Using AADL to Enable MBSE for NASA Space Mission Operations

Dr. Michela Muñoz Fernández
NASA Jet Propulsion Laboratory, California Institute of Technology

May 6, 2014

© 2014 California Institute of Technology. Government sponsorship acknowledged.
Published by The Aerospace Corporation with permission.

MBSE for Space Systems

- Use of Model-Based Systems Engineering (MBSE) to enable more robust and complete systems engineering and integrated analysis of complex System-of-Systems (SoS) problems which have historically been implemented via paper/ presentation-based design capture, disparate models, in documents, and in the brains of expert engineers across many disciplines
- Can new tools and technologies be used in future missions starting at earlier phases to reduce risk?

The Architecture Analysis & Design Language

- The SAE Architecture Analysis & Design Language (AADL) is an architecture description language for real-time, fault-tolerant, scalable, embedded, modular multiprocessor systems.

- AADL enables the development of highly evolvable systems, early and quantitative analyses of a system's architecture, and evolution of an architecture model for continued analysis throughout the lifecycle.

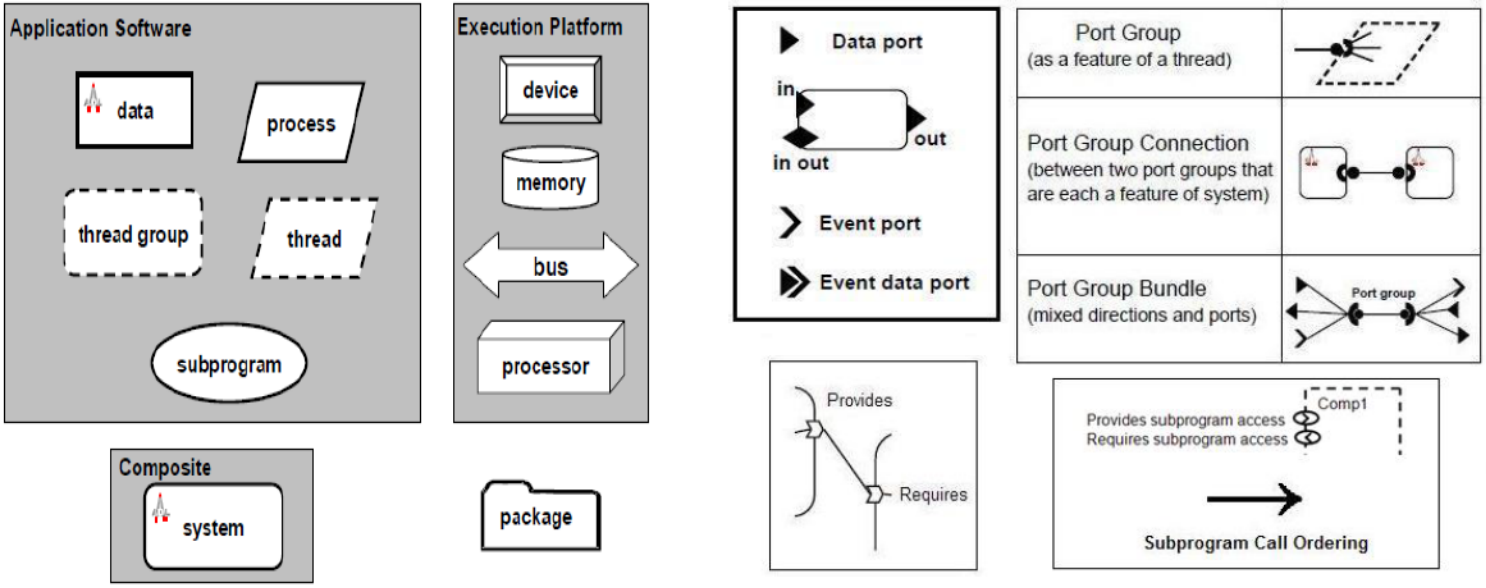
- Customers: System architects that would like to optimize the decision on system architectures and/or any engineer in general that would like to model embedded systems

