

- Component error models with recovery & reconfiguration (cycles)
- Tractability requires generation of high-level and structured models (e.g. HI MetaH stochastic automata into UIUC Möbius).

Fault Trees

- Restricted to cycle-free component error models

AADL Component Error Models



An error model specifies one or more sets of:

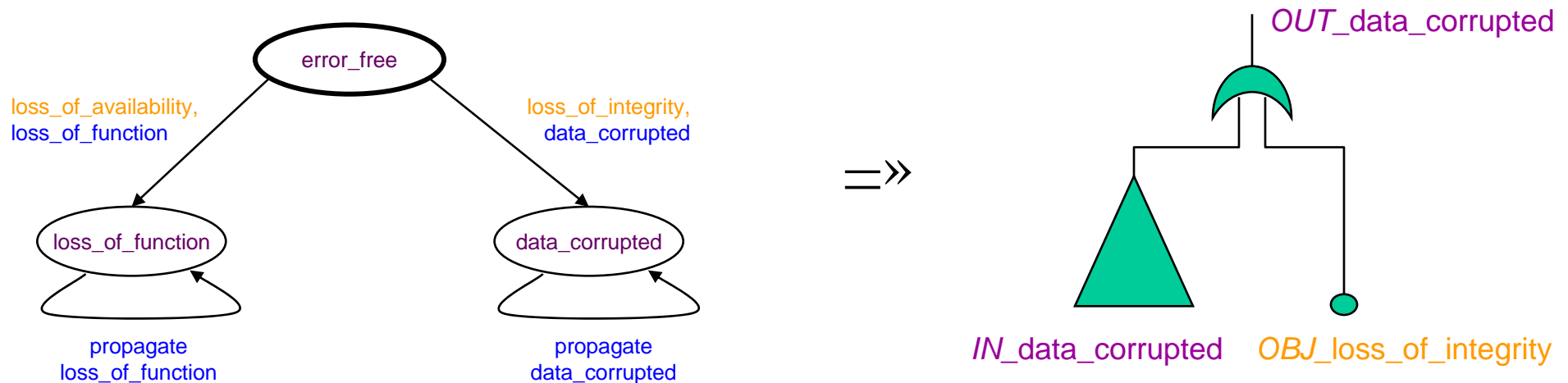
- faults that can occur in a component
- error states that a component can be in
- errors that can propagate into or out of a component
- transitions between error states due to faults and error propagations

Every component has an error model:

- specified for a category of component, and/or individually
- fault and error propagation rates can be specified individually

Honeywell

Translating Error Models to Fault Tree Templates



One fault tree template is constructed for each error state.

- One Priority-And gate for each path from the initial error state to the propagate-able error state. Inputs are the sequence of fault events and error propagations that label that path. Only needed if there are paths of length > 1 (none in this model).
- Root is an Or gate that combines all path fault trees for that propagate-able error state. Only needed if there is more than one path.
- Incoming error propagations are fault (sub)trees (as explained in following slides).
- The fault tree for error_free is the negation of the conjunction of the other fault trees.

Restricted to cycle-free error models for fault tree generation (cyclic models usually require dynamic analysis, e.g. Markovian).

Honeywell

AADL Error Propagations and Voting

Potential error propagations can be automatically determined from the component error models and the overall architecture.

- The system model can be automatically generated by composing the component models.
- Legality rules insure consistency between all component error models.

