



---

# SW Architectural Modeling and Assurance with AADL

**NASA SARP TIM  
August 21-24, 2012**

**PI: Dr. Michela Muñoz Fernández**

**Dr. Shang-Wen Cheng, Kenneth Evensen  
Dr. Allen Nikora, Dr. Kathryn Anne Weiss**

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program. This activity is managed locally at JPL through the Assurance and Technology Program Office.



08/21/2012

Copyright 2012 California Institute of Technology. Government sponsorship acknowledged.



# Agenda

---

- **Problem statement, background, and overview**
- **Major accomplishments:**
  - Software Architecture Modeling and Assurance with AADL for the JPL SMAP Project
  - Software Architecture Modeling and Assurance with AADL for the JPL Juno Project
  - Papers published in 2012
  - Plug-in to convert UML to AADL using MARTE profile
- **Future plans and direction:**
  - SMAP
  - Juno
  - Grace follow-on
- **References**



# Problem Statement

- How to accurately represent the behavior of
  - ... complex systems by properly selecting the key attributes
  - ... particularly when model-based techniques are increasingly used for their development
- Can new tools and technologies be used in future missions starting at earlier phases to reduce risk?

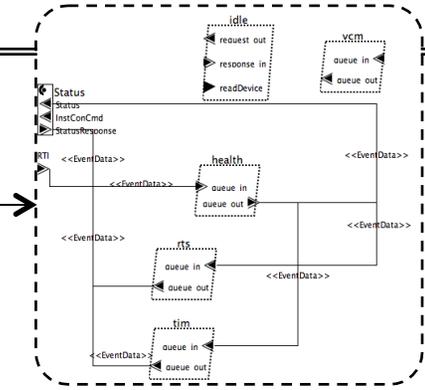
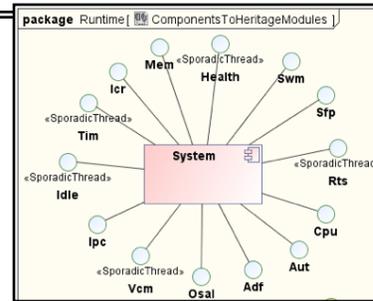


# Enabling Formal Mission Assurance

National  
Aeronautics and  
Space  
Administration

## Historically

- FSW has been developed w/o characterizing *performance* of the real-time system being built ... until integration!
  - UML dev't models insufficient
  - Assurance: document-based
- Finding execution-related issues at that point is extremely costly



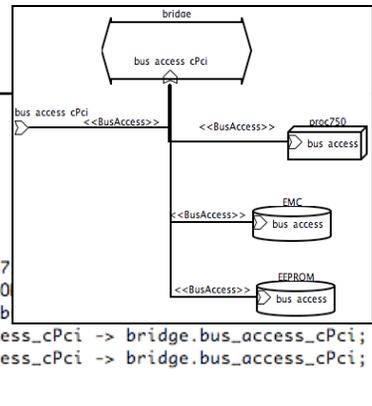
## Now and the *Future*?!

- **AADL** (Analysis and Architecture Design Language) **model shows execution interactions between high-level system components**
  - Enables early quality attribute analyses
- **AADL reduce possibility of doing rework later in the lifecycle**
  - Increases confidence at gateway reviews, by providing independent, semantically accurate analyses

```

system implementation board750.msap
subcomponents
  bridge: bus bridge;
  proc750: processor proc750;
  EEPROM: memory EEPROM;
  EMC: memory EMC;
connections
  bus_access_01: bus access bridge -> proc7
  bus_access_02: bus access bridge -> EEPRO
  bus_access_04: bus access bridge -> EMC.b
  BusAccessConnection1: bus access bus_access_cPci -> bridge.bus_access_cPci;
  BusAccessConnection2: bus access bus_access_cPci -> bridge.bus_access_cPci;
end board750.msap;

```

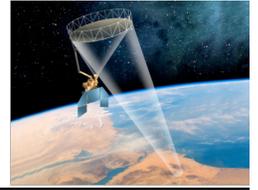


## Values to NASA:

- Reduction of risk of increased cost due to rework later in lifecycle
- Rigor and formalism added to development lifecycle and assurance activities
- Formal semantics provide accurate performance analyses at gateways
- Provision of not *just* software or hardware assurance
  - but *system* assurance, and therefore **mission assurance!**



# Task Background



- **Objective – Demonstrate use of AADL to analyze *quality attributes* of integrated spacecraft flight software architecture**
  - In the context of verification and validation activities
  - Using Architecture Analysis & Design Language (SAE AS5506/A)
    - Strict Formalisms and Semantics
    - Formal analysis framework to conduct early assessment of project quality attribute requirements using Figures of Merit
  - Applying Architectural Modeling for Aerospace Software Engineering (AMASE) – IEEE-1471 compliant
- **Project – Soil Moisture Active Passive (SMAP)**
  - JPL, proposed Earth-orbiting mission, exp. launch 2014
- **Continuation of research work**
  - FY09 – Developed framework using an MDS example
  - FY10/11 – Apply framework to real JPL flight project, SMAP
  - FY11/12-Apply framework to another real JPL flight project on its way to Jupiter-Juno





# Software Architecture Modeling and Assurance with AADL for the JPL SMAP Project

---

National  
Aeronautics and  
Space  
Administration

- **AADL architecture model has consistently augmented SMAP FSW gateway deliveries (hence *not* shelf-ware)**
  - Showed: detailed design continually consistent with software architecture
  - Demonstrated preliminary implementation also consistent
- **Assurance able to provide independent resource analysis**
  - Conducted CPU resource analysis, presented at SMAP FSW PDR
  - Results indicated large CPU margin
- **Provided performance analyses to SMAP FSW Team**
  - Bus Bandwidth Analysis
  - Memory Resource Analysis
  - Deadlock Analysis (UPPAAL)
  - Reachability Analysis (UPPAAL)
- **Applicability**
  - Real-time embedded software systems – the types of systems NASA builds!
  - Very useful in the rigorous analysis of these systems, as primary notation
  - Even if only UML, independent AADL models increase confidence and reduce risks in the avionics software architecture



# Software Architecture Modeling and Assurance with AADL for the JPL Juno Project

---

- **Problem statement:**
  - How to avoid or minimize Juno command errors?
    - By modeling the Juno spacecraft and applying new tools, some errors could have been revealed in real time.
- **Substantial modeling of the Juno Spacecraft (primarily Avionics view):**
  - C&DH, science, telecom, flight software
- **Have stored models in model repository (library)**
  - Subversion server maintained by SQA
- **Developed a series of reliability plugins for OSATE**
  - Will use on JUNO, then SMAP:
    - End to end data flow: data latency analysis-> revealed scenarios where commanding errors can occur.
    - Data generation and memory analysis revealed the scenario when data overflow would occur- could have prevented loss of science data.