The Architecture Analysis & Design Language

- AADL's capabilities:
  - Create and analyze component-based models of a task and task interaction architectures of embedded software
  - Predictive analyses of operational characteristics (meeting deadline, response time, and throughput requirements)
  - Discover system integration problems early in a development effort

Using AADL

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

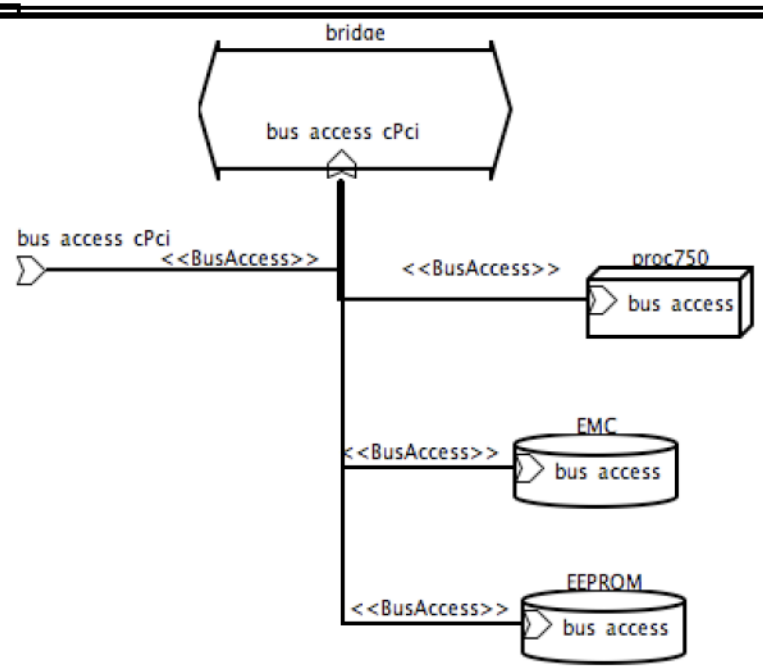


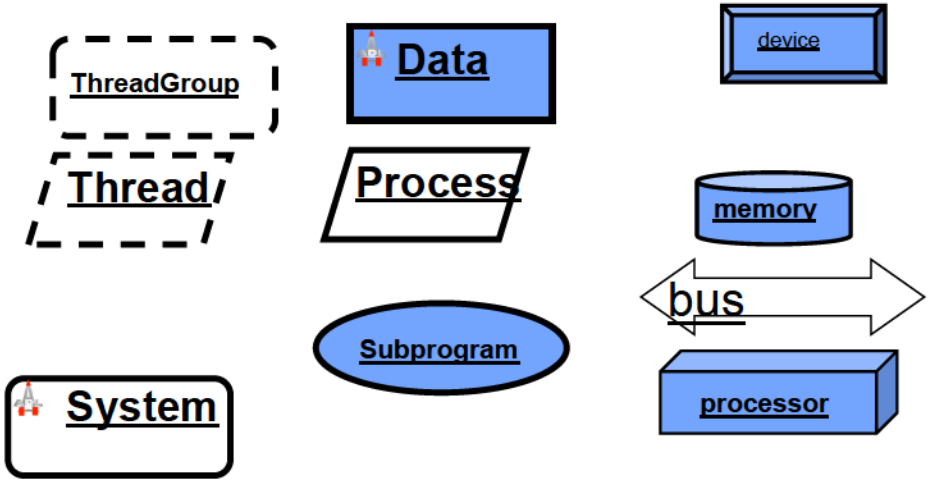
Source: http://www.aadl.info

5

AADL Textual and graphical representation

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





MBSE for Space Systems

- **Space systems software has been developed without characterizing *performance* of the real-time system being built until integration**
- **Finding execution-related issues at that point is costly**

- **AADL** (Architecture Analysis and 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

AADL SysML Comparison

- AADL was born as an avionics-focused domain-specific language and later on was revised to represent and support a more general category of embedded real-time systems
- SysML is an extension of the Unified Modeling Language (UML) intended to support modeling system engineering applications
- SysML focuses on the “big picture” architectural views, whereas AADL addresses the more detailed platform-oriented and physical aspects of such systems

