CSSE 490 Model-Based Software Engineering: MBSysE and Architecture Description Languages

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

You may feel like this about now…

Learning Outcomes: MBE Discipline

Relate Model-Based Engineering as an engineering discipline.

- Discuss Case Study Paper
- Examine why Model-Based Systems Engineering needed
- Explore ADLs
- Discuss AADL (if time)

Case Study/Homework:

“Model-Driven Systems Engineering” by Balmelli et. al.

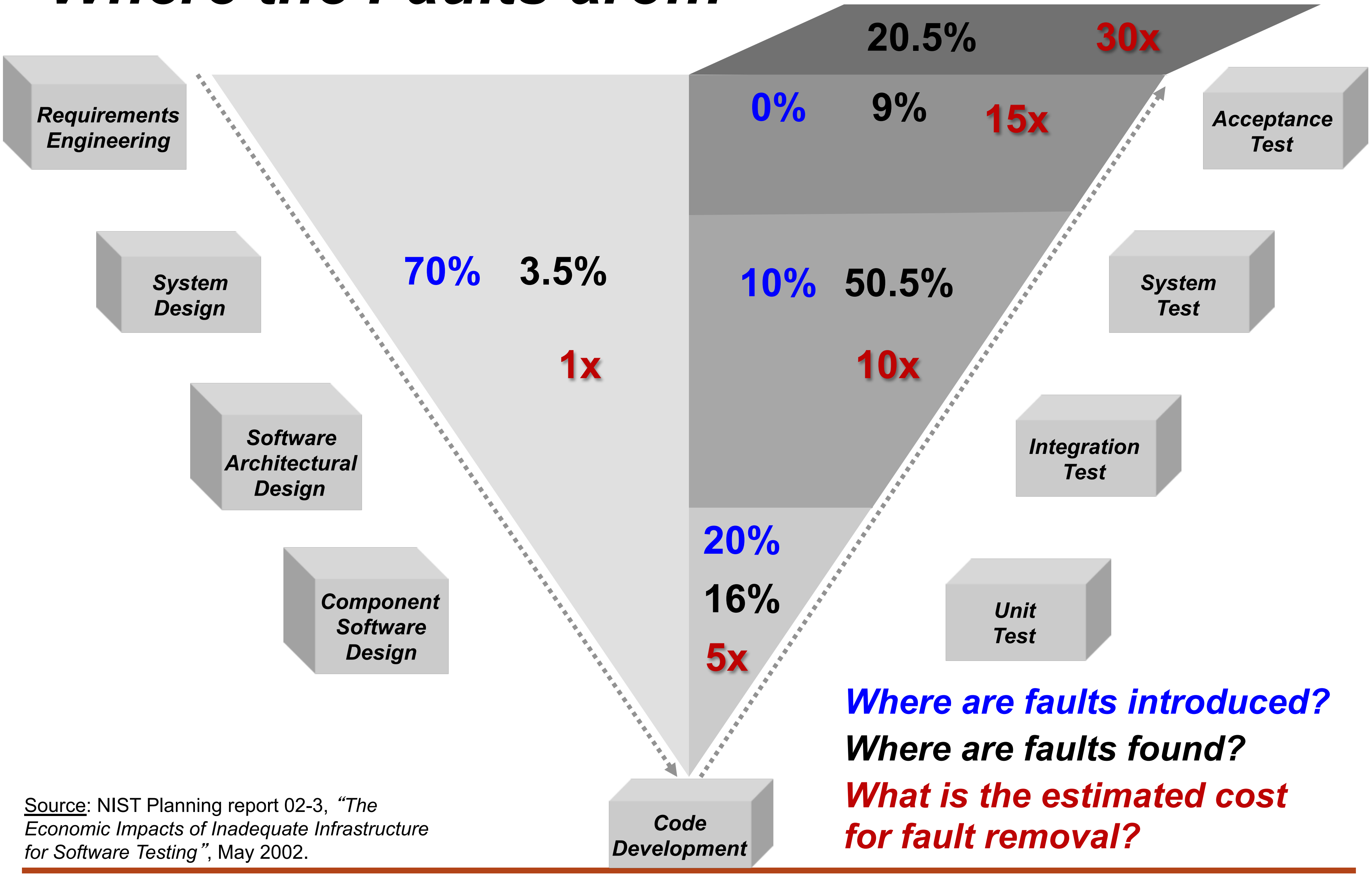
- What the authors mean by requirement-driven systems development methods?
- How does RUP Architecture Framework serve as a basis for MDSysD?
- What is the authors’ view of SysML and how it supports MDSysD?
- What do the authors say about the use of transformation methods? Are they effective? What is their metamodel like?

Late Discovery of System Problems

- System integration problems
  - System instability & failures
  - Implicit & mismatched assumptions
  - Complexity of component interaction

- Current practice
  - Build components first
  - Then integrate & test
- Way forward
  - Analyze system models early & often
  - Evolve components & integrated system