# Software Architecture Modeling and Assurance with AADL for the JPL Juno Project-data latency analysis

---

- **Example of one data latency analysis (proof of concept):**
  - **JADE Mass Memory Overflow during High Voltage Checkout (ISA 50603, criticality 3).**

During the activities to close out the day on 11/17, the configuration for the JADE instrument was changed from LVENG to HVENG after discussion with the Mission Manager: the jad\_hveng\_hvenable.log sequence was sent at 04:13, which put JADE in a mode which produced telemetry at approximately 18 kbps. This filled their 541 Mbits soft partition (SP07) at approximately 12:43 UTC. The question of data rate production rate in the new configuration was asked, but was not answered or not answered properly. The new configuration produced data which overfilled the instruments memory partition leading to remaining data being discarded.

- Immediate fix: Start of activities on day 5 was delayed for 75 minutes while the memory partition emptied enough to proceed with commanding, and a determination was made that the JADE instrument and spacecraft were in an state to proceed with the day's activity. The error triggered a separate anomaly, which added to the delay, but was found to not interfere with continuing checkout (ISA 50604 Discarded Frames and Data Volume for SP07 Much Greater than Production Rate).
- Proximate cause: Command Product content not fully understood/communicated for use at different time.



# Software Architecture Modeling and Assurance with AADL for the JPL Juno Project-data latency analysis

National Aeronautics and Space Administration

```
flows
sci_hs_source: flow source science_hs {Throughput=>18000 bitsps;};
telem_source: flow source telem;
end jade;

device implementation jade.juno
flows
sci_hs_source: flow source science_hs;
telem_source: flow source telem;
end jade.juno;

inst_telem: in event data port;
inst_science: in event data port {Source_Data_Size=> 541000000 bits;};
flows
science_sink: flow sink inst_science;
```

## Description

Infos (2 items)

In end-to-end-flow jade\_hs\_end\_to\_end\_a and component nvmcard.juno, feature inst\_science will fill in 8 H 20 M 55 S

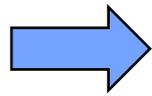


# Software Architecture Modeling and Assurance with AADL

## for the JPL Juno Project-data latency analysis

---

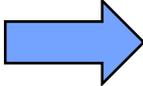
- Decision point on changing the data production rate during JADE high voltage checkout:



The data latency reliability plugin for OSATE could have been run in real time and it would have revealed the data overflow that was going to happen 8hr 20min 55sec later (before the next downlink could occur)

### Beginning and end of track for day 322:

- |       |           |          |
|-------|-----------|----------|
| • DOY | BOT (UTC) | EOT(UTC) |
| • 322 | 17:30     | 04:20    |

- **JADE commanding error could have been avoided**  
 **preventing loss of science return**



# Papers 2012

---

- ***“Assuring Software Fault Management with the Architecture Analysis and Design Language,”*** Kenneth Evensen, and Dr. Michela Muñoz Fernández. Infotech@Aerospace Conference 2012.

Objective:

-To demonstrate a model based framework targeted at assuring software fault management components

- **Four Principles**

- Architecture Analysis and Design Language (AADL) is a formal, declarative modeling notation
- AADL Error Annex is a formal notation for modeling dependability
- Building models of software systems helps conceptualize complex mechanisms (e.g. fault management)
- Assurance is an independent activity with the objective of increasing confidence in a system



# Papers 2012-What is Fault Management Assurance?

---

National  
Aeronautics and  
Space  
Administration

- Gaining increased confidence that a system can and does handle propagated faults and repairs
- Identifying potentially risky areas of the system and communicating that to the project
- We can automated tools (developed at JPL) in OSATE to assure a system can behave reliably



# Papers 2012-What Can we address with coverage?

---

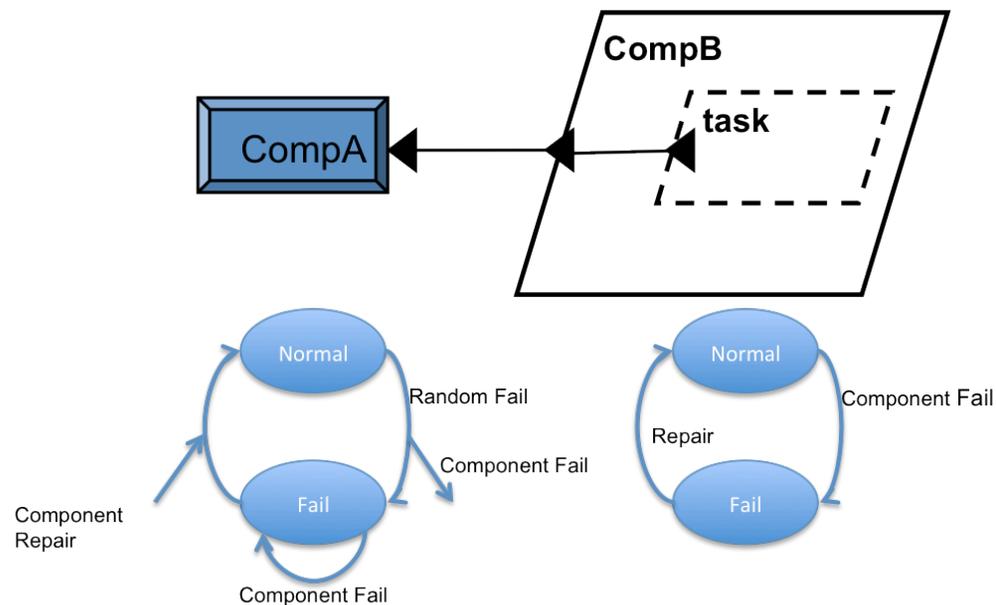
- **What percentage of possible propagations, occurring anywhere in the system, could affect the software?**
- **What percentage of actual propagations, occurring anywhere in the system, affect the software?**
- **What percentage of actual propagations, occurring anywhere in the system, are handled by the software?**
- **What percentage of actual propagations originating in software, are handled by hardware**