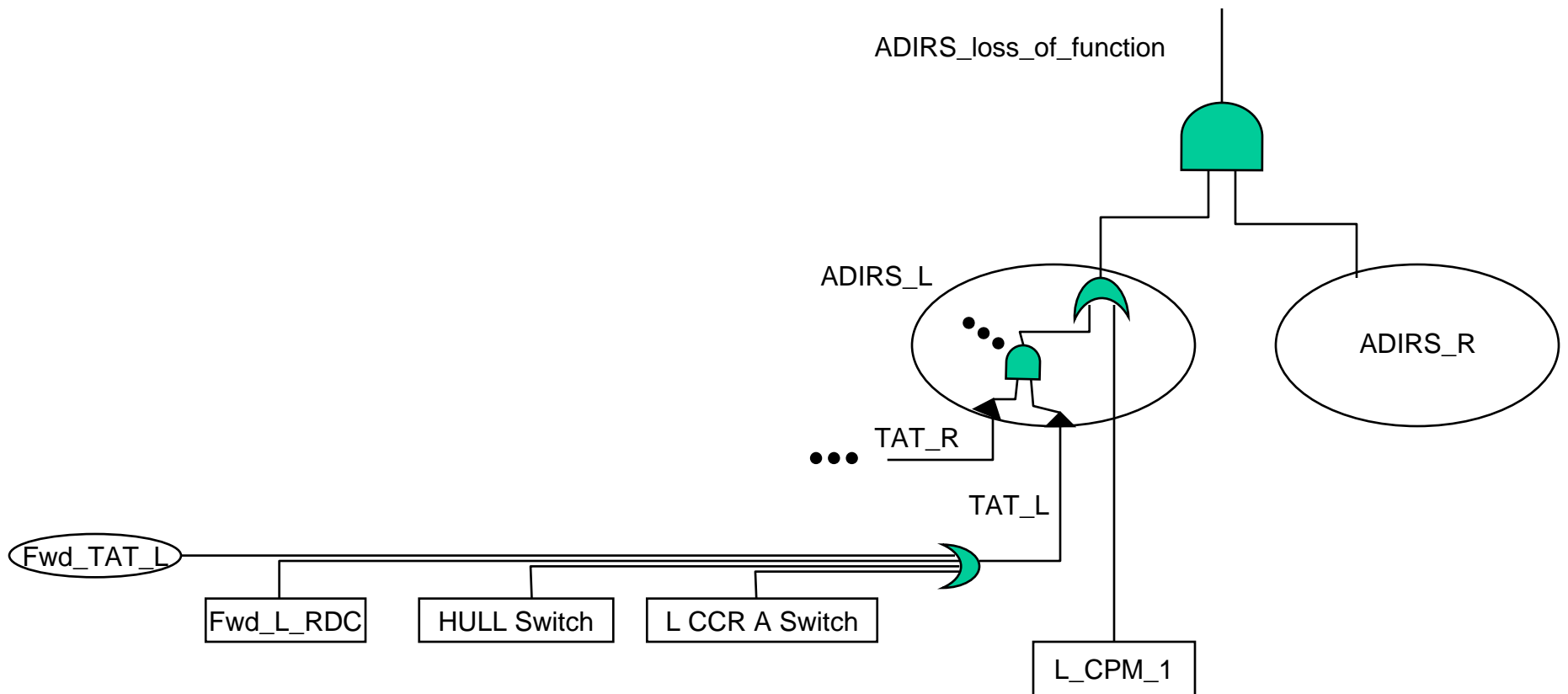
Properties `Vote_In` and `Vote_Out` may be used to vote or map errors propagated into or out of a component via message ports or shared objects.

Property `Vote_Subcomponents` may be used to specify internal redundancy.

Property `Model_Abstractly` may be used to substitute a simpler abstract model for the fully detailed automatically generated one for a subsystem.

Honeywell

What Generated Fault Trees Sort of Look Like



Honeywell

Fault Tree Results

Generated two fault trees per function:

- loss of availability
- loss of integrity

(Multi-function analysis is possible but was not specified or performed.)

Fault Tree sizes ranged from about 1000 to 15000 gates.

Full set generated from AADL spec in about 30 seconds.

Largest tree required about

- 4 hours to generate pretty graphical layout (vendor could by-pass)
- 10 minutes to analyze

Honeywell

Goals of 7E7 Project, Revisited

Assess whether the AADL can capture sufficient detail to automatically generate timing and safety models for a system of real-world complexity.

Yes, subject to the caveats/risks:

- Safety models are actually more complex and function-dependent
- End-to-end signal latency bounds were inferred (# flow specs?)
- Specification effort not assessed (we generated most of spec)

Assess whether automatic generation and analysis of timing and safety models is feasible and computationally tractable for a system of real-world size.

Yes, subject to the caveats/risks:

- Data did not include all functions (perhaps roughly half)
- We used some research methods and some ROTS tools
- More appropriately detailed models should be integrated

Provide useful preliminary feed-back to baseline architecture team.

No

- Schedule, customer and legacies forced earlier decisions
- Additional data needed for assured detailed decisions

Honeywell

A Few Comments

These were all periodic workloads.

Efficient level A partitioned periodic+incremental+aperiodic workloads are flying today.

Watch out for timing anomalies, race conditions, non-robust behaviors.

Analytic modeling lays a good foundation for avoiding such defects.

You can't avoid integrating a development environment suited to your product line architecture family.

Can't avoid working with tool suppliers.

Can't avoid filling gaps with custom scripts & tooling.

Appropriate models must be selected & integrated.

Architecture specification is one among many essential work products, but it plays a central role.

A "hub" or integrating specification.

Traceability between requirements, functions, and architecture is important.

How are you going to specify your architecture?

Everybody uses MathWorks stuff.

What about the other 60-95% of the code?

How well do you automate system integration and verification?

There is still a lot of inefficient resource usage in current systems due to poor allocation of system performance and dependability requirements to architectural elements.

Honeywell