

# **GENERAL DYNAMICS**

## Advanced Information Systems

### **AADL and the Plug and Play Weapon**

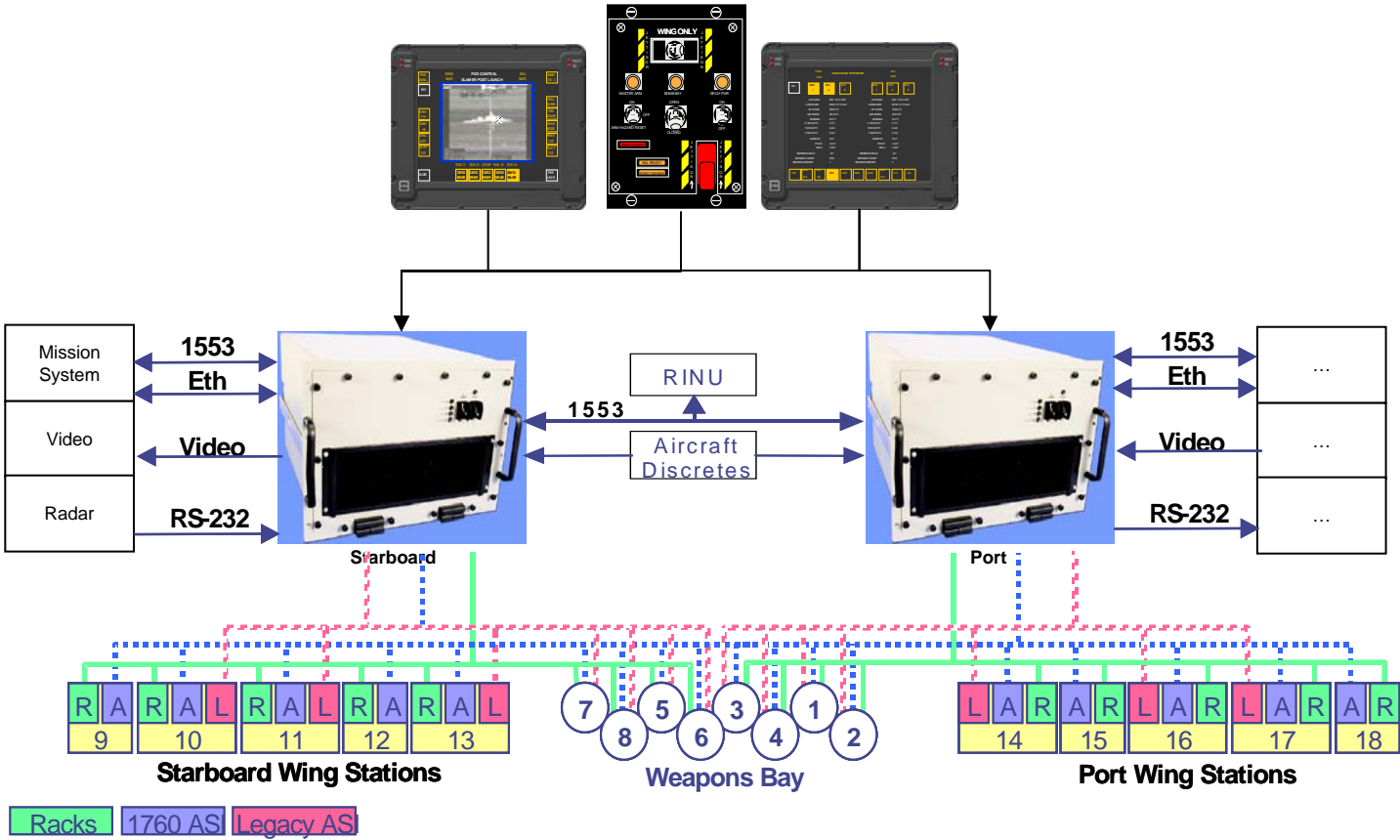
Early Experience Using the Architecture Analysis & Design Language