# Papers 2012-Missing a repair

- **Software Centric**

- Hardware “expects” an in propagation
- Expected that software propagates out a repair

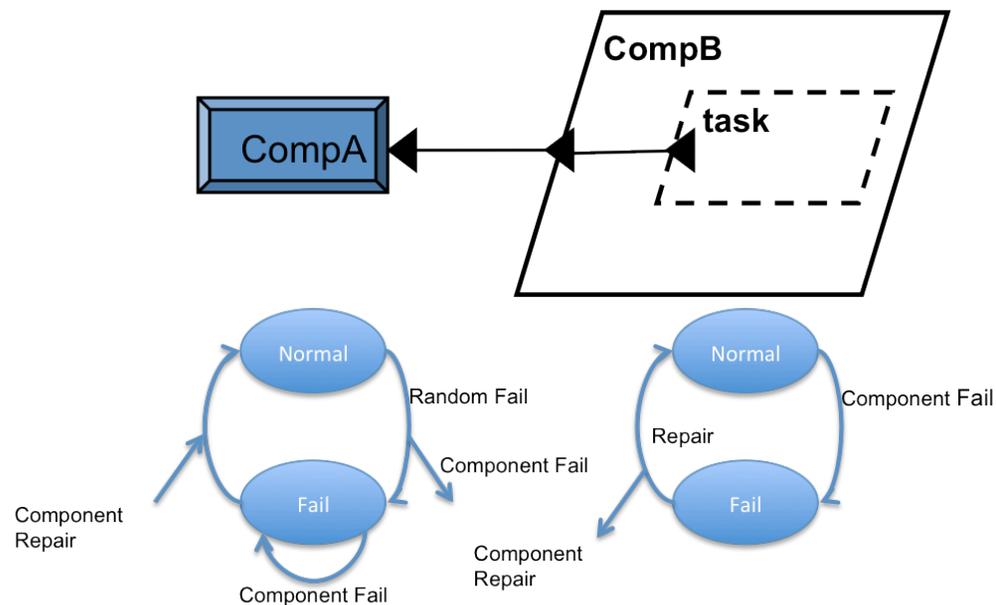


**i** Percent of actual propagations expected by hardware, not addressed by software 100% - Count ( 1 ) -



# Papers 2012- Repair added

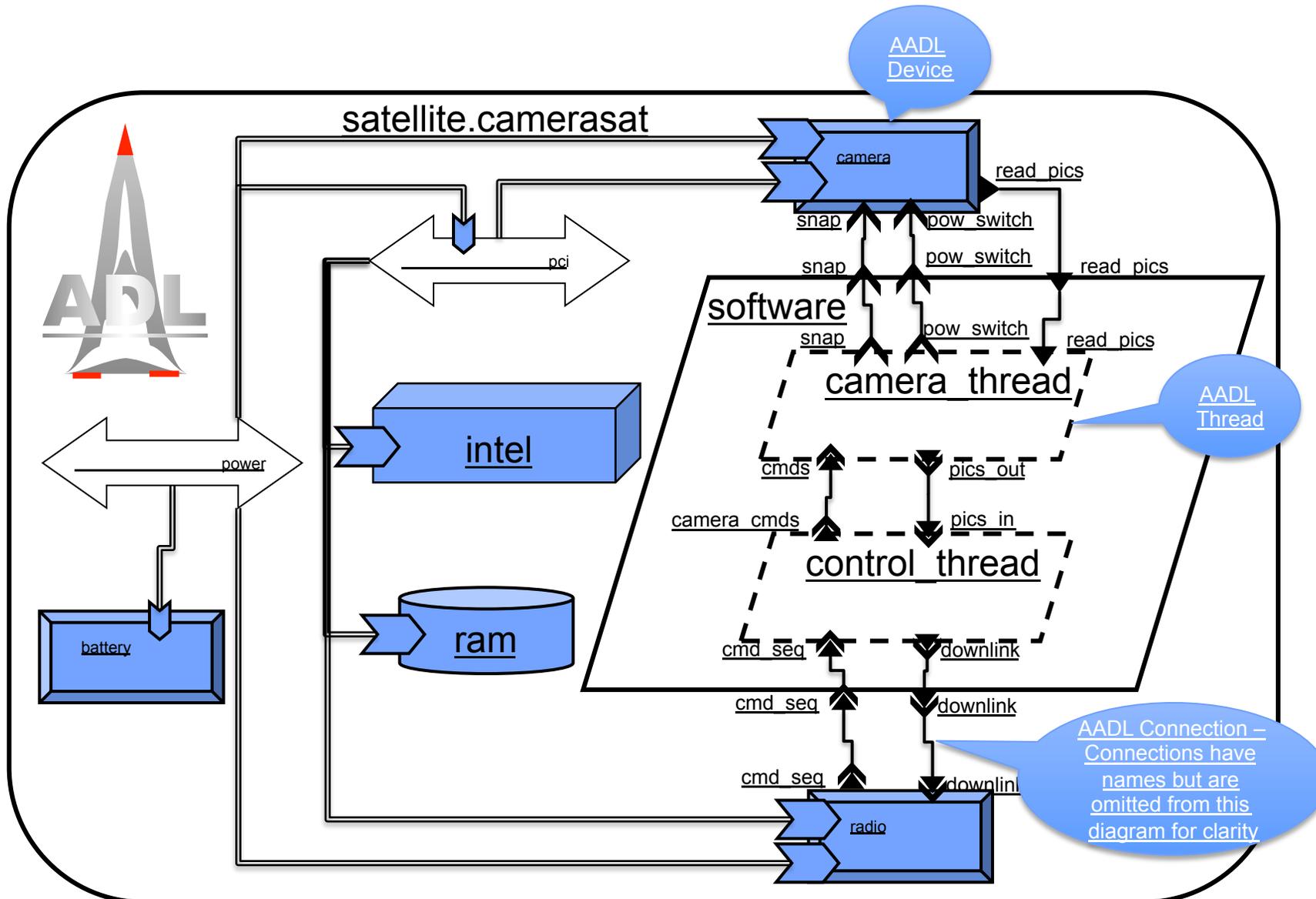
- AADL Error Model of “task” was missing “Component Repair”



