

**Methods and Tools
For
Embedded Distributed System
Scheduling and Schedulability Analysis**

Steve Vestal
Honeywell Labs
Steve.Vestal@Honeywell.com

18 October 2005

Honeywell

- **Background**
- Binding and Routing**
- Scheduling and Analysis**
- AADL Tools**

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.

Binding then Scheduling

In principle binding (aka allocation, assignment, routing) and scheduling are inter-dependent, e.g.

Bindings must be schedulable

In practice a phased approach is often used, with earlier phases using simple approximate models of what later phases are capable of accomplishing, or no models at all, e.g.

Bind processes to processors without exceeding a given breakdown utilization.

Bind processes to processors first, then route connections through the network.

This presentation is phased, but keep this in mind!

The Multi-Resource Problem is Hard

Multi-resource binding and scheduling is theoretically and practically very difficult, e.g.

Cyclic job shop scheduling is NP hard, even when a binding is specified.

As far as I know, this problem has no approximation algorithms with tight performance guarantees and no generally accepted heuristics that are fast and efficient for all varieties of problem.

Many intuitively good approaches (e.g. extensions of single resource approaches that have optimality or performance guarantees) are not good for the multi-resource problem, e.g.

RMA and EDF for symmetric multi-processing of periodic threads have arbitrarily small breakdown utilization.

Many multi-resource algorithms are subject to anomalies, e.g.

A deadline may be missed when a thread completes earlier than its specified WCET.

A deadline may be missed when a sporadic arrives less frequently than its specified minimum inter-arrival time.

Offline, Online, Dynamic

For high assurance systems, binding and scheduling are almost entirely offline development activities.

Simplifies the product that needs to be certified

Simplifies offline verification, validation, certification

Incremental development processes and tools are desirable.

Minimize change impacts during upgrade

Minimize differences between different product configurations

There is a spectrum of choices of what to do offline versus what to do online.

There is a spectrum of assurance, cost and schedule, and product functionality and efficiency trade-offs.

Background



Binding and Routing

Scheduling and Analysis

AADL Tools

The Classical Binding Problem

Given

- A hardware architecture
- A software architecture
- Scheduling assumptions and timing constraints

Produce

- An assignment of software components to hardware components such that all timing constraints are satisfied.

Something Closer to the Real Problem

Given

- Physical sensor and actuator locations
- Functions (algorithms, control/data flows, I/Os)
- Constraints on

Size	Power	Cabling	Availability
Weight	Placement	Integrity	Timing

Produce

- A co-designed software + hardware architecture that satisfies the constraints.

Capabilities in Various MetaH/AADL Tools

Given a software and hardware architecture,

do

Multi-mode assignment of processes to processors

then

Multicast multihop multimode multiplexing & routing of connections

Timing and scheduling constraints are approximated by assuming a breakdown utilization for each resource.

Additional constraints are declared using properties. It is still uncertain exactly what is a reasonably convenient and complete set of properties.

Pattern-Based Architecting

Integrity, availability, reliability, etc. are often addressed by choosing from a variety of well-studied redundancy patterns, e.g.

- {ED, EDAC} information coding**
- Self-checking circuits**
- Fail-stop self-checking pair**
- {Triple, Quad} modular redundant error masking**
- Primary with {hot, warm, cold} spare**

These are often combined to form more complex patterns, e.g.

- Self-checking pair primary and spare**
- TMR primary with single spare**
- QMR processing with dual EDAC networking**

There are choices about what to do in software and what to do in hardware.

We have been experimenting with pattern-based architecting, supported by suitably tailored traditional binding and routing capabilities (e.g. define a good set of supporting constraint properties).

Background

Binding and Routing

 **Scheduling and Analysis**

AADL Tools

Compositional Global Scheduling and Analysis **Honeywell**

We assume single-resource scheduling technology is available for each individual resource in the system.

The global scheduling problem is to assign local scheduling parameters (e.g. release times, deadlines) in a way that achieves desired global scheduling properties.