Where the Faults are…



Requirements Engineering

System Design

Software Architectural Design

Component Software Design

Code Development

Unit Test

Integration Test

System Test

Acceptance Test

70% 3.5% 1x

20% 16% 5x

10% 50.5% 10x

0% 9% 15x

20.5% 30x

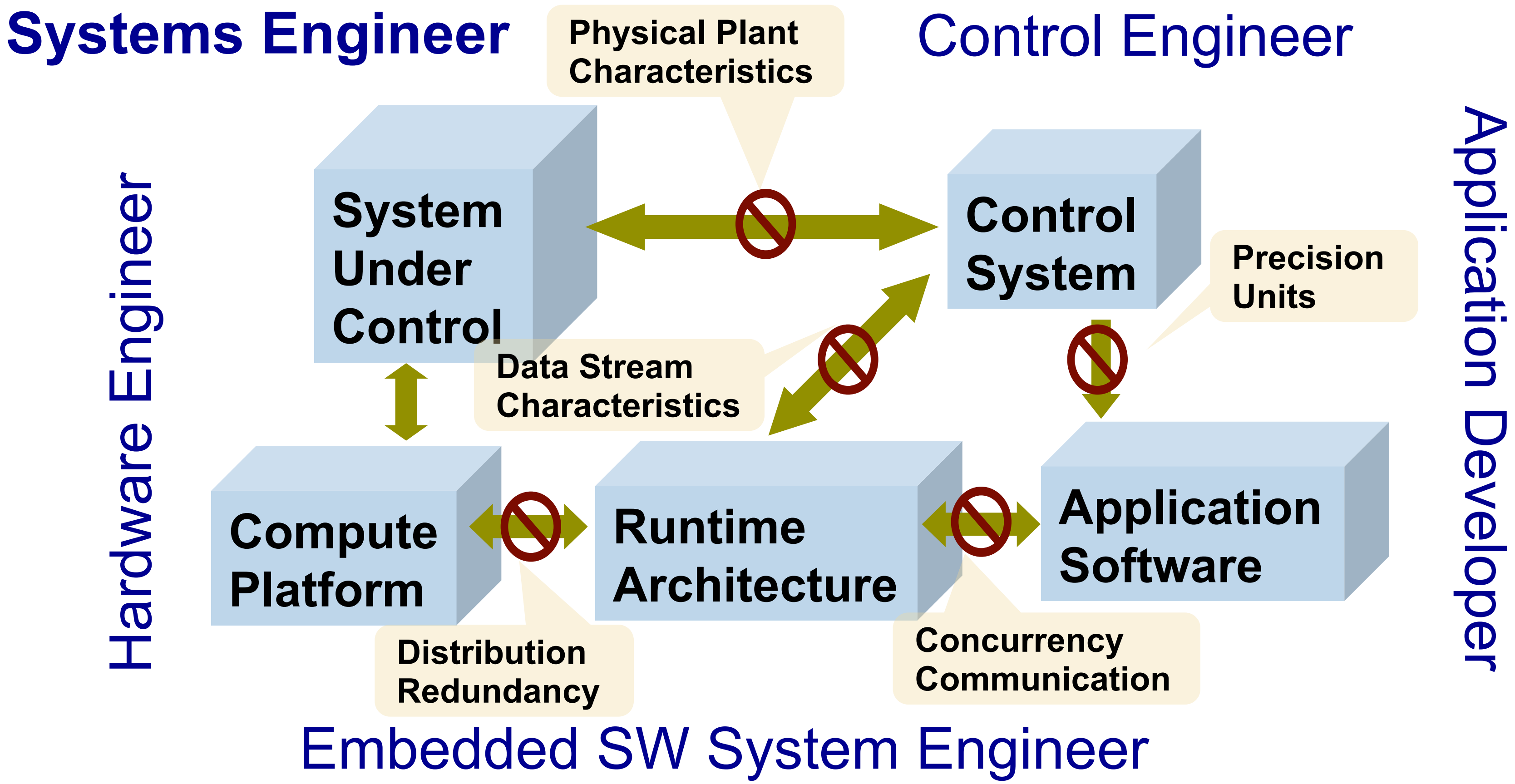
Where are faults introduced?

Where are faults found?

What is the estimated cost for fault removal?

Source: NIST Planning report 02-3, *"The Economic Impacts of Inadequate Infrastructure for Software Testing"*, May 2002.

Mismatched Assumptions



Why do system level failures still occur despite fault tolerance techniques being deployed in systems?

Architecture – A Definition

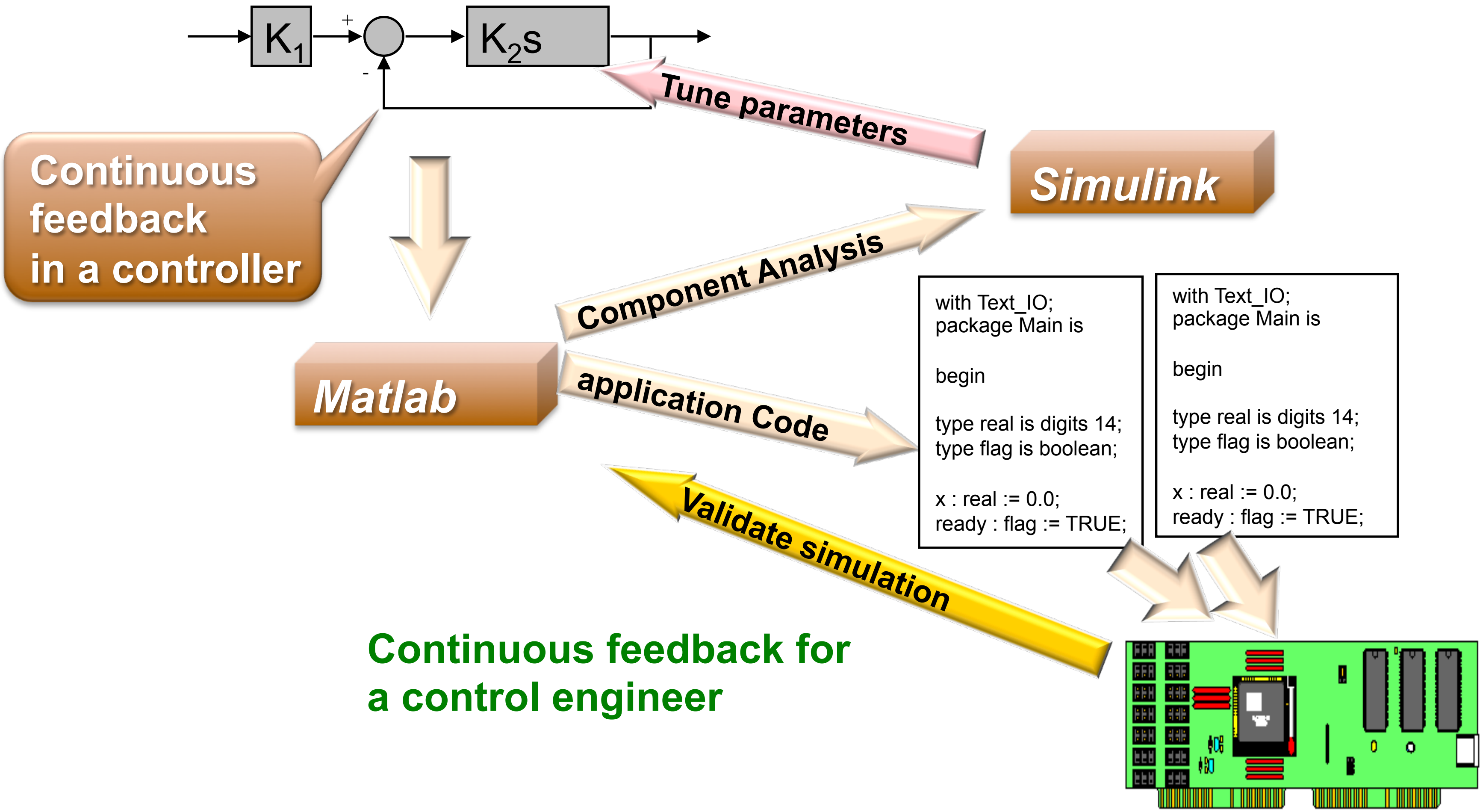
“The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.”