AADL SysML Comparison

- Mutually complementary:
- SysML: standarized language for systems engineering. Provides support for requirements engineering, traceability, and precise modeling of diverse physical phenomena.
- AADL: oriented towards the modeling of real-time embedded systems and includes a comprehensive catalogue of hardware and software elements common in such systems and their characteristics, allowing relatively precise and dependable analysis of different system properties such as performance, timing, or power consumption

AADL SysML Comparison

SysML

Requirements, Traceability, Parametric models, Interactions

AADL

Quantitative Analysis, Hardware-Software Component Categories, Software to Hardware Binding

SysML and AADL

Modes/State Machines, Components/System Blocks, Component Interactions/Block Flows

AADL modeling applied to NASA space systems

- MBSE techniques applied to software quality assurance provide a rigorous framework for the verification and validation of software systems through the systematic modeling and analysis of formal architecture representations
- This type of framework has been applied to several JPL missions and systems: Mission Data System (MDS) reference architecture, Soil Moisture Active Passive (SMAP), and the Juno mission to Jupiter.
- These cases have been studied using AADL as a Model-Based Engineering language for architectural analysis and specification of real-time embedded systems with stringent performance requirements (e.g. fault-tolerance, security, safety-critical).

Examples of applications for NASA Systems

Unmanned test flight -Exploration Flight Test-1 or EFT-1. Image Credit: NASA.gov

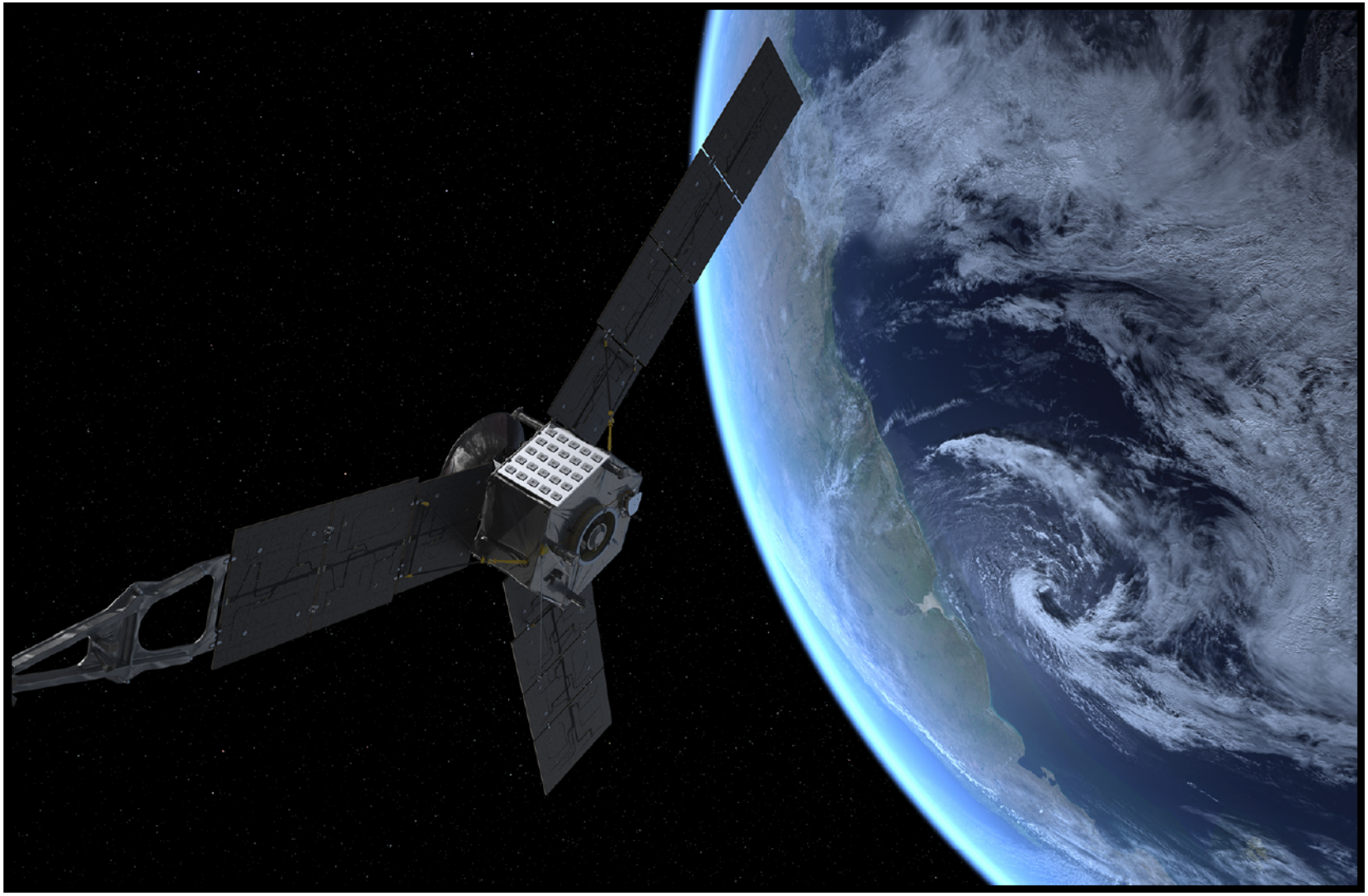
AADL's capabilities enable modeling of systems comprised of hardware and software subsystems connected to each other via hardline and RF communications links that support the exchange of critical data such as Commands (CMD), various forms of Telemetry (e.g., Operational, Developmental, Engineering), File Exchanges, Primary and Dissimilar Voice, Video/Motion Imagery, Time

