CSSE 490 Model-Based Software Engineering: Architecture Description Languages (ADL)

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu

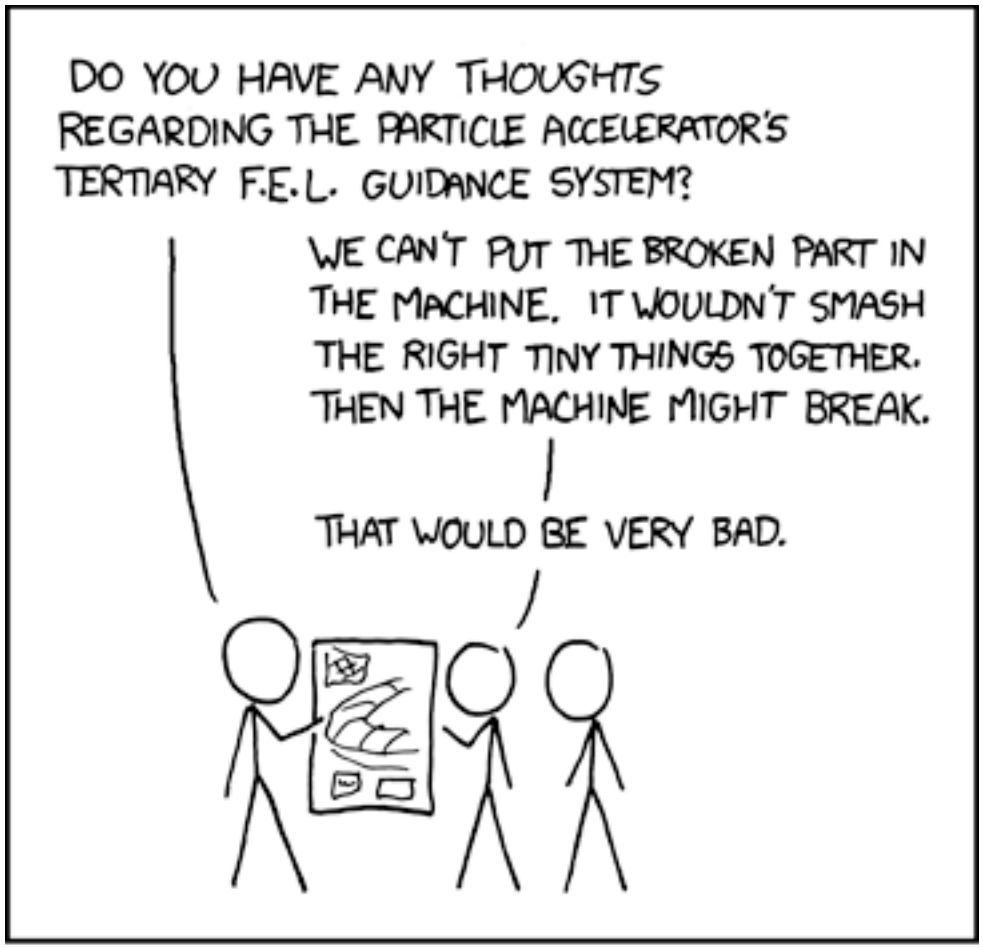
ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Learning Outcomes: MBE Discipline

Relate Model-Based Engineering as an engineering discipline.

- Discuss Case Study Paper
- Examine ADL Perspectives
- Discuss AADL

Einstein says… simple, but not too



Case Study/Homework:

