How to describe an architecture?

- **Architecture Description Language (ADL)** - A language to describe complex system at a high level of abstraction that:
  - Exposes a system’s structure as a collection of interacting components
  - Allows engineers to reason about system properties, such as performance, schedulability, and security.

- **ADLs**
  - Aesop: Development of architectural styles
  - Adage: Avionics navigation and guidance
  - C2: User interface systems using event-based style
  - Meta H: Real time avionics control software
  - Acme: Generic ADL that supports extension
  - AADL: Safety-critical real time embedded system

Common Architectural Elements

```
System
  Component A - Property A
  Connection C - Property C
  Component B - Property B - Constraint B
```
Architecture Analysis and Design Language (AADL)

- Society of Automotive Engineers (SAE) standard
- Industrial ADL for safety critical domains like avionics, aerospace, and automotive
- Based on over 15 Years of DARPA funded research technologies
- Core AADL (Nov 2004 (V1), Jan 2009 (V2), Sept 2012 (V2.1), and Jan 2017 (V2.2))

**Standardized AADL Extensions**
- Error Model language for safety, reliability, security analysis
- ARINC653 extension for partitioned architectures
- Behavior Specification Language for models and interaction behavior
- Data Modeling extension for interfacing with data models (UML, ASN.1, ...)

**AADL Extensions in Progress**
- Requirements Definition and Assurance Language
- Synchronous System Specification Language
- Hybrid System Specification Language
- System Constraint Specification Language

**AADL Representation Forms**

**Strongly typed language with well-defined semantics**

**Textual syntax and graphical representation**

**Standardized XMI interchange format**
Architecture Analysis and Design Language?

- Precise execution semantics for components
  - Thread, process, data, subprogram, system, processor, memory, bus, device, virtual processor, virtual bus
- Continuous control & event response processing
  - Data and event flow, call/return, shared access
  - End-to-End flow specifications
- Operational modes & fault tolerant configurations
  - Modes & mode transition
- Modeling of large-scale systems
  - Component variants, layered system modeling, packaging, abstract, prototype, parameterized templates, arrays of components, connection patterns
- Accommodation of diverse analysis needs
  - Extension mechanism, standardized extensions

AADL Components and Connections Overview

![Diagram of AADL Components and Connections](attachment://AADL_components_connections.png)

applicative parts of the system
computing hardware and physical environment
composition of software and platform components or system components
Architecture Analysis and Design Language

- Component Type
  - Represents the externally visible characteristics of a component – name, component category, interfaces, properties, modes, and logical flows.

- Component Implementation
  - Represents a blueprint of its internal structure in terms of subcomponents
  - Defines the subcomponents, connections, calls, and modes (if they are not defined in the type); details the flows associated with the component type that traverse the various subcomponents; and add or modify properties

- A component may not require a component implementation declaration if it is not composed of other components.

- Note: A component classifier is either a component type or a component implementation.

Architecture Analysis and Design Language

- AADL elements
  - Components
    - Application software components: applicative parts of the system
      - Process
      - Thread
      - Data
      - Subprogram
    - Execution platform components: computing hardware and physical environment
      - Processor
      - Bus
      - Memory
      - Device
    - System: composite of software and platform components or system components
  - Connections
    - Port (data, event, and event data)
    - Subprogram
    - Parameters
    - Subcomponent access
  - Properties
Illustration of a Car System

- Wheel Rotation Sensor
- User Input
- Brake Pedal
- Engine
- Cruise Control System
- Antilock Brake System
- Stability Control System
- Processor
- Memory
- Bus
- User Display
- Actuator

System Component

- It represents a complete system or a subsystem or an abstraction of functionalities.
- It can contain components of any category, including other system components to create a component hierarchy.
- It provides a grouping mechanism.
- It enables incremental modeling.