**i** Percent of actual propagations expected by hardware, not addressed by software 00% - Count ( 0 ) -



# Papers 2012- Example – Satellite with a camera





## Papers 2012-CAMERA EXAMPLE

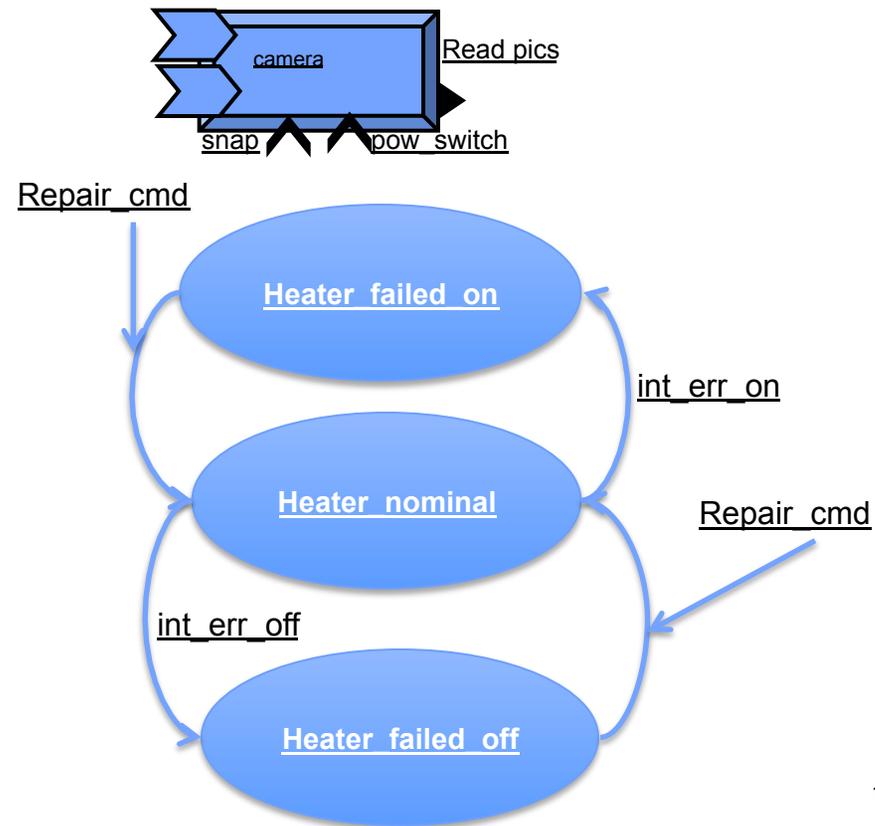
---

- The camera has a heater to regulate its temperature.
- Heater initial state: nominally off
- 1<sup>st</sup> failure mode: An internal event causes the heater to be turned on inadvertently.
- 2<sup>nd</sup> failure mode: The heater is nominally on and it is erroneously turned off.
- Higher instrument temperatures above Allowable Flight Temperature (AFT) limits will increase the thermal noise in the detector of the camera, lowering the signal to noise ratio, and therefore leading to degraded science return.



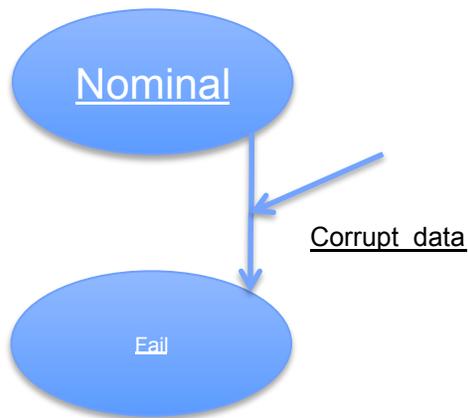
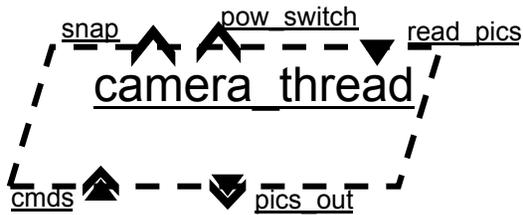
# Papers 2012- Camera – error model

- “camera” has three error states:
  - “heater\_failed\_on”
  - “heater\_nominal”
  - “heater\_failed\_off”
- Internal error causes transition to off-nominal state
- In propagation causes transition to nominal state





# Papers 2012- Camera thread-ERROR MODEL



- “camera” has two error states:
  - “nominal”
  - “fail”
- External error causes transition to fail state



# Papers 2012-ANALYSIS RESULTS I

---

- **None of the two propagations that the hardware expected to originate from software were addressed by the software**

**i** Percent of actual propagations expected by hardware, not addressed by software 100% - Count ( 2 ) -

- **The output above illustrates that the appropriate propagations are missing**
- **It does not show exactly what it is missing**
  - More detailed results are necessary for sufficient analysis