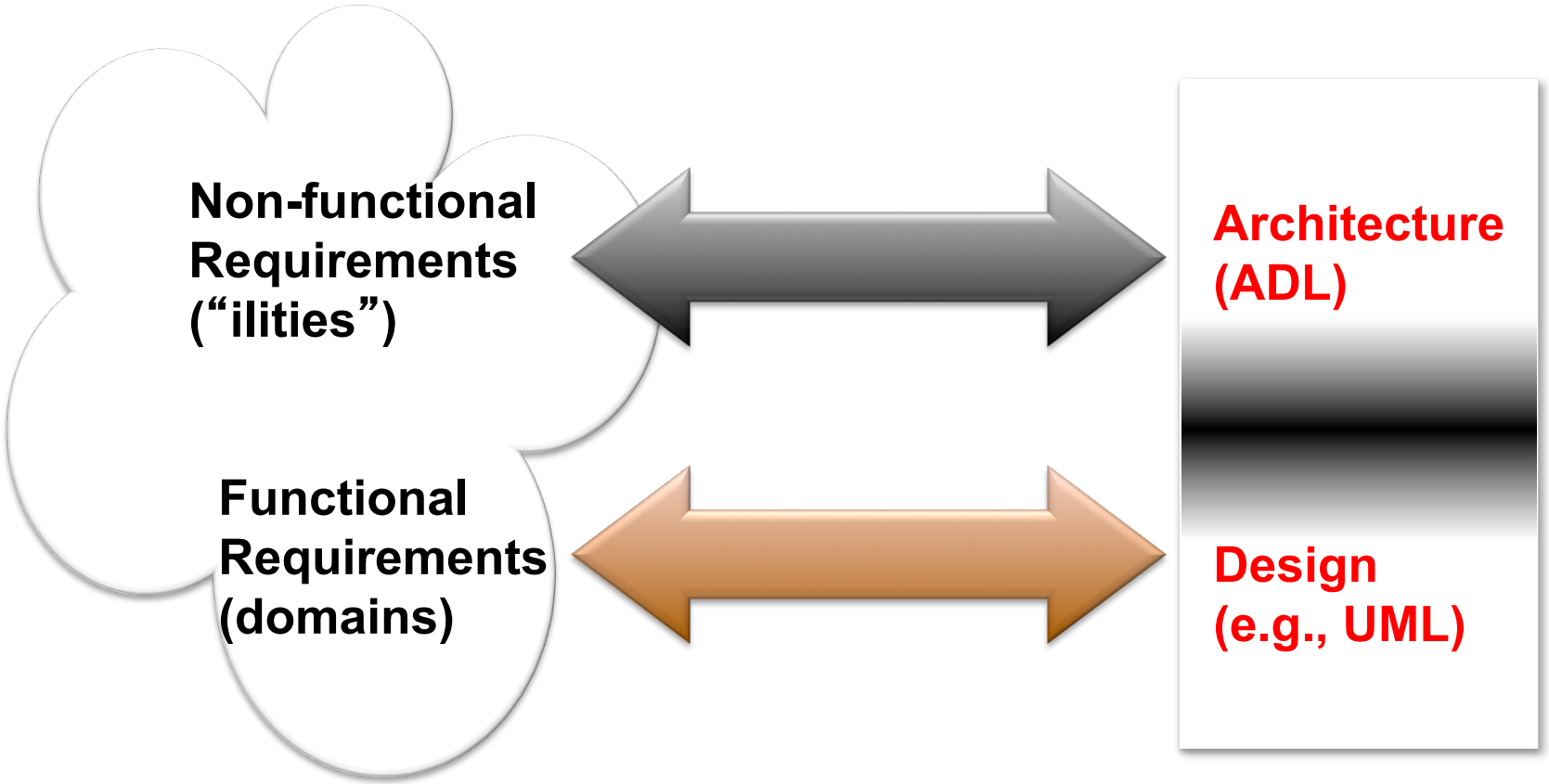
"SysML-based systems engineering using a model-driven development approach" ***by Hans-Peter Hoffman***

- What are some parallels between Unified Modeling Language (UML) and SysML?
- How can SysML be used to produce Models for the Model/Requirement and Test Data Repositories?
- What do you think of SysML for specifying Black-box activity diagrams from use cases? What about modeling behaviors be conveyed down to the lower levels?
- What do the authors say about SysML for articulating Architecture Design? Do you think that this is viable for software?