The configuration of the systems, required data exchanges and communications links may change significantly between mission phases

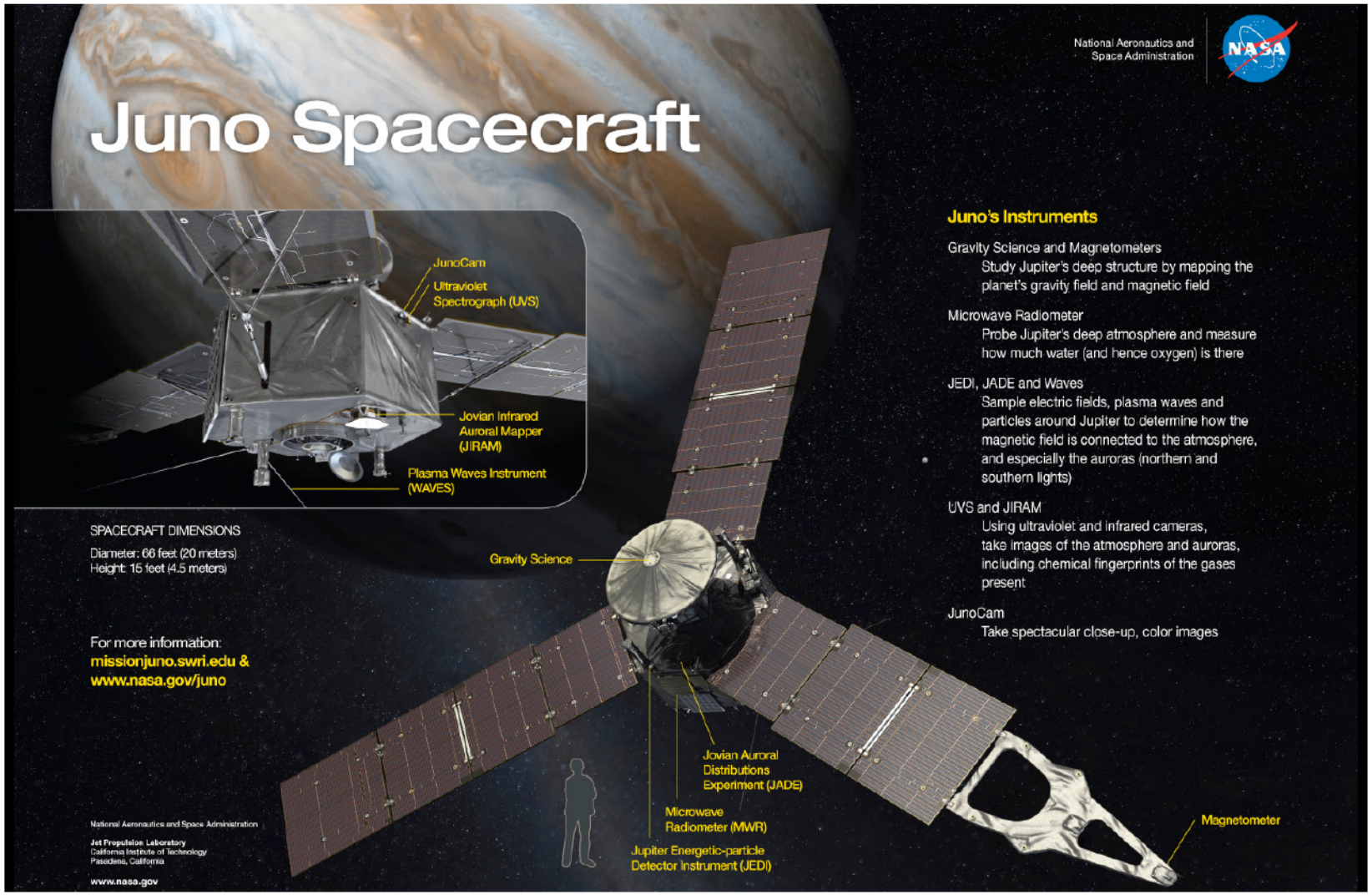
Case Study of an Application to a Flight Mission -The Juno Mission to Jupiter