## Papers 2012- ANALYSIS RESULTS II

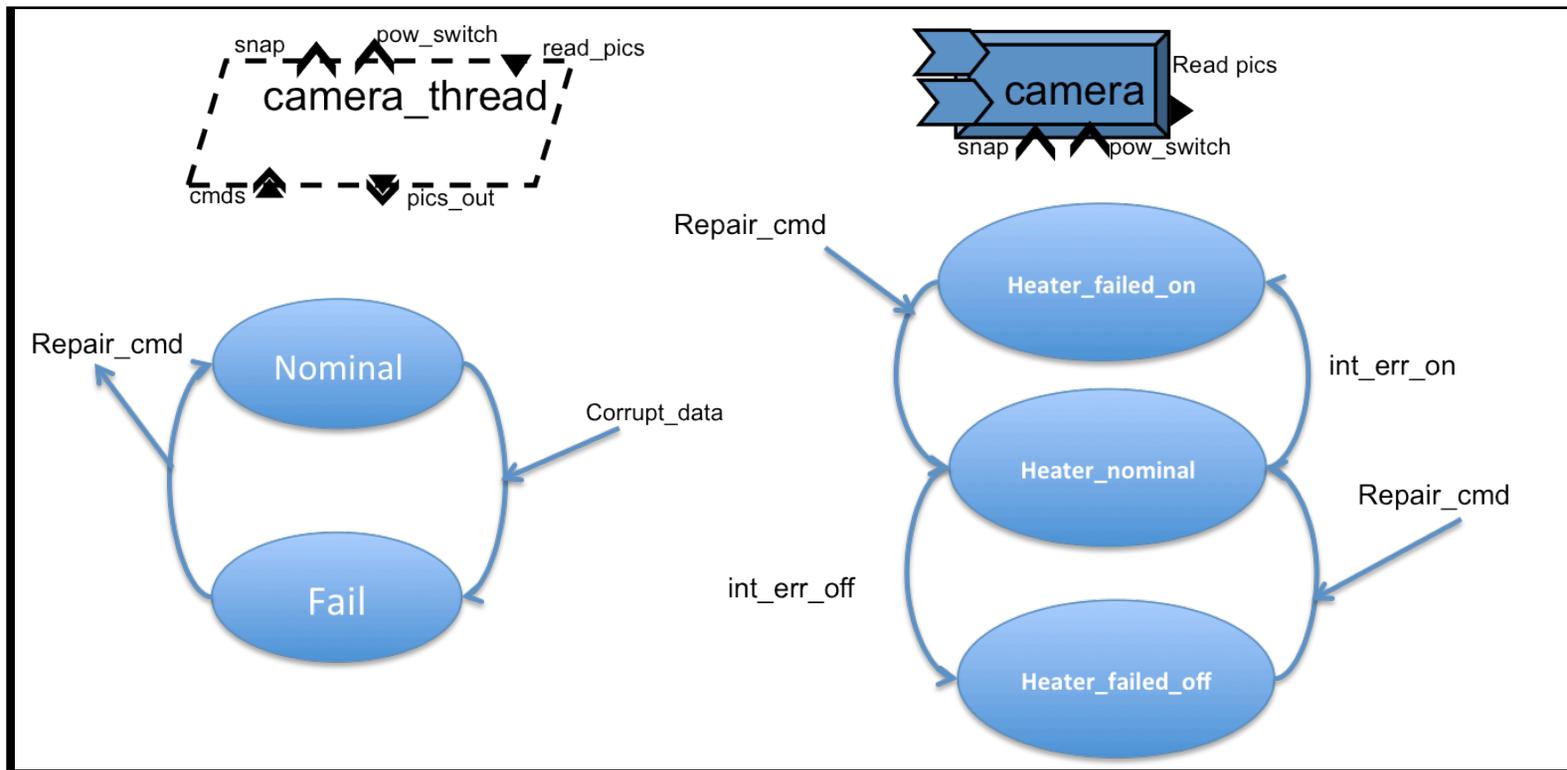
- The table below shows the detailed results of executing the analysis

Source	Rule	Destination	Propagation
software.camera_thread (thread)	D11	camera (device)	repair_cmd (Missing Out)
[...]	[...]	[...]	[...]

- It lists the binary relationship between the source, the “camera\_thread,” and the destination, “camera”
- The “Propagation” column indicates that the missing propagation is a “repair\_cmd” out propagation
- The AADL error model for the “camera\_thread” is missing an out propagation



- The discrepancy is corrected by adding an out propagation named “repair\_cmd” to the AADL error model in the “camera\_thread”





- The “repair\_cmd” out propagation is added to the transition between the “Fail” and “Nominal” states
  - This satisfies both the cases in the “camera’s” AADL error model, between the “Heater\_failed\_on” to “Heater\_nominal” transition and the “Heater\_failed\_off” and “Heater\_nominal” transition
- Executing the “Fault Coverage” tool a final time produces the desired result. The percent of actual propagations expected by hardware, not addressed by software becomes 0%

**i** Percent of actual propagations expected by hardware, not addressed by software 00% - Count ( 0 ) -



# Papers 2012-Summary

---

- **Reduced ambiguity in fault modeling because component definitions are formal**
  - AADL Provides foundation for the formal modeling of an real-time, avionics system
  - AADL Error Annex provides formal mechanics for modeling fault behavior
- **Assuring fault coverage increases confidence in the software fault management's ability to detect symptoms**
  - The software can recognize a fault originating from hardware
  - The software can address repairs expected by hardware (as seen in the example)



# Plug-in to convert UML to AADL using MARTE profile

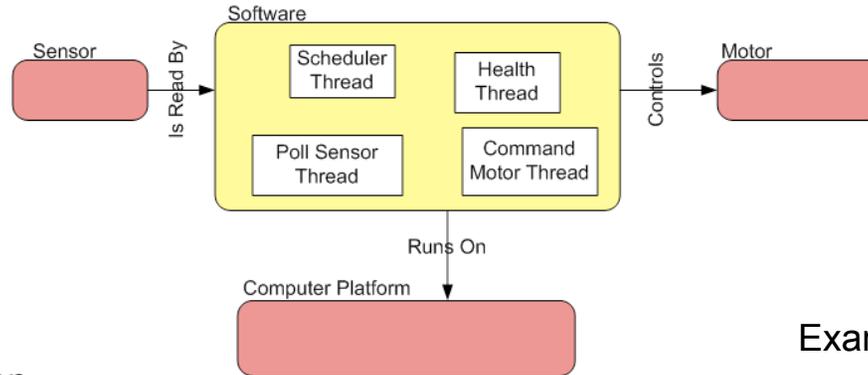
---

- MARTE (Modeling and Analysis of Real Time Embedded Systems) is a UML profile designed to handle real-time embedded systems and software concepts. UML was simply not designed to address concerns such as scheduling, performance, and time.
- MARTE's underlying meta-model, based on AADL's meta-model, provides the capability to address these concerns using UML notation. Various components from SysML (Systems Modeling Language), such as blocks and ports, are carried over into MARTE. MARTE does not introduce any new diagrams, as part of its meta-model.