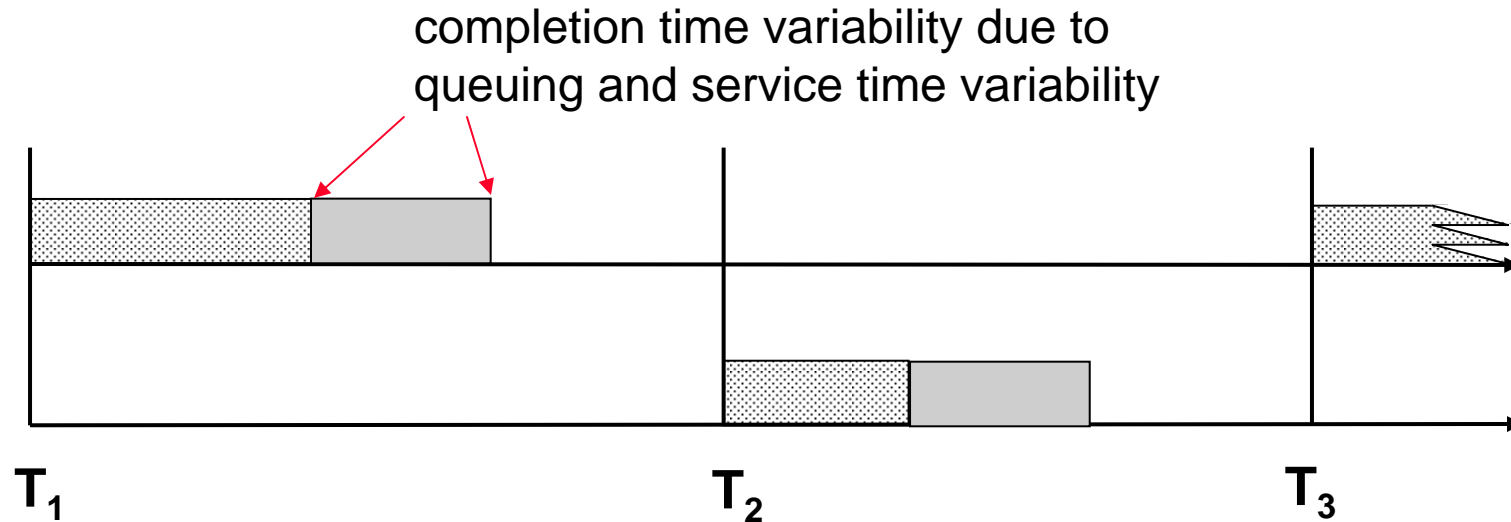
The global schedulability analysis problem is to combine local schedulability analysis results to produce end-to-end timing analysis.

This depends on the temporal interfaces between components and subsystems:

- Globally Time-Triggered**
- Precedence-Constrained Sequencing**
- Asynchronously Sampled Internal Interfaces**

Many systems combine more than one.

Globally Time-Triggered



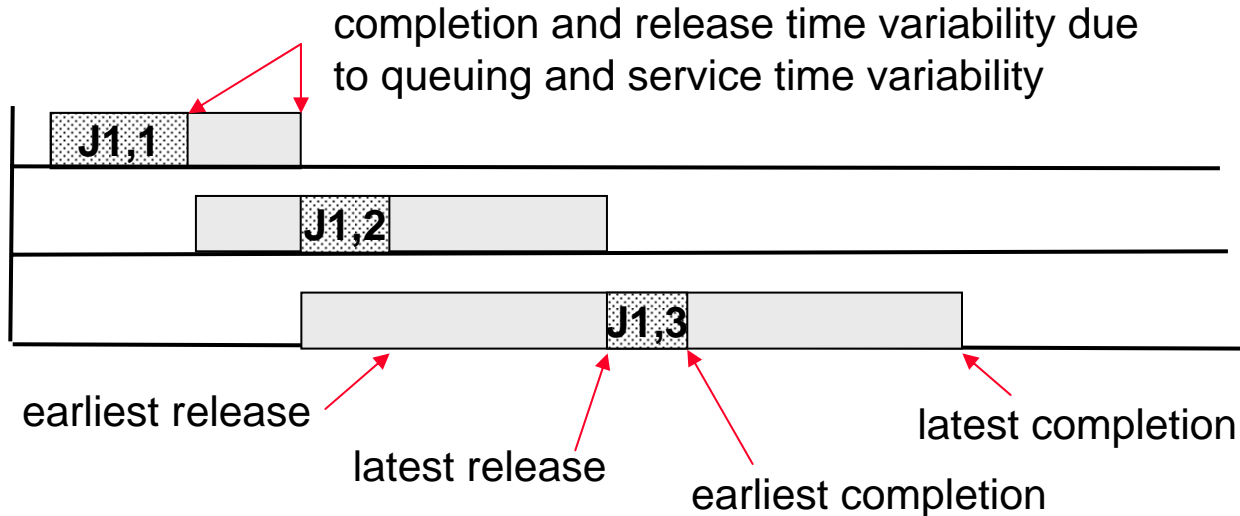
Globally synchronized clocks are maintained on every processor.