Recall: Architecture vs. Design

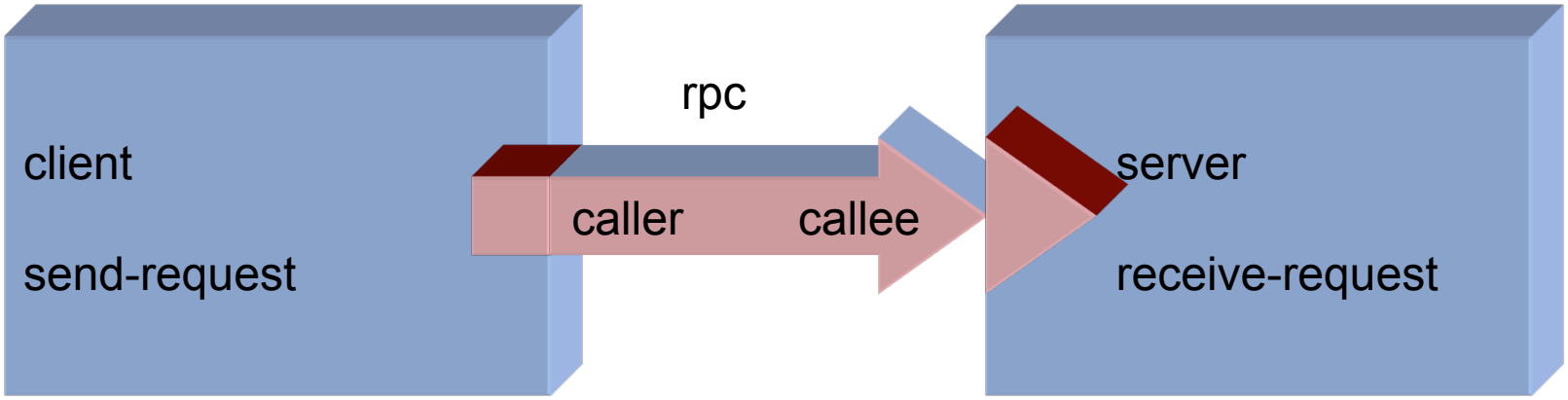
Architecture: where non-functional decisions are cast, and functional requirements are partitioned