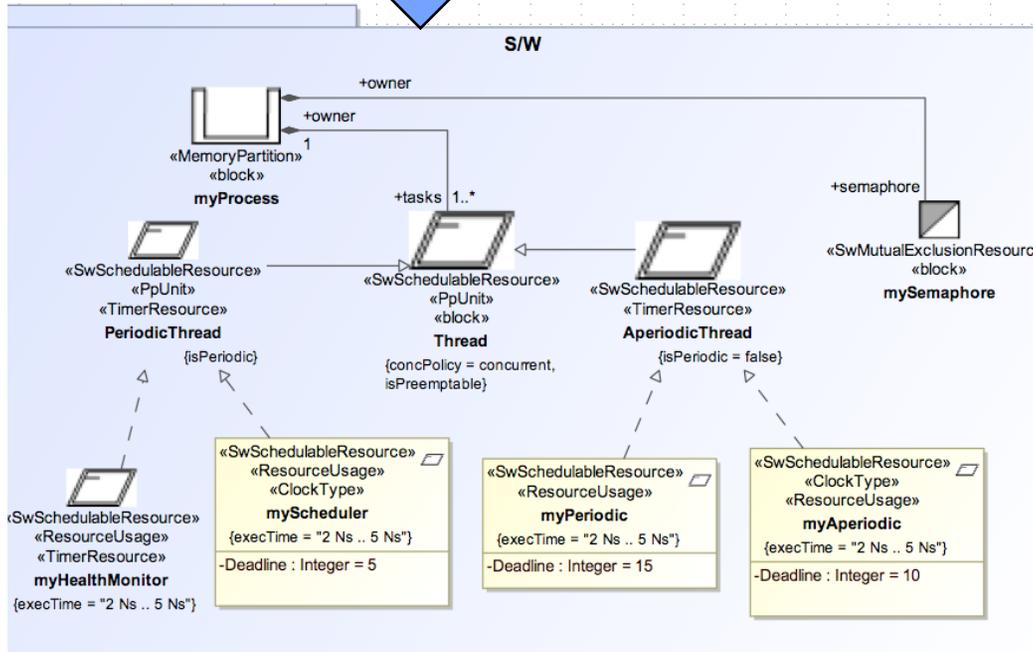
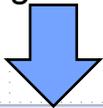


# Plug-in to convert UML to AADL using MARTE profile

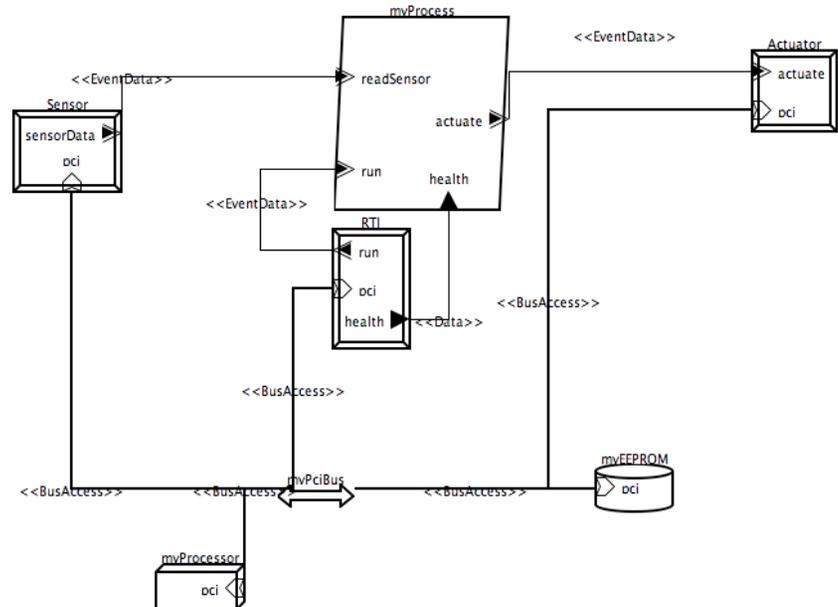
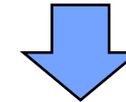
Basic context diagram for a software system



MARTE Software Organization



Example in AADL





# Future plans and direction

---

- **Continue working on SMAP models through launch**
  - Provide analysis updates at appropriate milestone reviews
  - Test analysis results against reality using testbed
- **Continue working on Juno models**
  - Complete Juno models
  - Apply AADL Error Annex for reliability analyses
  - Model the complete end to end data flow including the downlink process
  - Incorporate Juno FMECAs into the AADL model
  - The current tool only models deterministic events. Propose to explore the possibility to model non-deterministic scenarios (data generation, downlink)
  - Work with Leila Meshkat on integrating results from Command Assurance task to build complete reliability model
- **Conversion of Grace follow-on UML models into AADL**



# References

---

- “Assuring Software Fault Management with the Architecture Analysis and Design Language,” Kenneth Evensen, and Dr. Michela Muñoz Fernández. Infotech@Aerospace Conference 2012.
- “Software Assurance Standard,” NASA-STD-8739.8 w/Change 1. 2004.
- Barbacci, Mario, Mark H. Klein, et al. “Quality Attributes.” CMU/SEI-95-TN-021. 1995.
- Vestal, Steve, Larry Stickler, Dennis Foo Kune, Pam Binns, and Nitin Lamba. *AADL - Documents*. Honeywell, 16 June 2004. Web. <[www.aadl.info/aadl/documents/AADL-MetaH%20for%20LAS.pdf](http://www.aadl.info/aadl/documents/AADL-MetaH%20for%20LAS.pdf)>.
- Feiler, Peter H., David P. Gluch, and John J. Hudak. *The Architecture Analysis & Design Language (AADL): An Introduction*. CMU/SEI-2006-TN-011. 2006.
- OSATE: Open Source AADL Tool Environment. <http://www.aadl.info>. October 2009.
- SAE Embedded Computing Systems Committee. SAE Architecture Analysis and Design Language (AADL) Annex Volume 1: Annex A: Graphical AADL Notation, Annex C: AADL Meta-Model and Interchange Formats, Annex D: Language Compliance and Application Program Interface Annex E: Error Model Annex. As-2c Architecture Analysis And Design Lanaguage. 2006.
- Evensen, Kenneth D. and Dr. Kathryn Anne Weiss. “A Comparison and Evaluation of Real-Time Software Systems Modeling Languages.” *AIAA Infotech@Aerospace 2010*. Atlanta, GA, Apr. 20-22, 2010.
- Feiler, Peter H., and Ana Rugina. “Dependability Modeling with the Architecture Analysis & Design Language (AADL).” CMU/SEI-2007-TN-43. 2007.
- Heimerdinger, Walter L., and Charles B. Weinstock. “A Conceptual Framework for System Fault Tolerance.” CMU/SEI-92-TN-033. 1992.