<u>Software Architecture in Practice,</u>
Bass, Clements, and Kazman

How can Model-Base Engineering techniques start to address the situations that hamper systems development?

- Think for 15 seconds…
- Let’s talk…

A Control Engineer Perspective



Continuous feedback in a controller

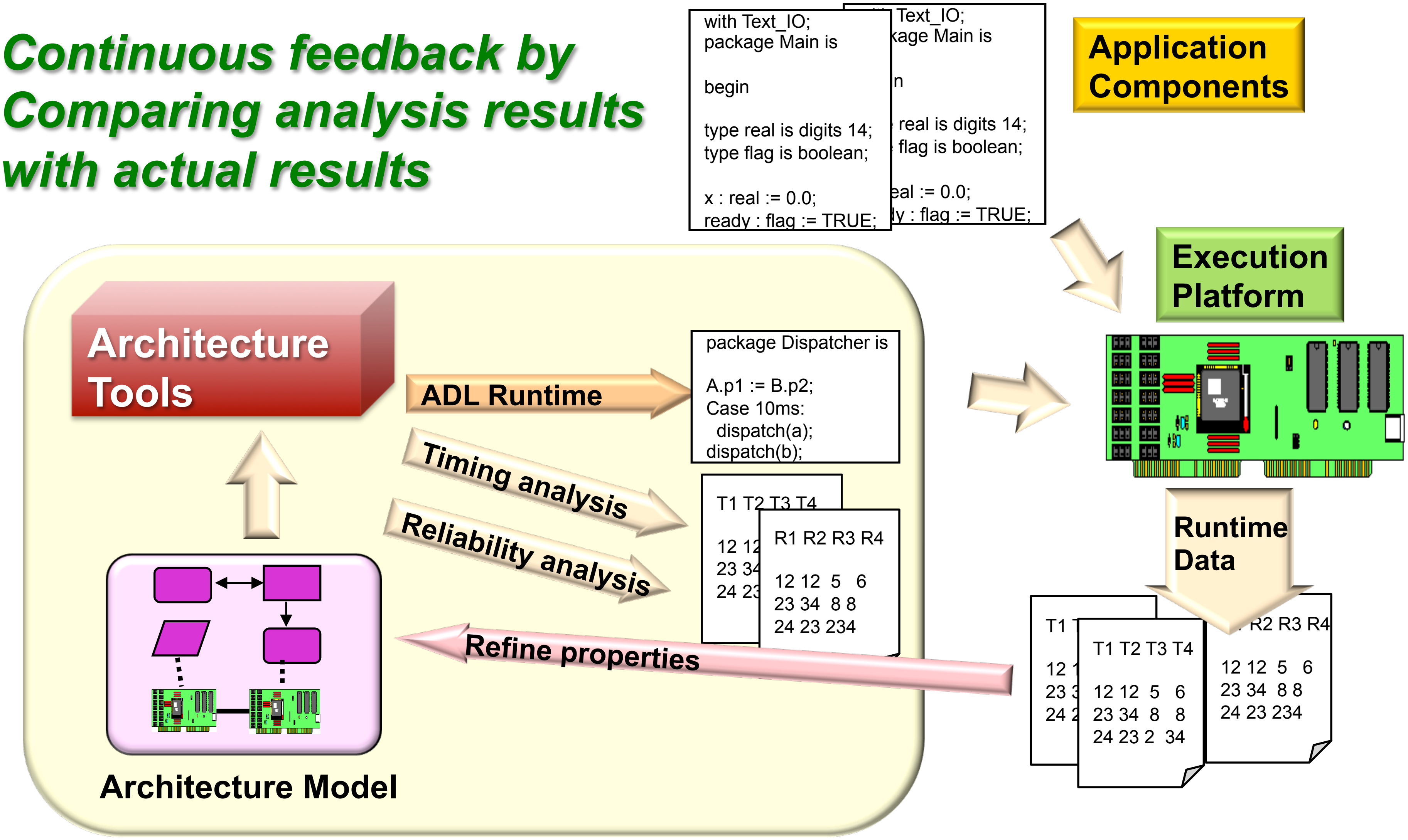
```
with Text_IO;
package Main is

begin

type real is digits 14;
type flag is boolean;

x : real := 0.0;
ready : flag := TRUE;
```

```
with Text_IO;
package Main is

begin

type real is digits 14;
type flag is boolean;

x : real := 0.0;
ready : flag := TRUE;
```