Design: where functional requirements are accomplished



Recall: 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}
}
```

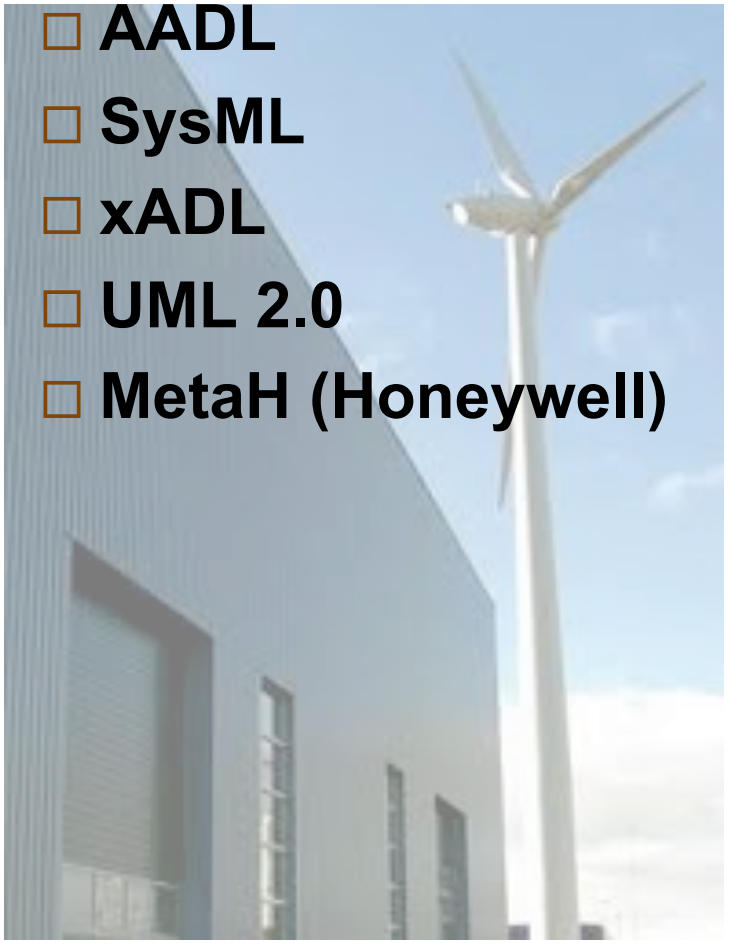


In principle, are System Architecture and Software Architectures substantially different? If so, how should the Architecture Description Languages (ADL) be different?

- Think for 15 seconds…
- Let's talk…

Example ADLs

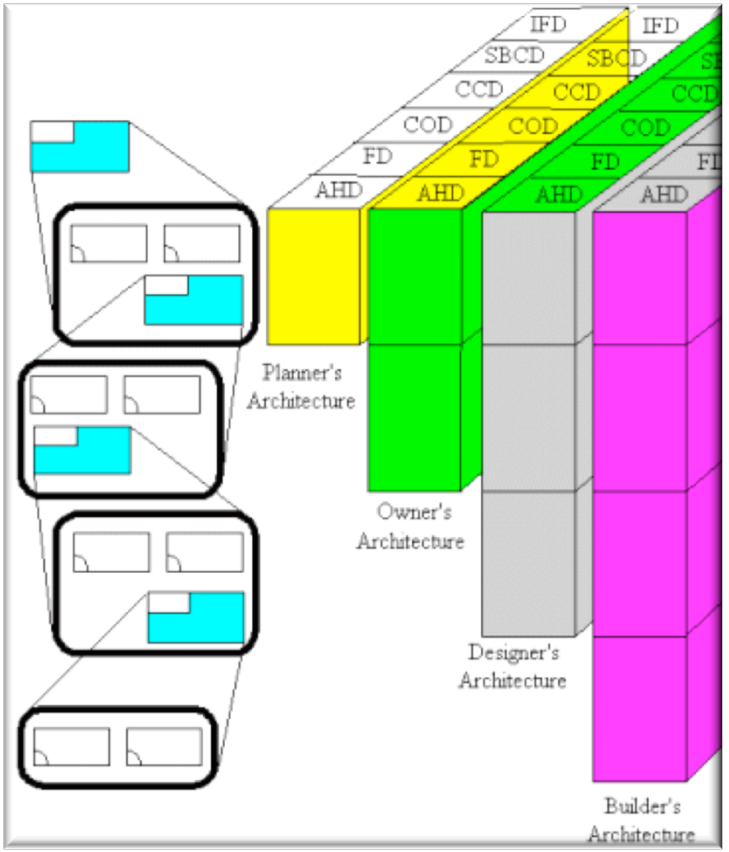
- Industrial
  - AADL
  - SysML
  - xADL
  - UML 2.0
  - MetaH (Honeywell)



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

ADL Upsides

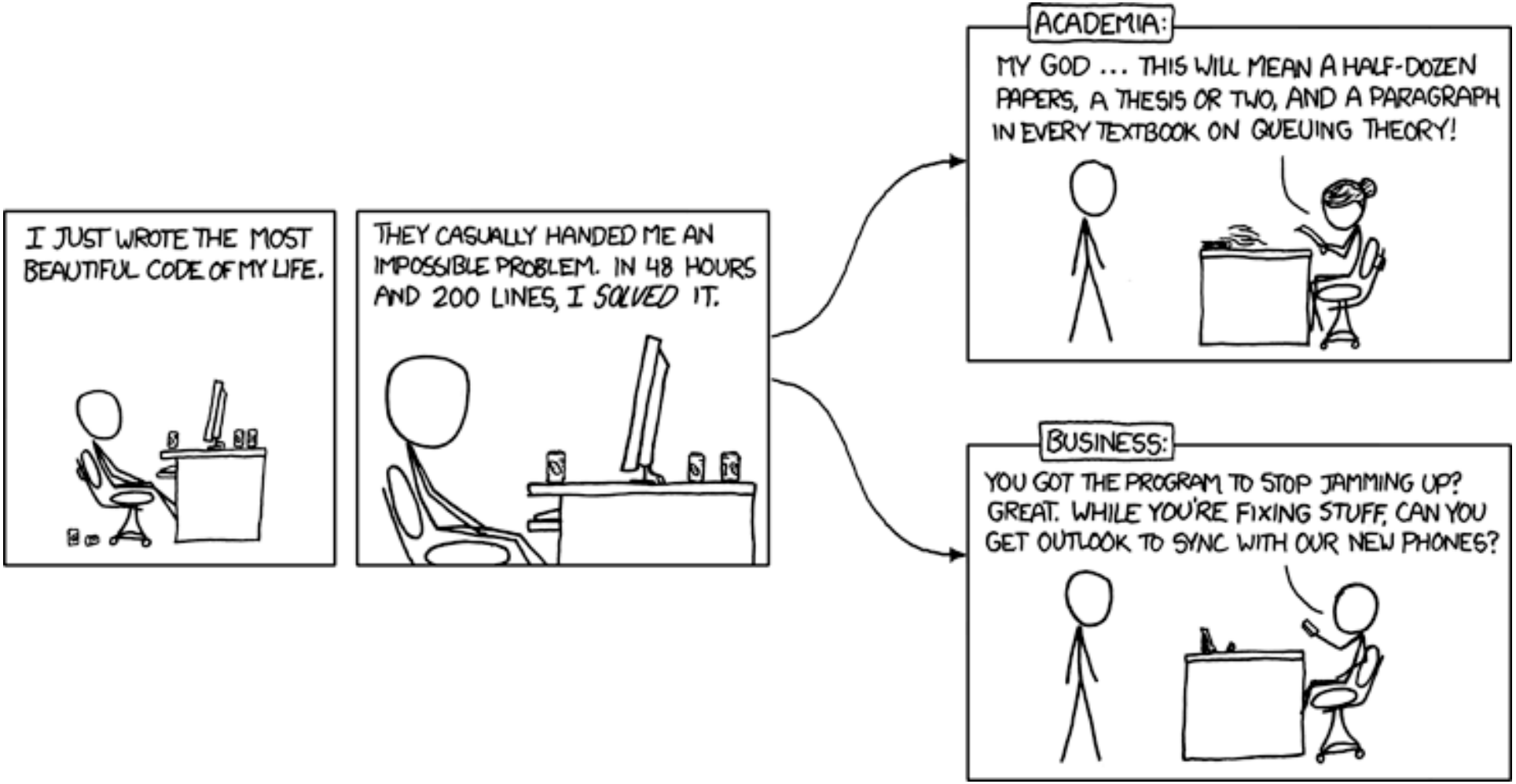
- ADLs represent a formal way of representing architecture
- ADLs are intended to be both human and machine readable