# Background slides



# Fault propagation

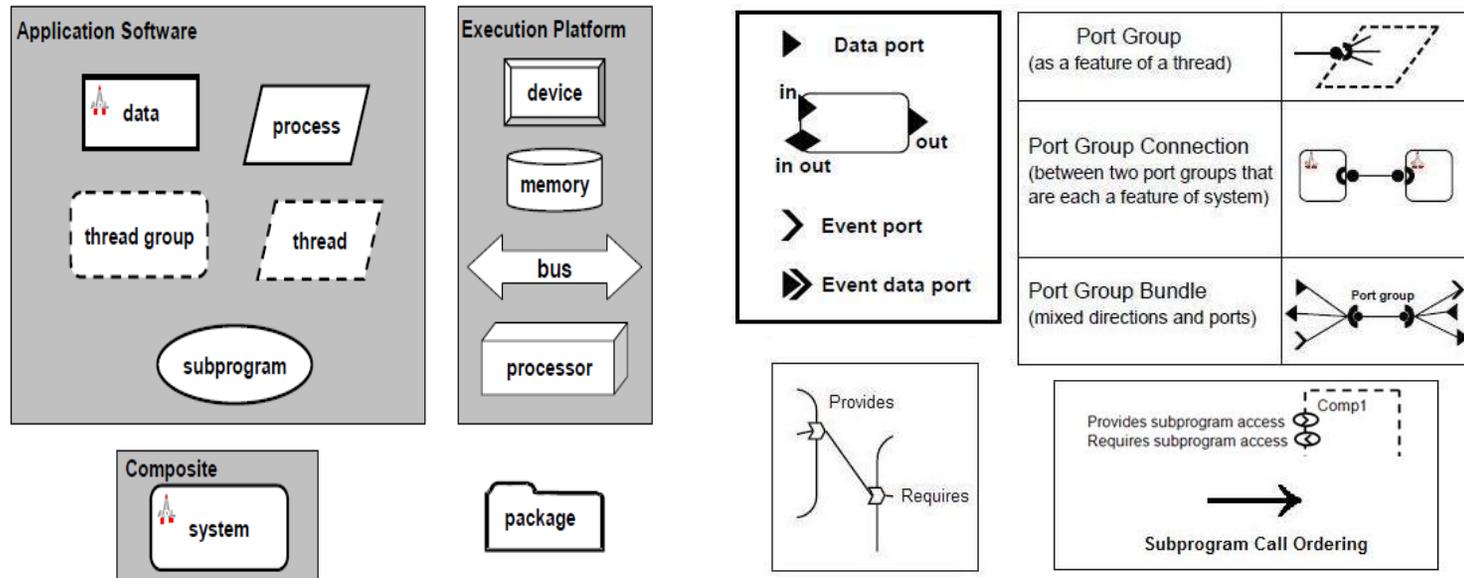
---

- **Fault propagation follows very specific rules per SEI TN 2007-TN-043**
  - Dependability Modeling with the Architecture Analysis & Design Language (AADL)
- **This provides a foundation for assessing susceptibility of error propagation without building a single error model!!!**
- **NOTE: Error propagation refers to both errors and repair propagations**
- **Error states refers to both good and bad states.**



# Using AADL

- Using AADL and OSATE
  - OSATE → Open Source AADL Tool Environment



For more information on AADL and OSATE, please visit: <http://www.aadl.info>



# Software Architecture Modeling and Assurance with AADL for the JPL SMAP Project

---

National  
Aeronautics and  
Space  
Administration

- **Refined SMAP FSW to model current planned implementation**
- **Built tool suite for OSATE to analyze various figures of merit related to reliability**
  - Coverage of errors that FSW handles
  - Time to criticality
- **Worked with Martin Feather to come up with some visualization ideas for fault coverage**
- **Worked with Allen on how to measure software reliability data from MSL testbed**



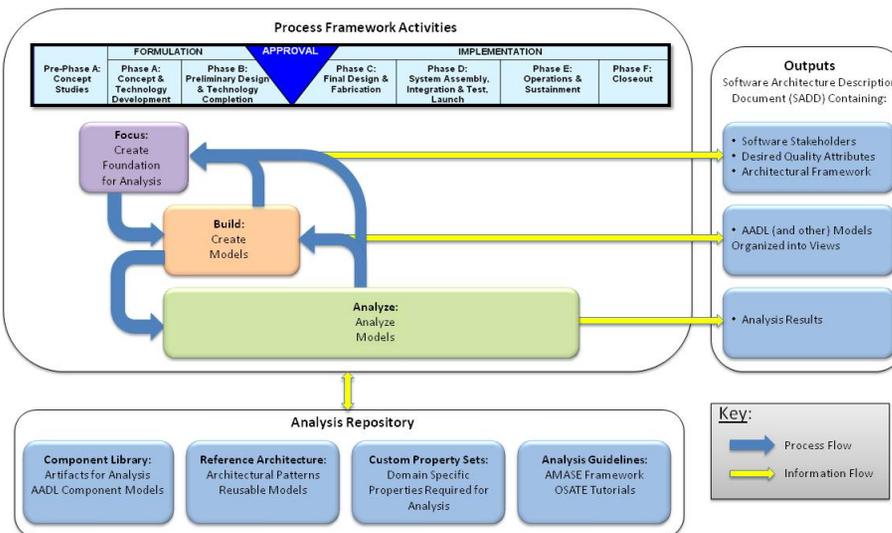
- **Begin to integrate SMAP hardware FMECA's into AADL Model**
  - Document procedure for mapping JPL FMECA's to AADL Error Annex
  - Getting some 3x help
- **Analyze the MSL testbed and ATLO data in order to estimate inherited reliability**
- **Bring Myron Hecht to JPL from Aerospace Corporation**
  - Has demonstrated tool chain for assessing dependability in AADL models
- **Continue to meet with Lorraine Fesq and John Day on software fault management**