Continuous feedback for a control engineer

Software Engineer Perspective

Continuous feedback by Comparing analysis results with actual results



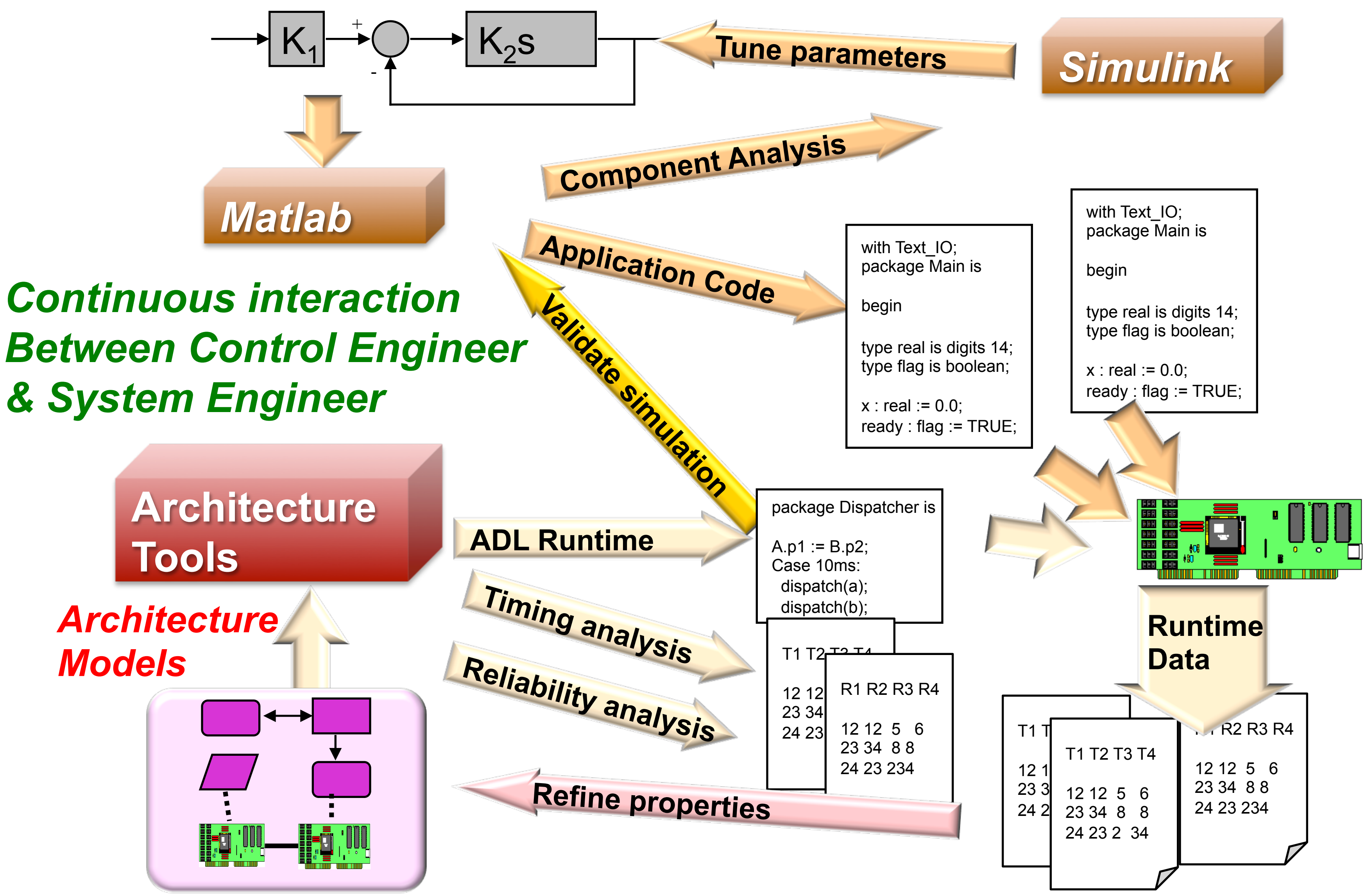
```
with Text_IO;
package Main is

begin

type real is digits 14;
type flag is boolean;

x : real := 0.0;
ready : flag := TRUE;
```

```
package Dispatcher is

A.p1 := B.p2;
Case 10ms:
  dispatch(a);
dispatch(b);
```

```
R1 R2 R3 R4

12 12  5   6
23 34  8 8
24 23 234
```

```
T1 T2 T3 T4

12 12  5   6
23 34  8   8
24 23 2  34
```

A Combined Perspective



Tune parameters

Simulink

Matlab

Component Analysis

Application Code

Validate simulation

Continuous interaction Between Control Engineer & System Engineer

```
with Text_IO;
package Main is

begin

type real is digits 14;
type flag is boolean;

x : real := 0.0;
ready : flag := TRUE;
```

```
with Text_IO;
package Main is

begin

type real is digits 14;
type flag is boolean;

x : real := 0.0;
ready : flag := TRUE;
```

Architecture Tools

ADL Runtime

```
package Dispatcher is

A.p1 := B.p2;
Case 10ms:
  dispatch(a);
  dispatch(b);
```

Timing analysis

Reliability analysis

```
T1 T2 T3 T4

12 12
23 34
24 23
```

```
R1 R2 R3 R4

12 12  5   6
23 34  8 8
24 23 234
```