- ADLs support describing a system at a higher level than previously possible
- ADLs permit analysis of architectures – completeness, consistency, ambiguity, and performance
- ADLs can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

Industry versus Academia



Approaches to Architecture

Industrial Approach

- Focus on wide range of development issues
- Families of models
- Practicality over rigor
- Architecture as the "big picture" in development
- Breadth over depth
- General-purpose solutions

Academic Approach

- Focus on analytic evaluation of architectural models
- Individual models
- Rigorous modeling notations
- Powerful analysis techniques
- Depth over breadth
- Special-purpose solutions

Source: N. Medvidovic, USC

SAE Architecture Analysis & Design Language (AADL) Standard

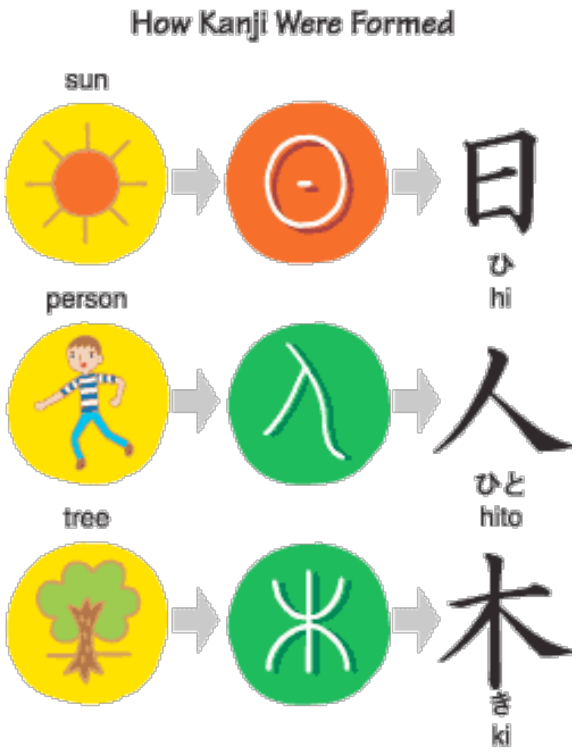
- Designed for Model-Based Engineering
  - Notation for specification of runtime architecture of real-time, embedded, fault-tolerant, secure, safety-critical, software-intensive systems
- Fields of application:
  - Avionics, Aerospace, Automotive, Autonomous systems, Medical devices …
- Industry-driven 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 1/2

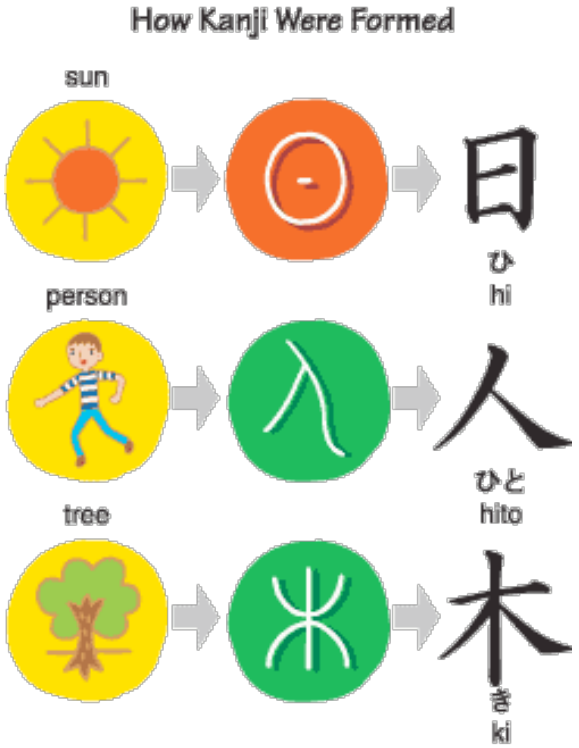
- Precise execution semantics for components & interactions
  - Thread, process, data, subprogram, system,
  - Processor, memory, bus, device, abstract component, virtual processor, virtual bus