TC04

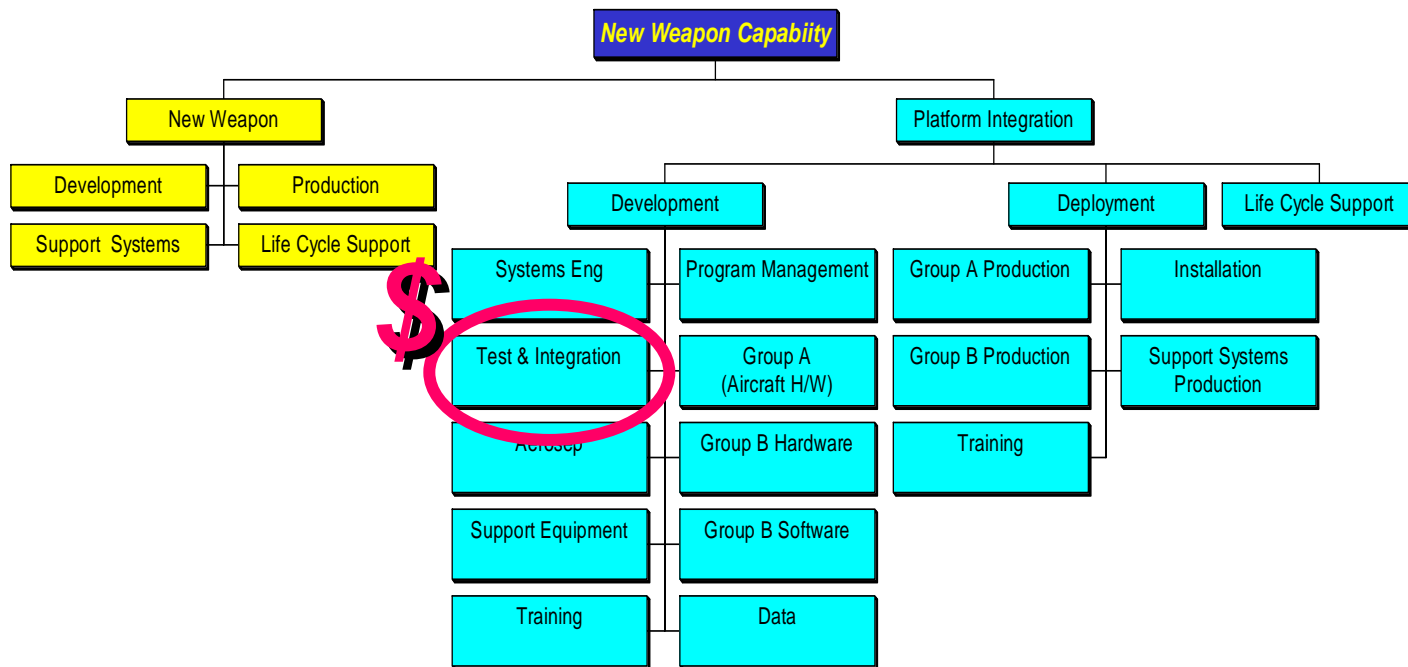
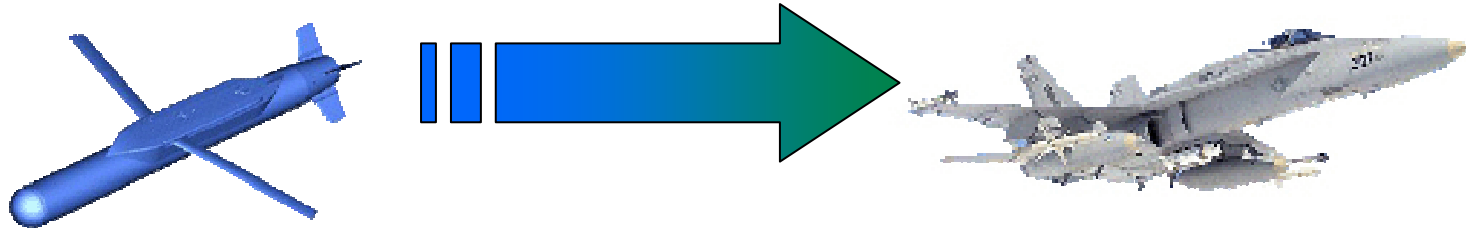
**Yves LaCerte**