Architecture Models

Refine properties

Runtime Data

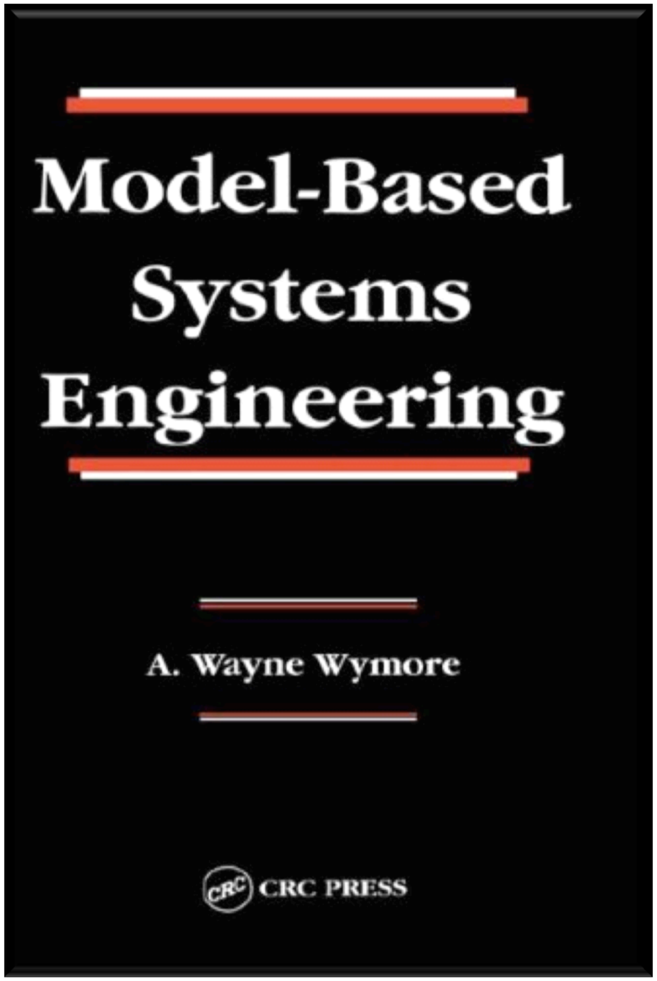
```
T1 T

12 1
23 3
24 2
```

```
T1 T2 T3 T4

12 12  5   6
23 34  8   8
24 23 2  34
```

```
R1 R2 R3 R4

12 12  5   6
23 34  8 8
24 23 234
```

Model-Based Engineering (MBE) for Computer-Based Systems

- Ensure system performance and reliability *prior* to system integration, test, or upgrade
  - □ Prediction through quantitative analysis & simulation based on architecture models
- System validation through model verification and implementation compliance checking