- Continuous signal processing & stochastic event processing
  - Data, event, message communication, unqueued & queued
  - Synchronous call/return, Shared data access
  - End-to-End flow specifications



AADL

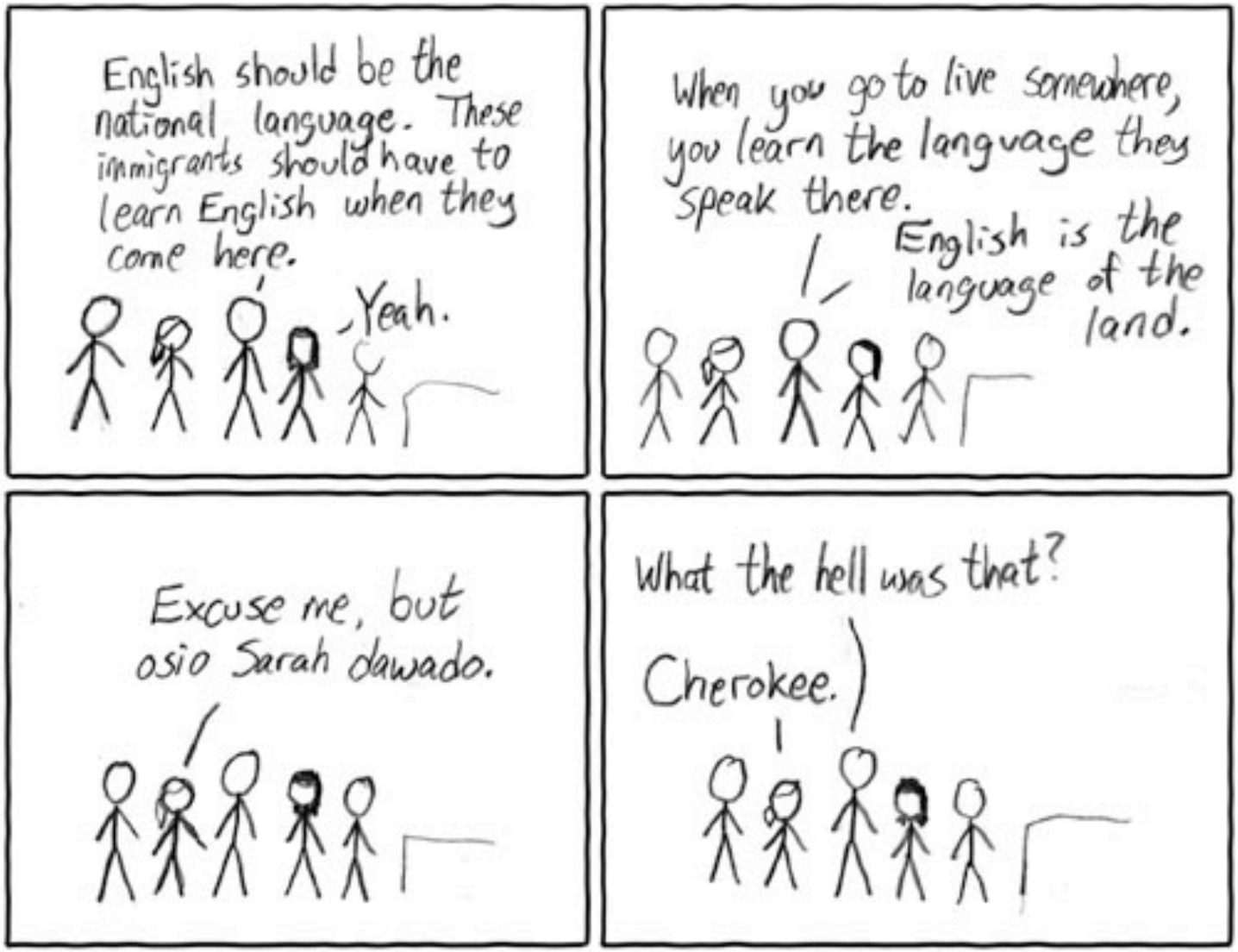
AADL: The Language 2/2

- Operational modes, fault tolerant configurations, levels of service
- Modeling of large-scale and configurable systems
  - Component variants
  - Packaging of component classifiers
  - Layered systems, parameterized templates, component arrays…
- Accommodation of diverse analysis needs
  - User-defined properties, sublanguage extensions



AADL

Language Etiquette ...



System Type

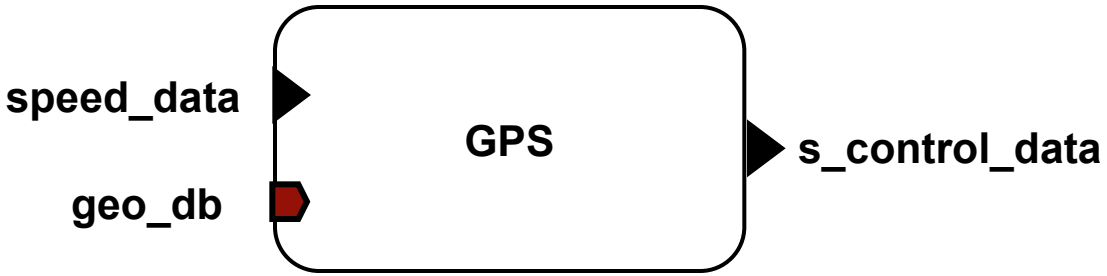
System