Release times and deadlines are statically scheduled within the hyperperiod.

A hyperperiod schedule of these events is loaded as a table into every resource and used at run-time to trigger dispatches, transmits, etc.

Precedence-Constrained Sequencing

A subjob $J_{i,k+1;j}$ is released at the completion of subjob $J_{i,k;j}$

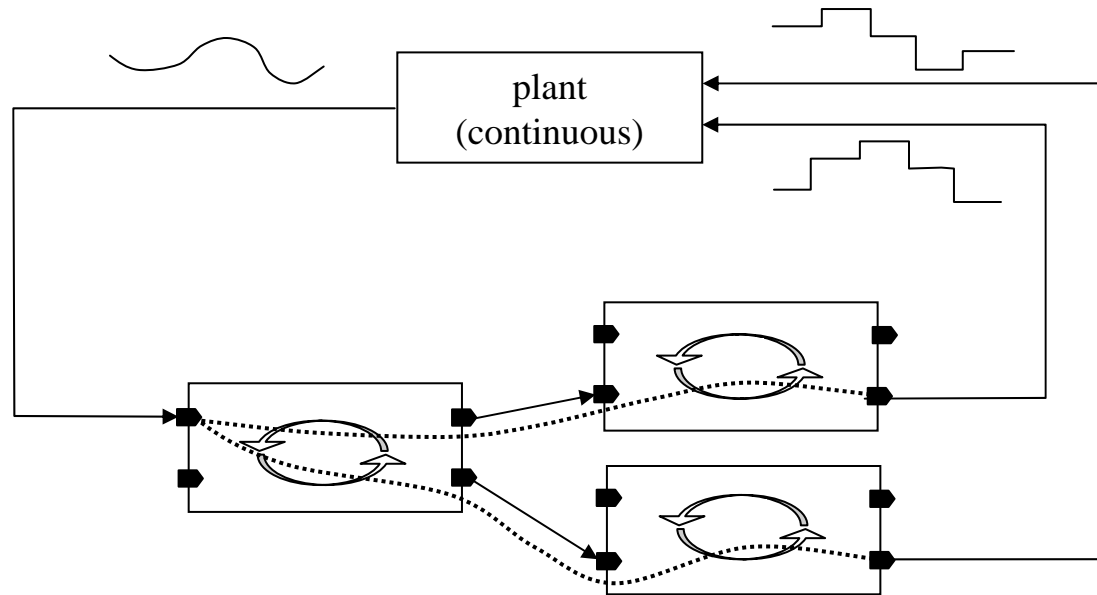


Run-time release signals are sent from the resource hosting the completing subjob to the resource hosting the successor subjob.

Release and completion time jitter grow at each step.

Traffic regulators are sometimes used to obtain tighter bounds and minimize anomalous scheduling effects.

Asynchronously Sampled Internal Interfaces



Each resource (subsystem) performs independent periodic sampling.

Periodic events on different resources are unsynchronized. They have unknown phase offsets and are subject to relative drift.