# Overview of Approach

National  
Aeronautics and  
Space

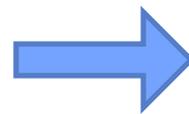
Applied AADL Practice Framework in the context of an architecture-centric development process



Described an architecture framework for documenting FSW

**AMASE → Architectural Modeling for Aerospace Software Engineering**

08/21/2012



**AMASE: Architectural Modeling for Aerospace Software Engineering**

---

**Conceptual Level of Abstraction**

**Conceptual Viewpoint**  
**Description; Rationale:** A conceptual architecture description, or conceptual view of an architecture, is an abstract description often used during requirements analysis and specification. Views constructed using this guideline are meant to explain the architecture's context, and how and why function is mapped to form through concept.  
**Stakeholders:** Systems Engineers, Project Management  
**Concerns:** Driving Functional and Quality Attribute Requirements, Compliance with Systems Architecture, Conceptual Integrity  
**Language:** Conceptual Architecture, Form, Function, Concept, Value Delivery  
**Modeling Techniques:** Context Diagrams, Box and Line Diagrams, Functional Decomposition, Architectural Principles and Constraints, Requirements Mapping  
**Analytic Methods:** Requirements Audit, Peer Reviews, Qualitative Evaluation  
**Source:** AMASE framework - Weiss '09.

---

**Realizational Level of Abstraction**

**Structural Viewpoint**  
**Description; Rationale:** Views constructed using this viewpoint describe (1) how the software is decomposed into component and interface types and (2) the rules that govern how component and interface types can be legally organized into collaborating structures. These views will contain descriptions of any patterns of structural organization that exist in the software that enforce the conceptual architecture principles and constraints.  
**Stakeholders:** FSW Developers, FSW Management  
**Concerns:** Architectural Principles and Constraints: Refined from Conceptual Architecture, Rense, Layering, Dependencies  
**Language:** Component Types, Interface Types, Ports, Roles, Patterns, Artifacts, Libraries  
**Modeling Techniques:** AADL → Systems and Port Groups  
**Analytic Methods:** AADL → Semantic Checking and Enforcement  
**Source:** Earliest reference in Perry and Wolf, Shaw and Garlan. This particular definition is part of the AMASE framework - Weiss '09.

**Information and Control Flows Viewpoint**  
**Description; Rationale:** Information and Control Flows are the key value delivery mechanisms for aerospace software, and therefore form the basis of the unitifying viewpoint. Views constructed using this viewpoint describe how high-level, cross-cutting requirements that involve information and control flow are satisfied by the software. In other words, these views characterize the functionality of the software system as a whole.  
**Stakeholders:** Systems Engineers, End-to-End Information (includes Flight-Ground Interface) Systems Engineer, FSW and System Testers, Operations Engineers  
**Concerns:** Value Delivery via Information Flow and Control Flow, Requirements Satisfaction  
**Language:** Information Flow (sources and sinks), Control Flow, Transfer of Control, Locus of Control  
**Modeling Techniques:** AADL → Flow Paths, End-to-End Flows, State Machines / State Charts  
**Analytic Methods:** AADL → Flow Analyses such as Latency and Jitter Simulation  
**Source:** AMASE framework - Weiss '09.

---

**Dynamic Viewpoint**  
**Description; Rationale:** Views constructed using this viewpoint describe the run-time architecture of the software, i.e. the components that exist at run-time such as threads and sub-programs, and the connections that enable their interactions while the software is executing. These views will contain the behavior, or logic, of the individual software components, as well as how several components behave in concert to accomplish higher-level behavior. These views also expose any operational modes of the software and how the software transitions between those modes.  
**Stakeholders:** FSW Developers  
**Concerns:** Synchronization, Interaction, Behavior, Deadlock, Starvation, Livelock, Execution Times  
**Language:** Run-Time Architecture, Components, Connectors, Processes, Threads, Behavior, States, Modes  
**Modeling Techniques:** AADL → Processes, Threads, Sub-Programs, Ports, Connectors, Configuration Modes, UML → Activity Diagrams, Sequence Diagrams, State Machines / State Charts  
**Analytic Methods:** AADL → Execution Time / Deadline Load Analyzer, AADL Behavior Annex → Simulation, Priority Inversion Checker, Model Checking  
**Source:** Krutchen 4+1 View Model. This particular definition is part of the AMASE framework - Weiss '09.

**Avionics Viewpoint**  
**Description; Rationale:** Views constructed using this viewpoint illustrate the embedded nature of aerospace software - they describe the mapping between the software and the underlying avionics, or flight hardware. Views that follow this guideline also address firmware, or functional allocation to FPGAs.  
**Stakeholders:** Avionics Engineers, Systems Engineers, FSW and System Testers  
**Concerns:** Resource (memory, bus bandwidth, etc.) Consumption, FPGA, Device Drivers, Low-Level Interfaces  
**Modeling Techniques:** AADL → Buses, Processors, Memory, Devices, Binding Properties, UML → Deployment Diagram  
**Analytic Methods:** AADL → Bandwidth Load Analysis, CPU Load Analyzer, Scheduling Analysis, Memory Utilization  
**Source:** AMASE framework - Weiss '09.

---

**Cross-Cutting Concerns**

**Implementation Viewpoint**  
**Description; Rationale:** Views constructed using this viewpoint illustrate the iterative and incremental nature of software development, i.e. they describe (1) how functionality is added to modules and (2) how modules are integrated into the larger software system, both as a function of time. These views also address how software modules are assigned to developers for implementation. Views that follow this guideline are important, because they address a key portion of software architecture - its evolution.  
**Stakeholders:** FSW Developers, FSW and Project Management, FSW and System Testers  
**Concerns:** Development Planning, Available Functionality, Architecture Evolution  
**Language:** Assignments, Functionality, Modules, Workforce Planning  
**Modeling Techniques:** AADL → Packages, Gantt Charts / PERT Charts, SysML → Activity Diagrams  
**Analytic Methods:** Managerial Peer Reviews, Cost and Risk Estimation and Characterization  
**Source:** AMASE framework - Weiss '09.