- The Juno spacecraft launched aboard an Atlas V-551 rocket from Cape Canaveral, Fla., on Aug. 5, 2011, and will reach Jupiter in July 2016
- Juno uses a spinning solar-powered spacecraft in a highly elliptical polar orbit that avoids most of Jupiter's high radiation regions
- The designs of the individual instruments are straightforward and the mission does not require the development of any new technologies. Juno will improve our understanding of the solar system's beginnings by revealing the origin and evolution of Jupiter

Case Study of an Application to a Flight Mission -The Juno Mission to Jupiter



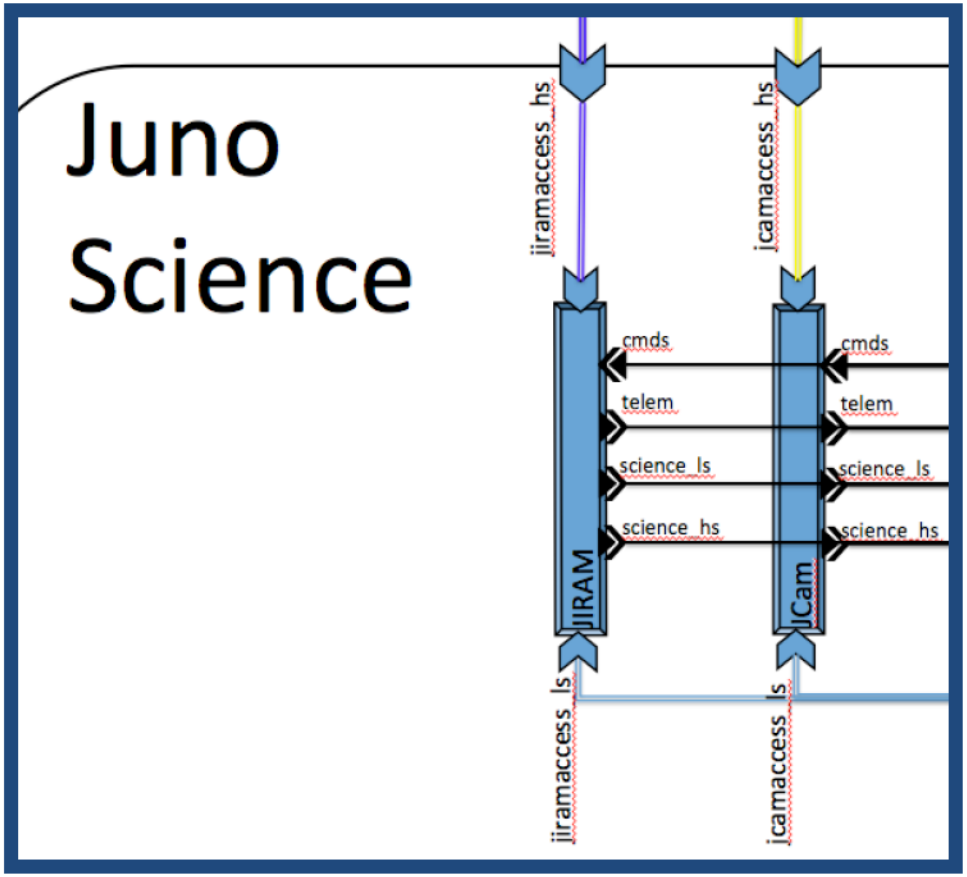
Case Study of an Application to a Flight Mission -The Juno Mission to Jupiter