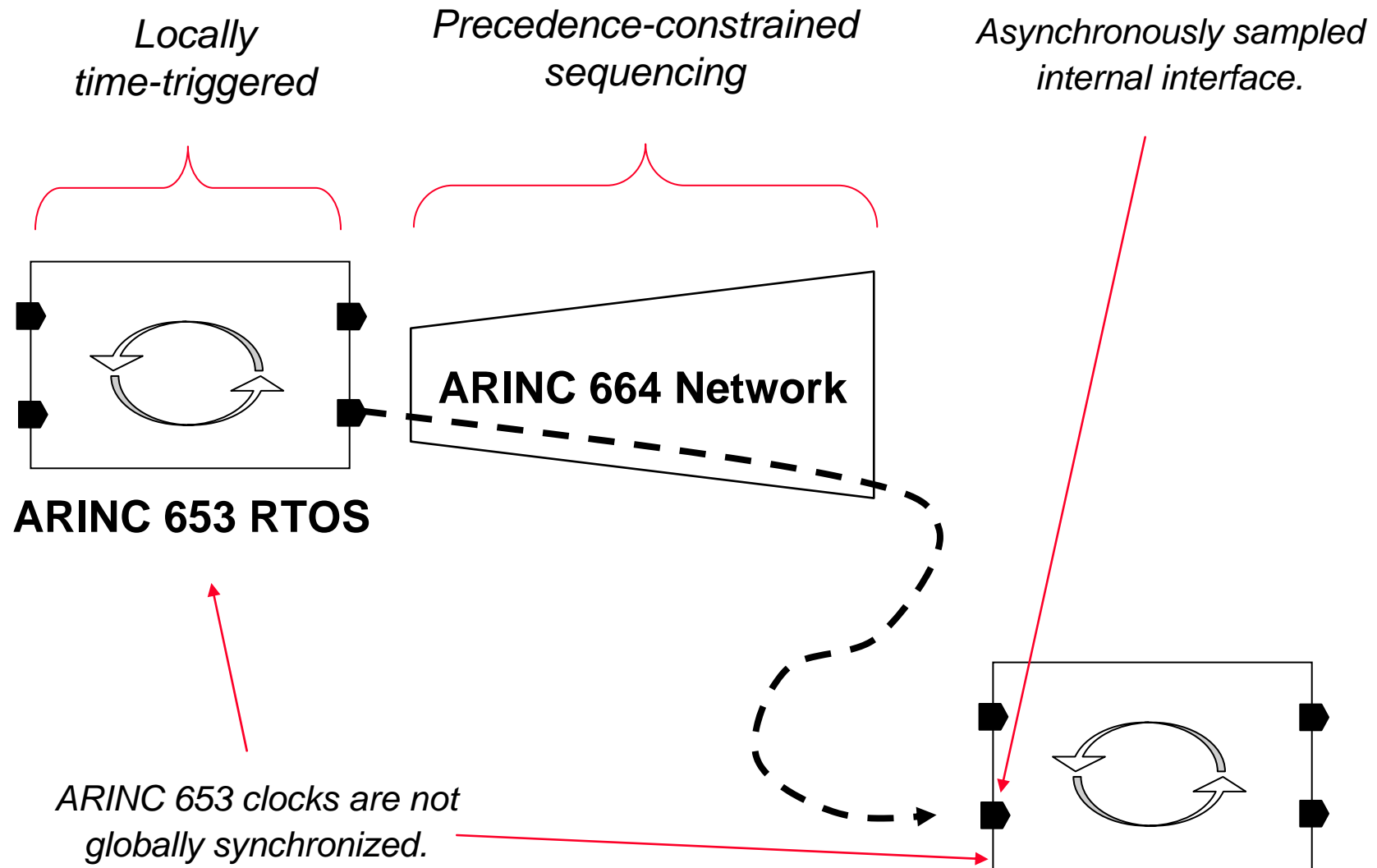
Each task asynchronously samples the outputs of the preceding tasks at dispatch and sets the values of its output buffers at completion.

Some Pros and Cons

Criteria	Time Triggered	Sequenced	Asynchronous
deterministic, no run-time anomalies	Green	Red	Yellow
ease of fan-in/fan-out	Yellow	Red	Green
aperiodics, dynamic adaptability	Red	Green	Red
loss-less	Green	Green	Red

How tractable and efficient are scheduling and analysis?

A Practical Example



Evaluations of various methods and tools have been carried out over the past few years using one or more of the following workloads.