3 November 2004

# A Weapon Management System



# Introducing a New Weapon



# Test and Integration Challenges

---

- Platform Perspective
  - ↗ Do not change the aircraft or store application software
  - ↗ Provide relevant functions and data
  - ↗ ***Observe resource, performance and timing constraints***
  
- Weapon Perspective
  - ↗ Identify/define relevant functions and data
  - ↗ ***Identify/define resource, performance and timing constraints***

# The Plug and Play Concept

---


- Demonstrate interoperability at design time
  - ↗ In terms of Functionality and Data
    - Open system approach via standards
  - ↗ ***In terms of Quality of Service (QoS)***
    - ***Safety, real-time, reliability, fault tolerance, security....***
- A system that can exchange information and services with multiple systems is more interoperable than one that can't
  - ↗ Can we measure the quality of interoperability?
  - ↗ Can we formulate strategies to improve interoperability based on measured quality?

# The Plug and Play Concept - Standards

## APIs / Protocols

|   |                                    |
|---|------------------------------------|
| Store Control API (SCA)                                 |                                    |
| Mil-Std-1760  | Universal Armament Interface (UAI) |
| Generic Aircraft-Store Interface Framework, SAE AIR5532 |                                    |
| Mil-Std-1553  |                                    |

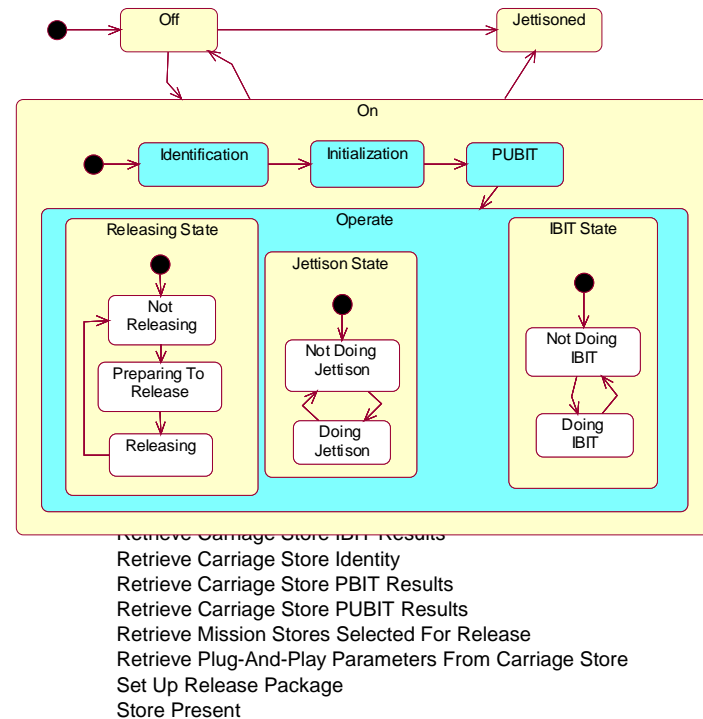
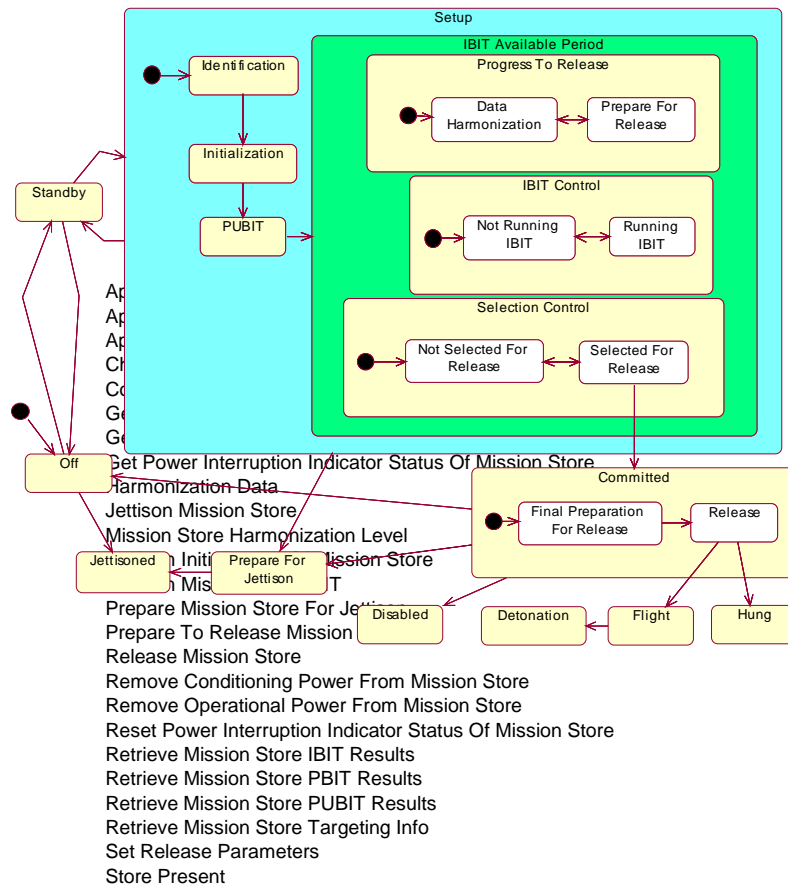
## Data



```

<?xml version="1.0" ?>
<catalog>
  <carrier id=" Carriage Store xyz ">
    <Emergency_jettison_permitted>Yes</Emergency_jettison_permitted>
    <IBIT_available_>Yes</IBIT_available_>
    <IBIT_completion_sign>event name xyz</IBIT_completion_sign>
  </carrier>
  <store id=" JCM xyz ">
    <Emergency_jettison_permitted>Yes</Emergency_jettison_permitted>
    <IBIT_available_>Yes</IBIT_available_>
    <IBIT_completion_sign>event name xyz</IBIT_completion_sign>
  </store>
</catalog>
  