```
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

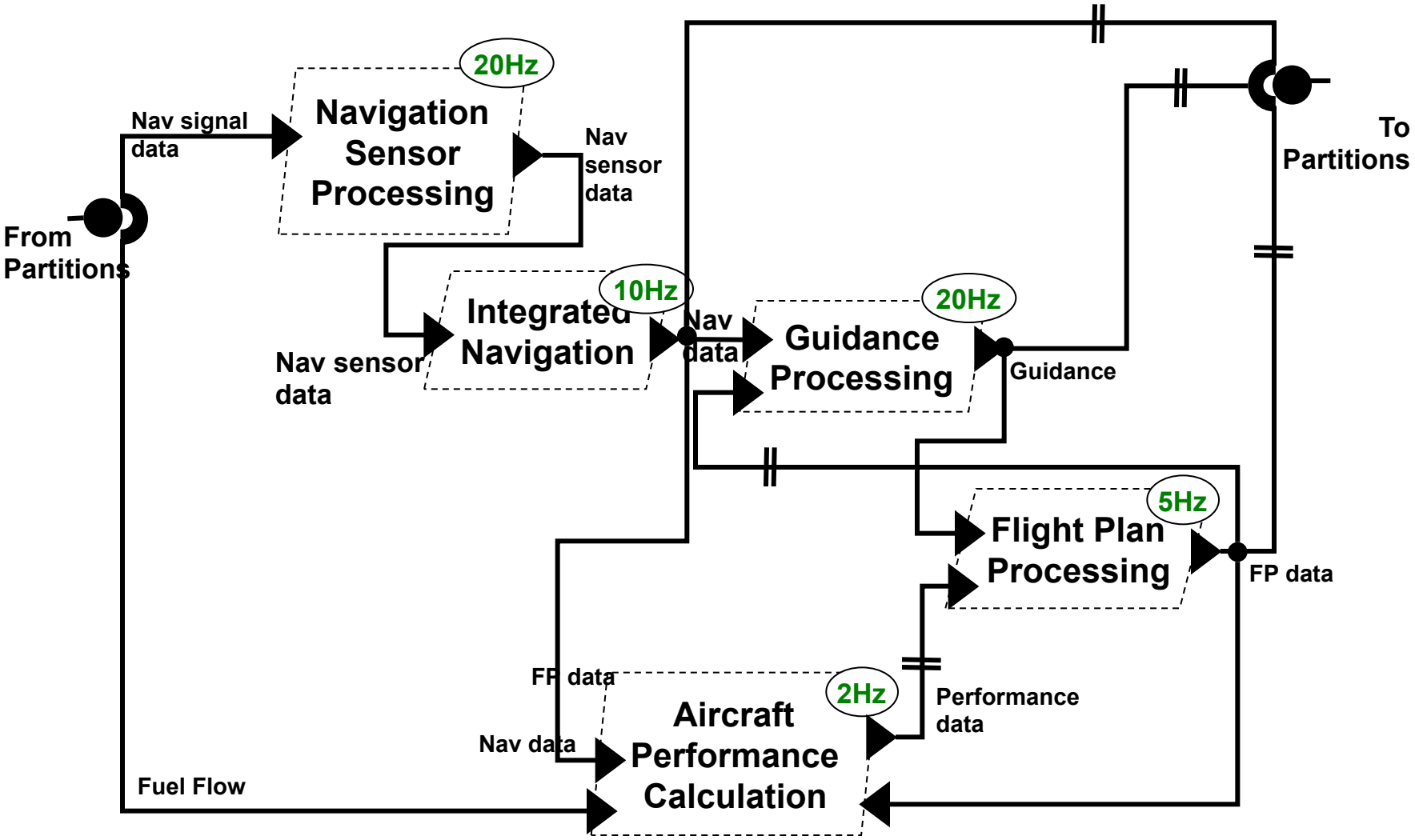
```
Thread_Swap_Execution_Time => 5 us.. 10 us;
Clock_Jitter => 5 ps;
```

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 Chapter 9 in text on Code Generation Techniques

- Milestone 3: Light-Weight Transformation Environment (see Milestone 3 assignment)
  - Due by 11:55pm, Tuesday, May 3rd, 2011.