Air transport aircraft IMA (simplified production workload)

Globally time-triggered
6 processors, 1 multi-drop bus
105 threads, 51 message sources

Military helicopter MMS (first release, partial)

Globally time-triggered
14 dual processors, 14 bus bridges, 2 multi-drop buses
306 threads, 979 [source, destination] connections

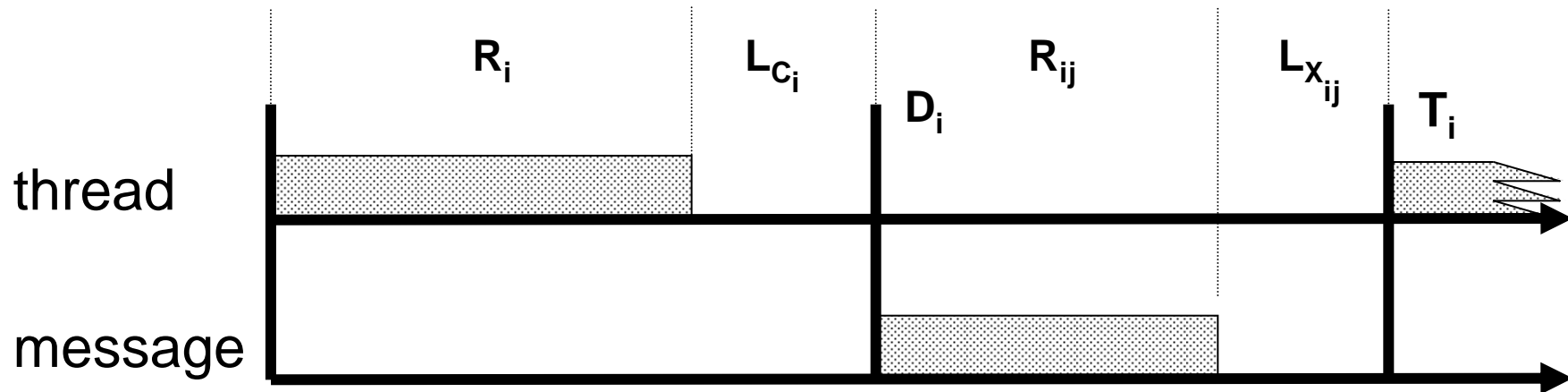
Air transport aircraft IMA (preliminary, partial)

Globally asynchronous processors, precedence-constrained switched network
26 processors, 12 switches
1402 threads, 2644 [source, destination] connections

Regional aircraft IMA (production workload)

Globally time-triggered
49 processors, 2 multi-drop busses
244 processes (TBD threads), 3179 [source, destination] connections

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.

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 time-triggered busses
- 1402 processes on 26 processors

Model generated from MetaH spec in about 45 seconds

Model scheduled and analyzed in about 10 seconds

There are several heuristics for priority assignment, e.g.

- **Ultimate deadline (final deadline)**
- **Effective deadline (final deadline minus remaining service time)**
- **Proportional deadline (final deadline apportioned by service times)**
- **Normalized proportional deadline (normalized by resource utilizations)**

Approaches to worst-case latency analysis

- **Extended busy interval (aka time demand)**
- **Network calculus**

Precedence Constrained Sequencing Results Honeywell

Extended busy interval network analysis tool developed
(network calculus tool in progress)

Only connections from the same partition were multiplexed.

Evaluated using

- regional aircraft and synthesized clock synchronization workloads
- cascaded star and fully interconnected switch synthesized networks

Routed, multiplexed, scheduled, analyzed in about 40 seconds.

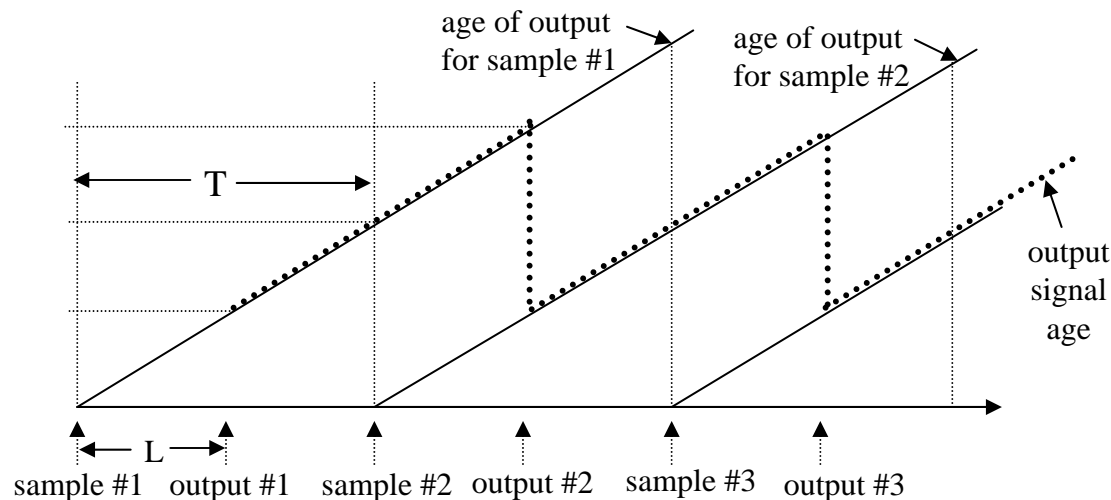
Binary search for breakdown workload for 5 switch 100mbs network