```

# Store Control API (SCA)



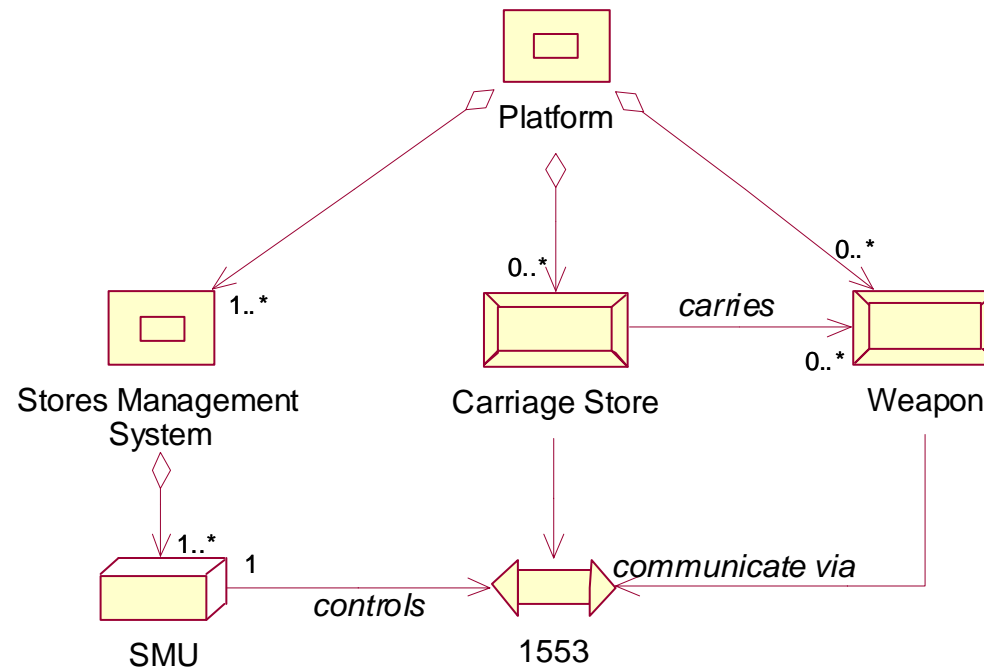
# The Plug and Play Concept - QoS

---

- Architecture Analysis and Design Language
  - Walk through architecture notions of a Stores Management System
  - Ask yourself “What analyses will I do?”
  - The answer will drive your modeling approach

# Platform

Diagram created from ROSE using Dr. Colbert's add-in



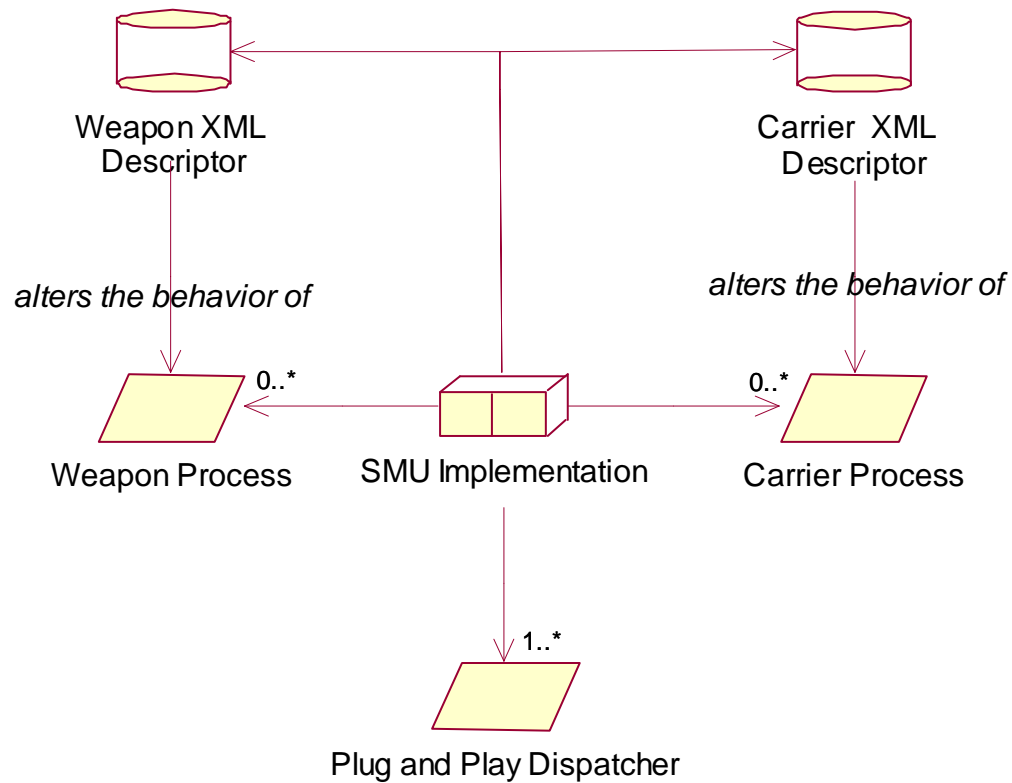
Rich *device* feature set? Enough to completely characterize a new weapon?  
Consider data .... Consider action language...