Model-Based Systems Engineering

A. Wayne Wymore

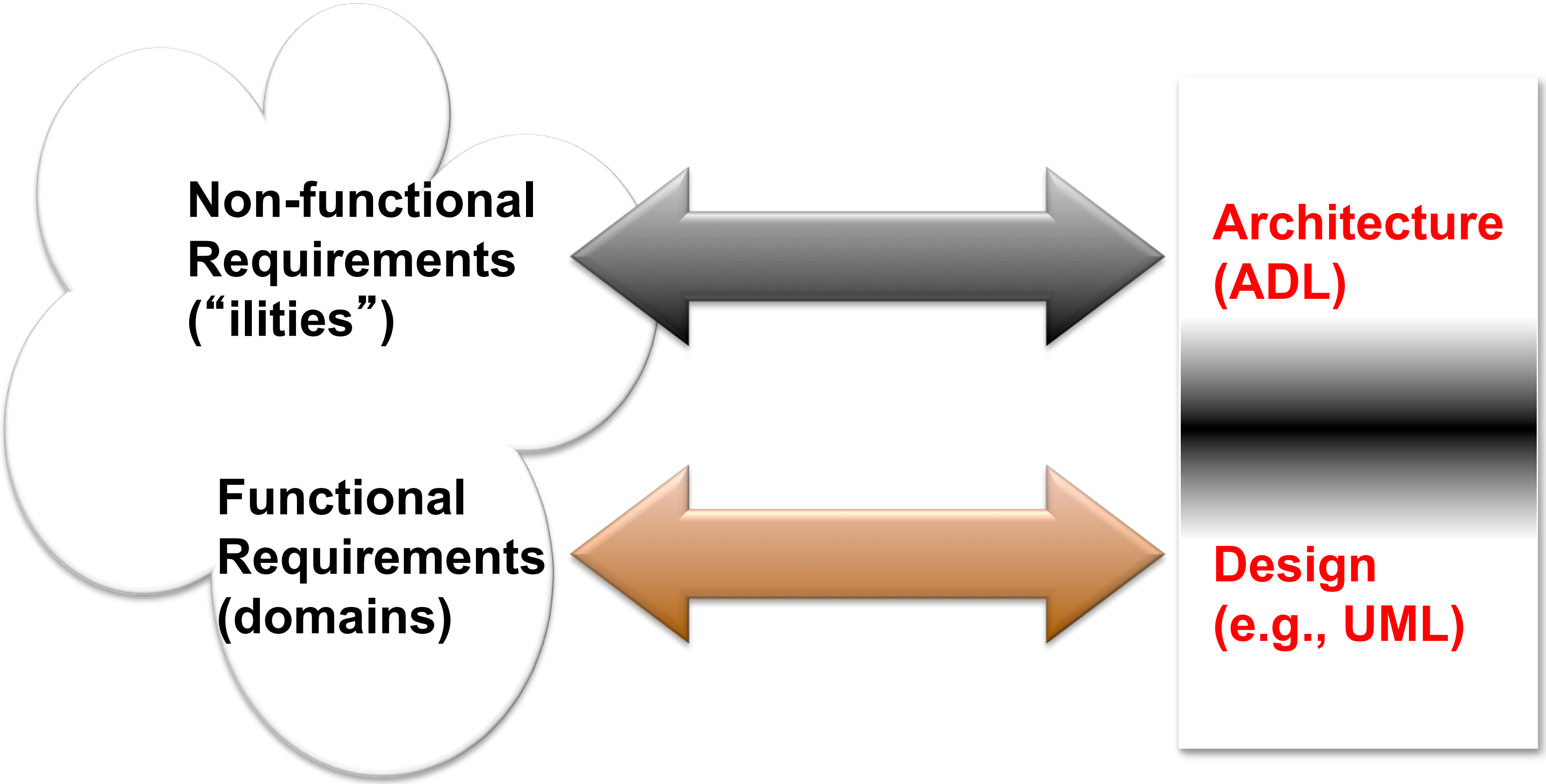
CRC PRESS

MDA can be like this sometimes…

Architecture vs. Design

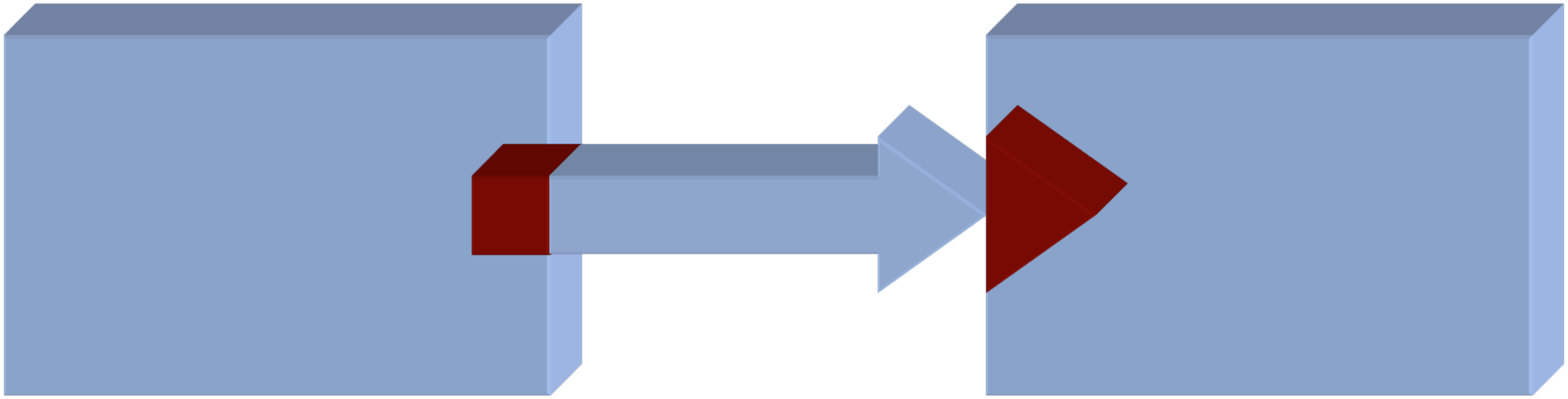
Architecture: where non-functional decisions are cast, and functional requirements are partitioned

Design: where functional requirements are accomplished


Non-functional Requirements ("ilities")
Functional Requirements (domains)
Architecture (ADL)
Design (e.g., UML)

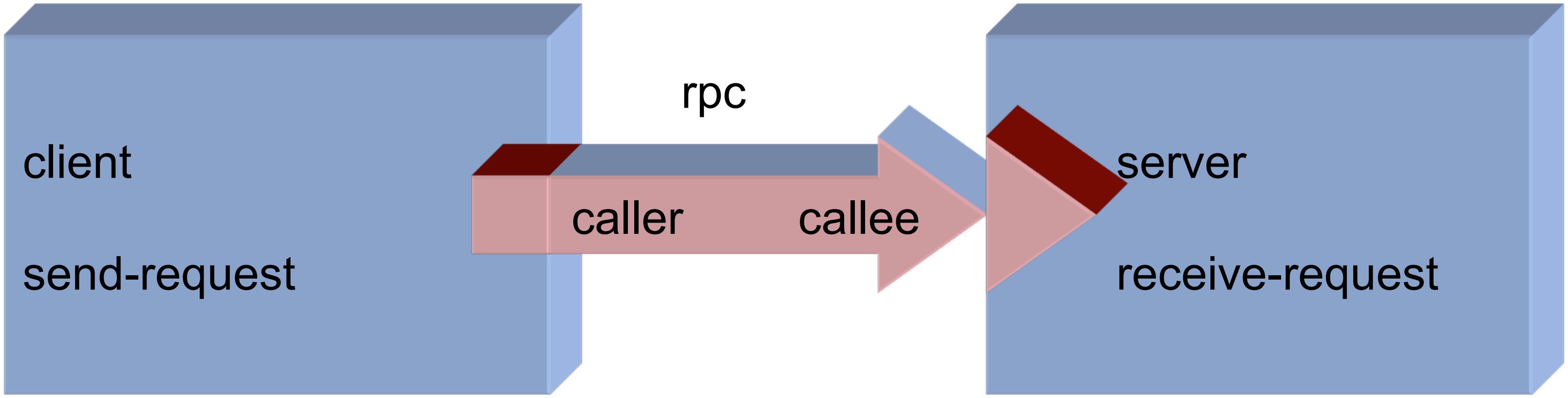
Software Architecture: ADL Perspective

- Architecture Description Language community agrees that Software Architecture is a set of components and the connections among them
  - □ Components
  - □ Connectors
  - □ Configurations
  - □ Constraints



An ADL Example

```
System simple_cs = {
    Component client = {Port send-request}
    Component server = {Port receive-request}
    Connector rpc = {Roles {caller, callee}}
    Attachments : {client.send-request to rpc.caller;
        server.receive-request to rpc.callee}
}
```



rpc

client

send-request

caller

callee

server

receive-request

Quality Attributes and Architectural Strategies

- Dependability
- Interoperability
- Usability
- Performance
- Adaptability
- Cost
- Schedule

Positive Effects

Negative Effects

- Assurance monitoring & control
- Layering
- Diagnostics
- Pipelining
- Architecture balance
- Parallelism
- GUI-driven
- API-driven
- Performance monitoring & control
- Change-source hiding
- COTS/reuse-driven