- Search took 14 iterations, ~ 30 minutes

- Breakdown workload ~1600 processes, ~26000 [source, destination] connections

Age Scheduling Across Asynchronous Interfaces

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 $\sum_{i \in \Psi_\phi} (T_i + L_i)$

Developed uni-processor efficiency bounds and age scheduling algorithms.

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

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

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 IO modules were merged

- 1472 merged to 203

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 MetaH spec in about 45 seconds

Feasible solution found by CONOPT in about 45 seconds

Background

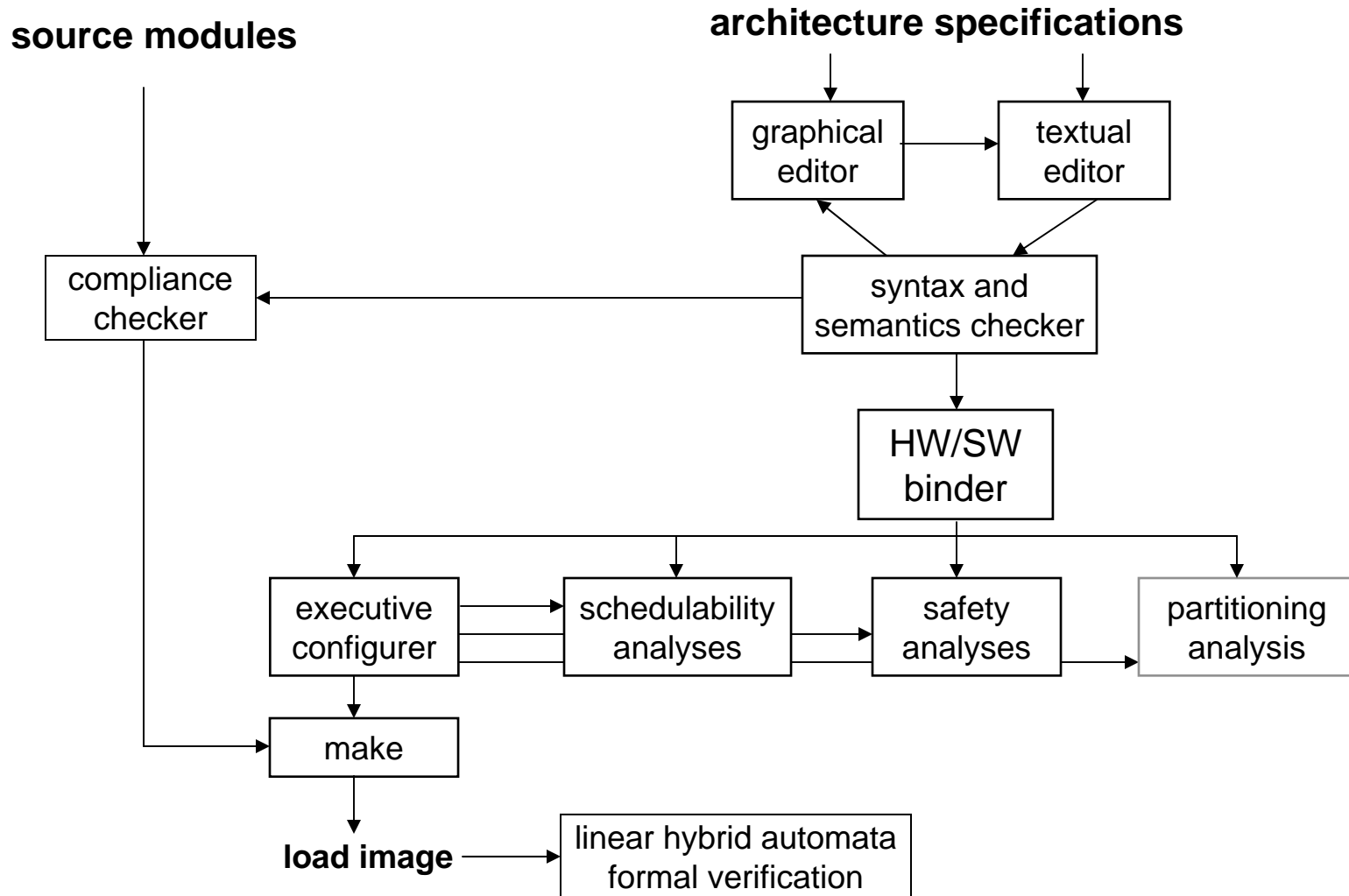
Binding and Routing

Scheduling and Analysis

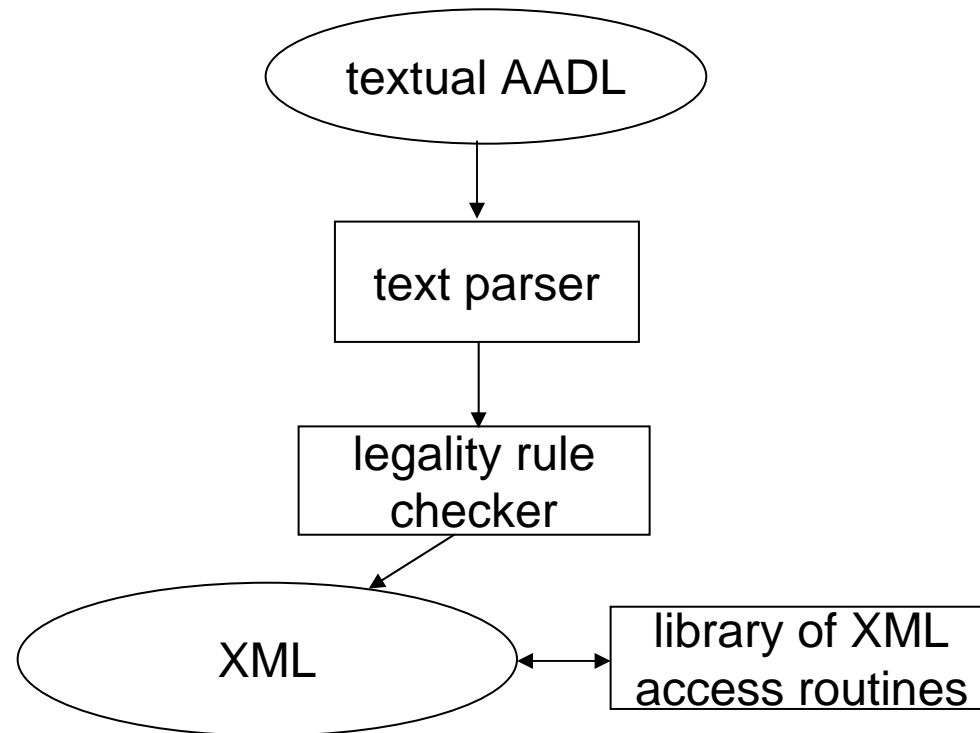


AADL Tools

In the Beginning There was MetaH



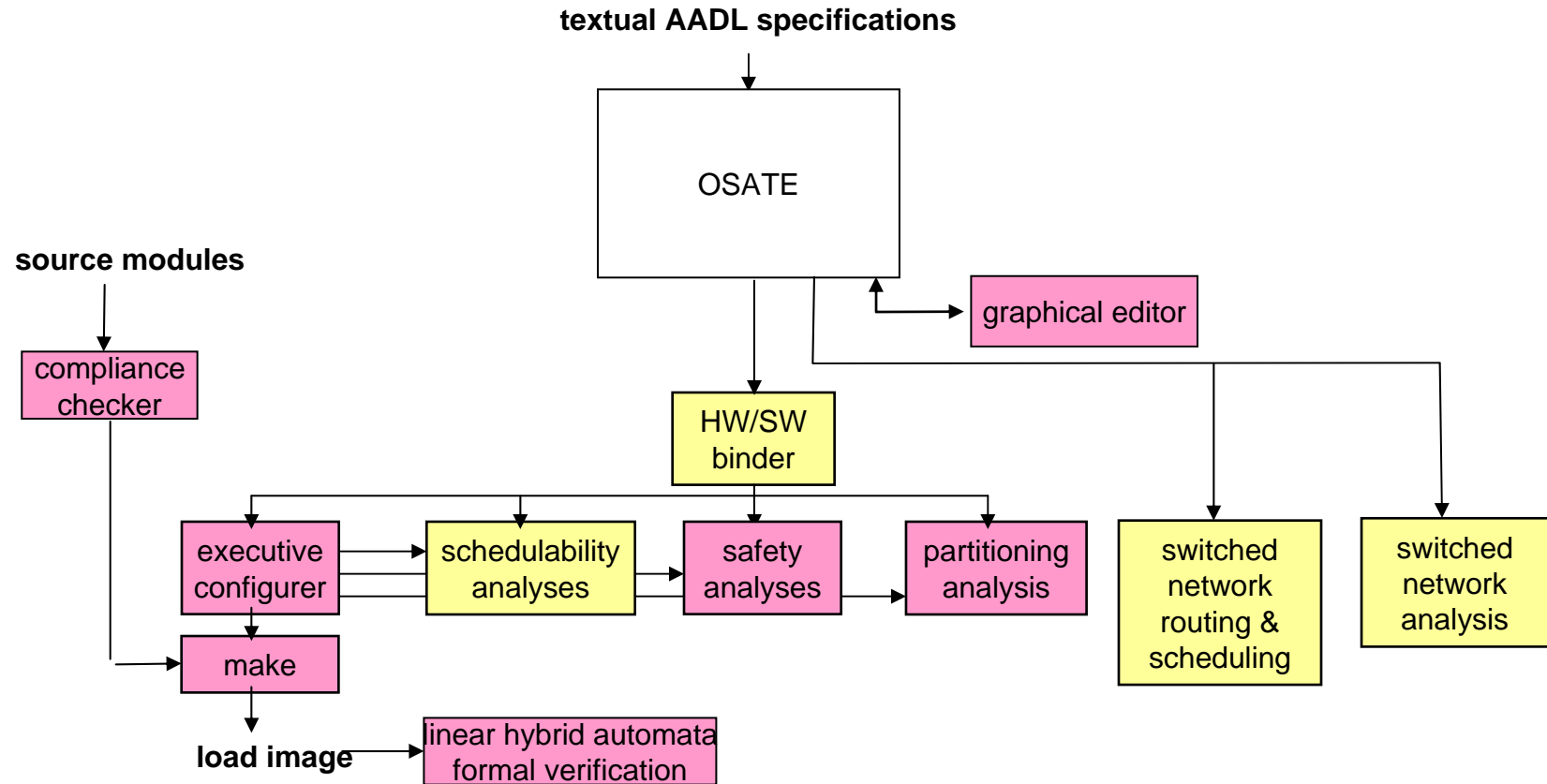
Open Source AADL Tool Environment (OSATE)



OSATE is an open source Eclipse-based IDE for AADL developed by the Software Engineering Institute (www.aadl.info).

TopCased is a European consortium working on system development environments, the two projects are coordinating (www.topcased.org).

AADL-Compliant Toolset



Yellow tools were delivered to Army AMCOM end of September 2005.

Pink tool integration funding and schedule are TBD.

Next Steps

Integrated binding and routing tool

Multi-modal

Multi-cast multiplexing and routing

Properties that support pattern-based architecting

Back-annotation into OSATE XML for use by others

Integrated global scheduling

Automated decomposition of thread, connection and flow timing constraints to individual resource scheduling problems

Integrated global schedulability analysis

User-specifiable global timing interfaces (time-triggered, sequenced, asynchronously sampled)

Automated composition of individual resource analysis results to obtain end-to-end thread, connection and flow analyses