Can the system scale up? Capacity Planning... Queueing Network Model....

Interoperable models

# System

Diagram created from ROSE using Dr. Colbert's add-in



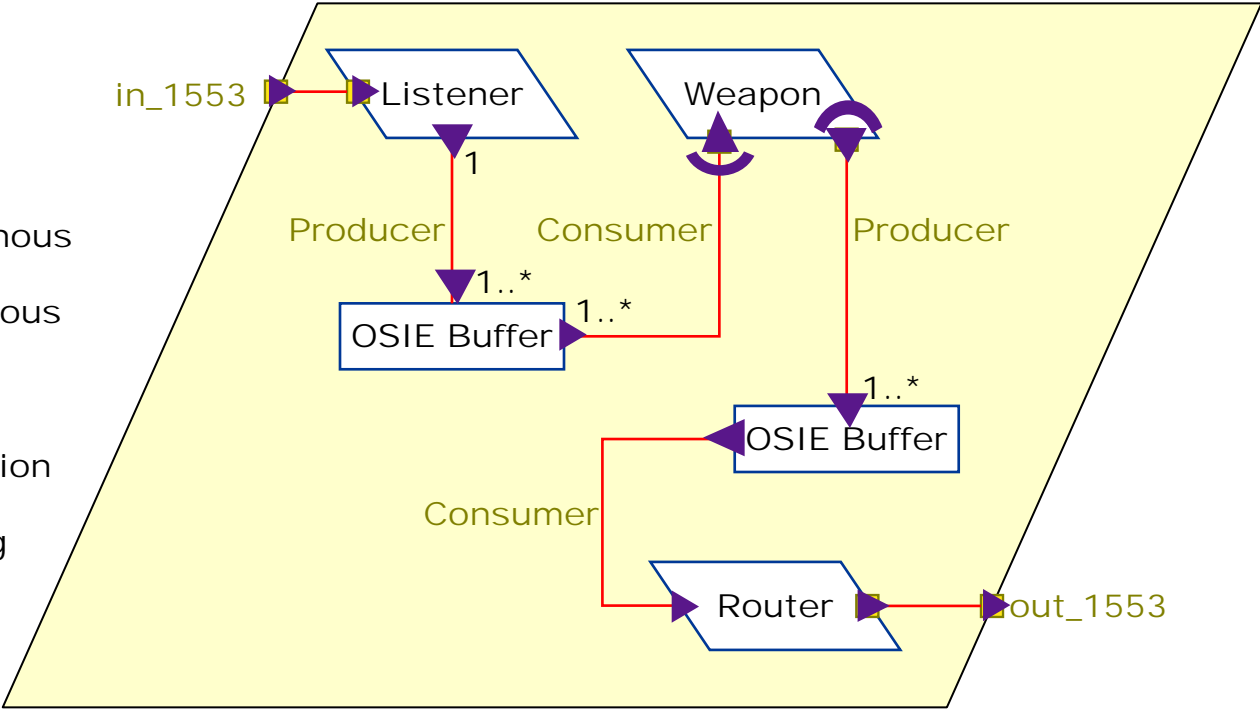
Can the XML Descriptor be used to characterize a device?  
Consider a weapon ontology.....  
Process vs partitions