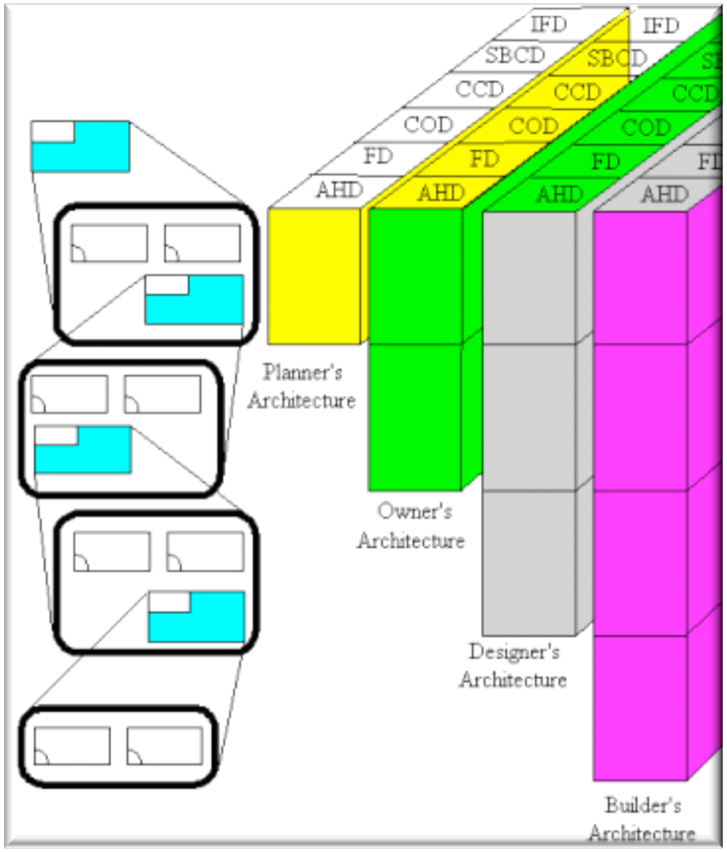
Example ADLs

- Industrial
  - AADL
  - SysML
  - xADL
  - UML 2.0
  - MetaH (Honeywell)

- Academic
  - ACME (CMU/USC)
  - Wright (CMU)
  - Unicon (CMU)
  - Aesop (CMU)
  - Rapide (Stanford)
  - SADL (SRI)
  - C2 SADL (UCI)
  - Lileanna
  - Modechart

ADL Upsides

- ADLs represent a formal way of representing architecture
- ADLs are intended to be both human and machine readable
- ADLs support describing a system at a higher level than previously possible
- ADLs permit analysis of architectures – completeness, consistency, ambiguity, and performance
- ADLs can support automatic generation of software systems



ADL Downsides

- Still disagreement on what ADLs should represent, particularly in the behavior aspects
- Representations sometimes difficult to parse and limited support by commercial tools
- Most ADL work today has been undertaken with academic rather than commercial goals in mind
- Most ADLs tend to be very vertically optimized toward a particular kind of analysis

Approaches to Architecture

Industrial Approach	Academic Approach
Focus on wide range of development issues	Focus on analytic evaluation of architectural models
Families of models	Individual models
Practicality over rigor	Rigorous modeling notations
Architecture as the “big picture” in development	Powerful analysis techniques
Breadth over depth	Depth over breadth
General-purpose solutions	Special-purpose solutions

Source: N. Medvidovic, USC

SAE Architecture Analysis & Design Language (AADL) Standard

- Designed for Model-Based Engineering
  - □ Notation for specification of runtime architecture of real-time, embedded, fault-tolerant, secure, safety-critical, software-intensive systems
- Fields of application:
  - □ Avionics, Aerospace, Automotive, Autonomous systems, Medical devices …
- Industry-driven International Standard
- www.aadl.info

Key Elements of SAE AADL Standard

- Core AADL language standard (SEI)
  - Textual & graphical, precise semantics, extensible
- AADL Meta model & XMI/XML standard (SEI)
  - Model interchange & tool interoperability
- UML profile for AADL
  - Subset of OMG MARTE profile being defined by MARTE
- Error Model Annex as standardized extension Fault/ reliability modeling, hazard analysis
- Behavior Annex
  - Externally observable behavior of components
- Programming Guidelines, Data Modeling Annexes

AADL: The Language

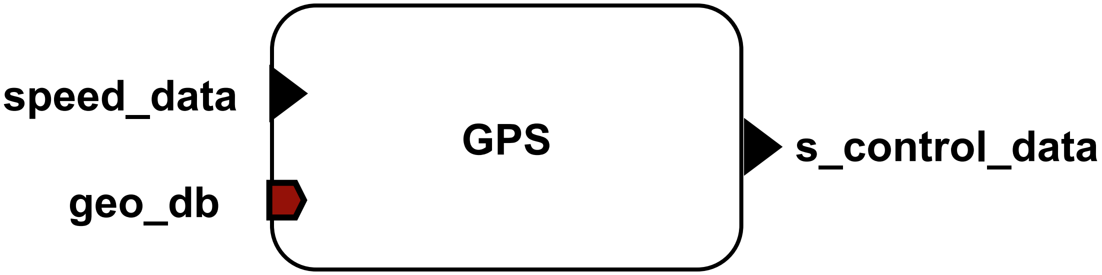
- Precise execution semantics for components & interactions
  - □ Thread, process, data, subprogram, system,
  - □ Processor, memory, bus, device, abstract component, virtual processor, virtual bus
- Continuous signal processing & stochastic event processing
  - □ Data, event, message communication, unqueued & queued
  - □ Synchronous call/return, Shared data access
  - □ End-to-End flow specifications
- Operational modes, fault tolerant configurations, levels of service
  - □ Modes & mode transition, error model annex
- Modeling of large-scale & configurable systems
  - □ Component variants, packaging of component classifiers, layered systems, parameterized templates, component arrays
- Accommodation of diverse analysis needs
  - □ User-defined properties, sublanguage extensions

System Type

System

```
system GPS
features
  speed_data: in data port metric_speed
     {SEI::BaseType => UInt16;};
  geo_db: requires data access real_time_geoDB;
  s_control_data: out data port state_control;

flows
  speed_control: flow path
       speed_data -> s_control_data;

properties SEI::redundancy => Dual;
end GPS;
```