- Identify potential system component(s) from the car system.
**System Component**

- In the context of the car system, system components are:
  - Cruise Control System
  - Antilock Brake System
  - Stability Control System
  - Execution Platform (Processor, Memory, and Bus)
  - Car

**You cannot have interfaces (ins and outs ports) on the top level system.**

---

**System Component (Type)**

```java
package SoftwareApps
public
system CruiseControl
features
  cc_ui: in data port;
  cc Ud: out data port;
properties
  none;
end CruiseControl;
end SoftwareApps;

package CarSystem
public
with SEI;
  system MyCar
  properties
    SEI: WeightLimit => 57.0 kg;
end MyCar;
end CarSystem;
```

---
Application Software Components

- It represents task architecture of the application in terms of processes, thread groups, and threads.
- It also represents application software artifacts such as data types, static and local data in terms of data component, and executable code such as functions and procedures, and source code libraries, in terms of subprograms and subprogram groups.

- Thread
  - Application task as a schedulable unit of concurrent execution

- Thread Group
  - An abstraction for logical organization of threads

- Process
  - Protected address space that is enforced at runtime

- Data
  - Static or local data and application data type

- Subprogram
  - Callable unit of sequential code

- Subprogram Group
  - An abstraction for logical organization of subprograms
Application Software Components

- At this level of abstraction, we haven’t modeled any application software components.
- Some examples of application software components are:
  - ScanInputPorts: A thread to acquire data from sensors and apply digital filters where required.
  - ComputeThrottleSetting: A thread to covert and scale the output to the throttle.
  - CruiseControlProcess: A process that contains the ScanInputPorts thread and ComputeThrottleSetting thread.

Execution Platform Components

- It represents physical resources of the computer system, such as processor, memory, and bus, as well as elements of the external physical environment.
Execution Platform Components

- **Processor**
  - Represents the hardware and associated software that is responsible for scheduling and executing threads.
- **Virtual Processor**
  - Represents a logical resource for scheduling and executing software.
- **Memory**
  - Represents storage components for data and executable code.
- **Bus**
  - Represents the physical hardware and associated communication protocols required to support the interactions among physical components.
- **Virtual Bus**
  - Represents a logical abstraction such as a virtual channel or communication protocol.

Some examples of Execution platform components are:
- CAN bus (bus component)
- PCI bus (bus component)
- RAM (memory component)
- ROM (memory component)
- Processor (processor component)
Device Component

- Device abstractions represent entities that interface with or are part of the external environment of an embedded software system. They are characterized by their interface, their internal structure is not modeled.
- Devices can represent a physical entity or a software simulation of a physical entity.
- Examples:
  - Sensors
  - Actuators
  - Engines
  - Global Positioning Systems
  - Cameras

Identify potential device component(s) from the car system.

Device Component

- In the context of the car system, devices components are:
  - Wheel Rotation Sensor
  - User Input
  - Brake Pedal
  - User Display
  - Actuator
  - Engine
Demo

Cruise Control System is to keep the speed of the vehicle equal to the desired speed, as selected by the driver. It monitors the speed of the vehicle via wheel rotation, and outputs a control signal in proportion to the difference between the current speed and the desired speed (the error) to the throttle actuator. It also monitors the brake pedal for state. If the cruise control is engaged, and the brake pedal is depressed, the controller will disengage the throttle. The cruise control receives input from the user input indicating that cruise control is on or off, and that a speed set point has been given (the cruise control is engaged). It provides a status indication to the driver indicating that it is on or off, and also if it is engaged or disengaged.
Individual Assignment

Cruise Control System is to keep the speed of the vehicle equal to the desired speed, as selected by the driver. It monitors the speed of the vehicle via wheel rotation, and outputs a control signal in proportion to the difference between the current speed and the desired speed (the error) to the throttle actuator. It also monitors the brake pedal for state. If the cruise control is engaged, and the brake pedal is depressed, the controller will disengage the throttle. The cruise control receives input from the user input indicating that cruise control is on or off, and that a speed set point has been given (the cruise control is engaged). It provides a status indication to the driver indicating that it is on or off, and also if it is engaged or disengaged.

Plane for the next couple of weeks

- Module view documentation: Due Tuesday 03/13 by 6:00 pm
- Monday 03/19 @ 6:00 pm: Lecture C&C Views
  - Exam 1 assigned
- Tuesday 03/20 Lecture AADL
- Tuesday 03/27: No class (Exam work)
- Sunday 04/01 Exam1 due by e-mail @ 11:59