# Weapon Process Implementation

Diagram adapted from Rhapsody

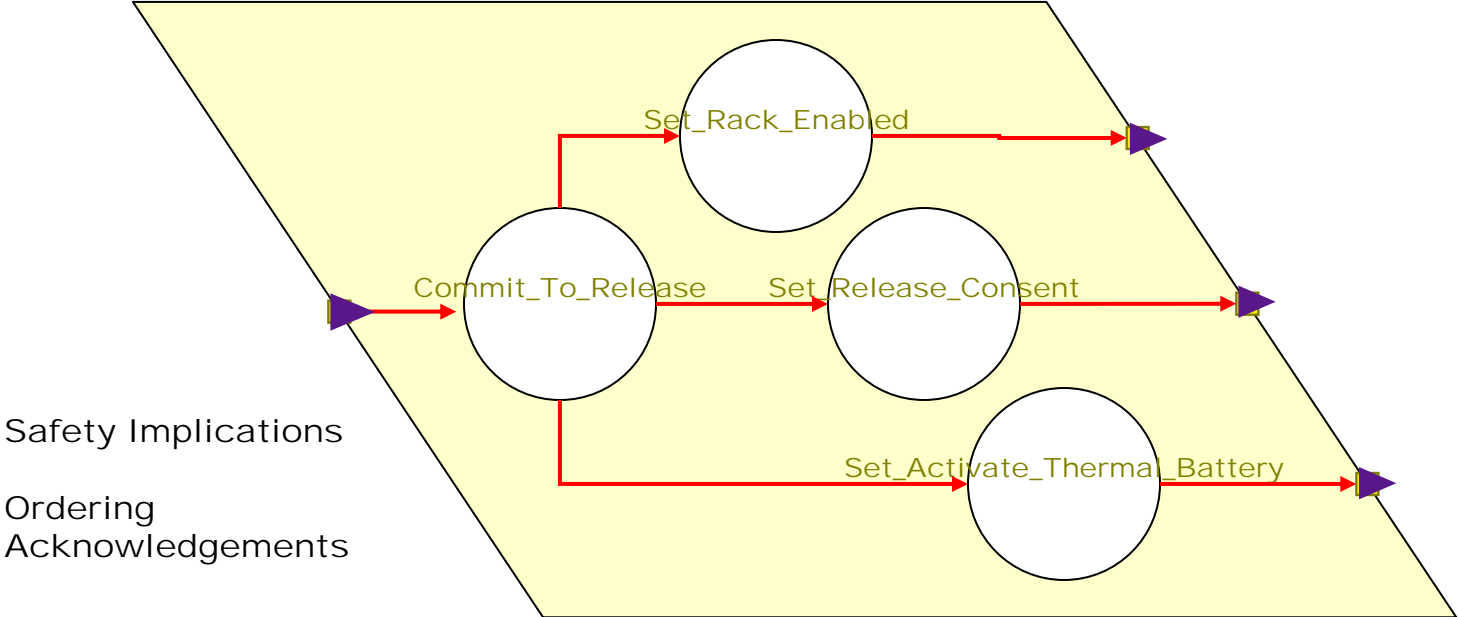
Asynchronous vs Synchronous

Notification vs polling

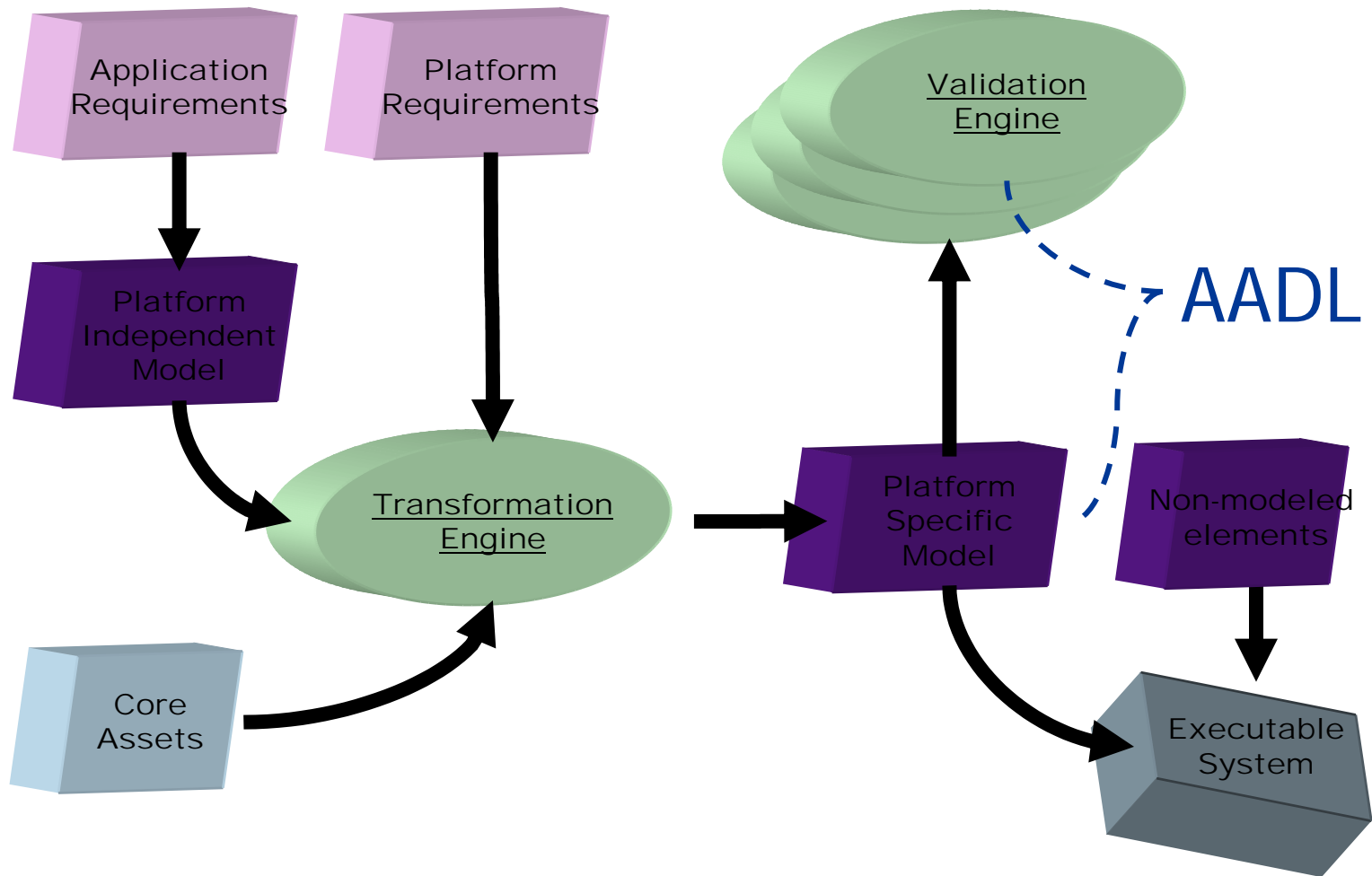


# Weapon Task Implementation

Diagram adapted from Rhapsody



# AADL and MDA

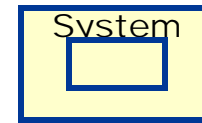




**Backup**

# Application Components

- System
  - ↗ Hierarchical organization of components
- Process
  - ↗ Protected virtual address space
- Thread group
  - ↗ Organization of threads in processes
- Thread
  - ↗ A schedulable unit of concurrent execution



# Application Components

- Data
  - Potentially sharable data

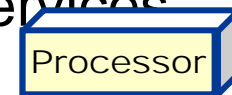


- Subprogram
  - Callable unit of sequential code

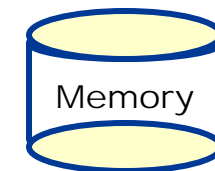


# Execution Platform Components

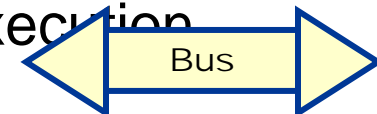
- Processor
  - ↗ Provides thread scheduling and execution services



- Memory
  - ↗ Provides storage for data and source code



- Bus
  - ↗ Provides physical connectivity between execution platform components



- Device
  - ↗ Interface to external environment