{type}
extends
features
flows
properties



System Implementation

```
system implementation GPS.secure
subcomponents
  decoder: system PGP_decoder.basic;
  encoder: system PGP_encoder.basic;
  receiver: system GPS_receiver.basic;

connections
  c1: data port speed_data -> decoder.in;
  c2: data port decoder.out -> receiver.in;
  c3: data port receiver.out -> encoder.in;
  c4: data port encoder.out -> s_control_data;

flows
 speed_control: flow path speed_data -> c1 -> decoder.fs1
             -> c2 -> receiver.fs1 -> c3 -> decoder.fs1
             -> c4 -> s_control_data;
modes none;
properties arch::redundancy_scheme => Primary_Backup;
end GPS;
```

{implementation}
- extends
- refines type
- subcomponents
- calls
- connections
- flows
- modes
- properties

Some Standard Properties

Thread

```
Dispatch_Protocol => Periodic;
Period => 100 ms;
Compute_Deadline => value (Period);
Compute_Execution_Time => 10 ms .. 20 ms;
Compute_Entrypoint => “speed_control”;
Source_Text => “waypoint.java”;
Source_Code_Size => 12 KB;
```

Code to be executed on dispatch

File containing the application code

Processor

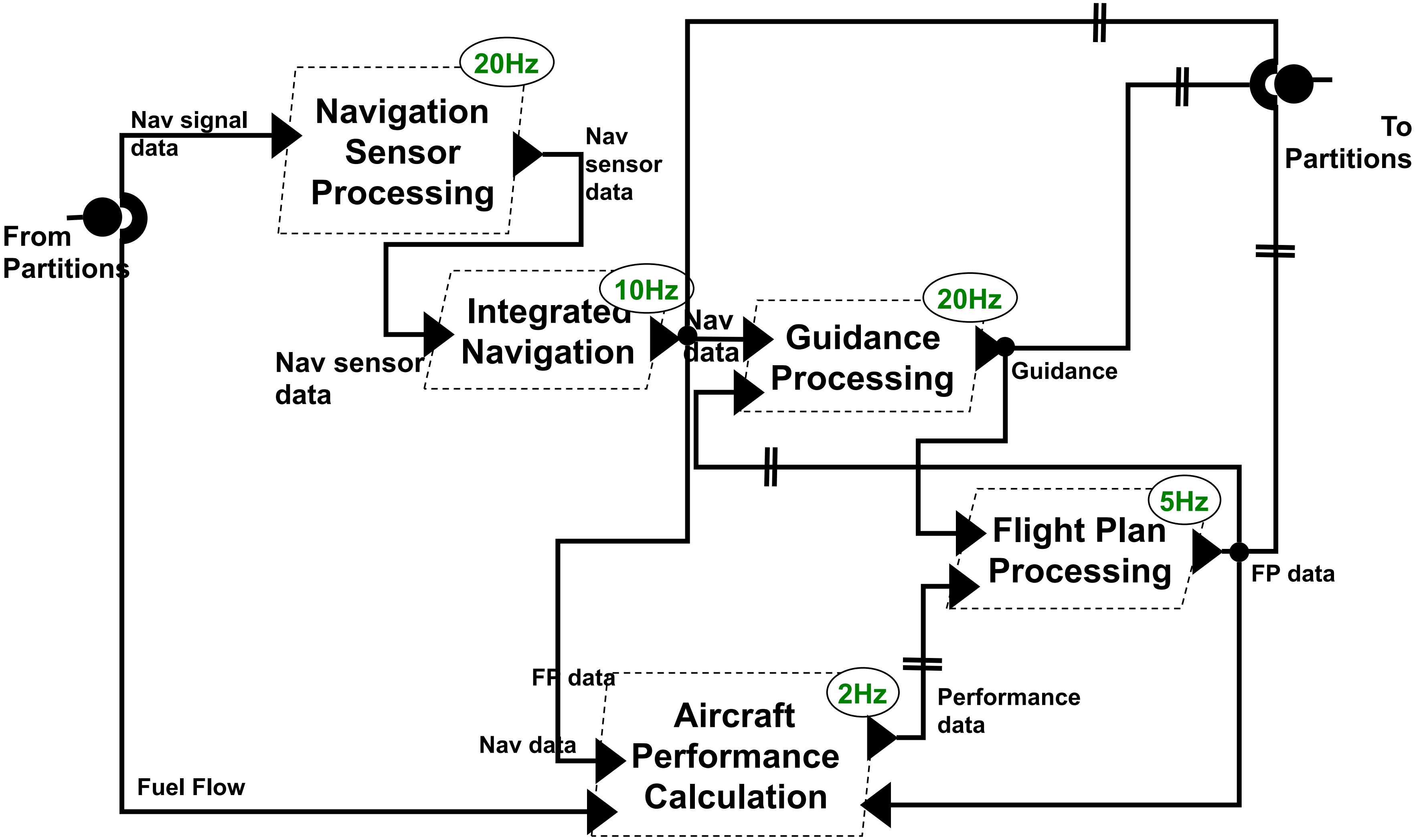
```
Thread_Swap_Execution_Time => 5 us.. 10 us;
Clock_Jitter => 5 ps;
```

Bus

```
Allowed_Message_Size => 1 KB;
Propagation_Delay => 1ps .. 2ps;
Bus_Properties::Protocols => CSMA;
```

Protocols is a user defined property

Example Graphical Specification: Flight Manager in AADL



Homework and Milestone Reminders

- Read Case Study Paper “SysML-based systems engineering using a model-driven development approach.”
  by Hans-Peter Hoffman
  - To be discussed in Class next Monday
  - Do assigned questions and bring document to class
  - Be prepared to discuss and even lead the discussion
- Milestone 3: Light-Weight Transformation Environment (see Milestone 3 assignment)
  - Due by 11:55pm, Friday, April 29th, 2011.