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
  - End to end data flow: data latency analysis-> revealed scenarios where command 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



The model was developed after the initial Juno instrument checkouts.

During some of the instrument checkouts there were command errors.

By modeling the Juno spacecraft, and applying new tools, errors would have been revealed in real time as it was demonstrated by performing AADL modeling

The figure captures part of the Juno science system showing two of the instruments and their connections:

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

```
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

Info: In end-to-end-flow jade_hs_end_to_end_a and component card.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**

Juno modeling benefits and plans

- Future work in his area includes refining the Juno model and providing it to the instrument teams (IOTs) with a GUI in order for them to run it with different scenarios before they plan to make a change in a sequence that for example would change the data rate or any other parameter of relevance to the specific science mode used
- The AADL model would be a tool that would allow the principal investigators and engineers have an additional way to ensure that the instruments will be safe as well as help prevent any loss of science data once Juno reaches Jupiter in 2016
-

AADL Error Annex

AADL has been extended to model fault management behavior through the AADL Error Annex, also an SAE standard

The AADL Error Annex can be used to assure dependability in the software fault management system in avionics real-time embedded systems

It enables modeling of different types of faults, fault behavior of individual system components, and fault propagation affecting related components in terms of peer-to-peer interactions and deployment relationships between software components and their execution platform

Error propagation with AADL

- Errors can propagate between software components and execution platform components they are bound to.
  - The keywords processor, bus, virtual processor, virtual bus, memory, and device are used to identify the binding point of a software component with the execution platform component it is bound to
  - The keyword binding is used for connections and virtual buses to identify their binding to execution platform components.
  - The keyword bindings is used in execution platform components to identify the binding point of components bound to them. Propagations with respect to bindings can be in both directions.

Error propagation with AADL

- It also allows modeling of aggregation of fault behavior and propagation in terms of the component hierarchy, as well as specification of fault tolerance strategies expected in the actual system architecture
- It supports qualitative and quantitative assessments of system dependability, i.e., reliability, availability, integrity (safety, security), and survivability, as well as compliance of the system to the specified fault tolerance strategies from an annotated architecture model of the embedded software, computer platform, and physical system

Error propagation with AADL

- The "Fault Coverage" analysis can help uncover any missing propagation:
  - It can be determined if the software system is not handling the appropriate propagations.

The Error Annex was used to perform some analyses on the instance of the Juno model, the figure below lists a subset of the information provided in the detailed output generated by the "Fault Coverage"

Source	Rule	Destination	Propagation
Jno telecom.sdst.sdst_a (device)	D14	Jno bus1553 (bus)	corrupt_seq (Missing In)
Jno cdh_a.FSW.io.mmm_mgr (thread)	D16	Jno cdh_a.FSW.payload.jade_io (thread)	corrupt_cmd
Jno cdh_b.FSW.io.ms1553 (thread)	D16	Jno cdh_b.FSW.payload.junocam_cmd (thread)	corrupt_cmd
Jno cdh_b.FSW.io.ms1553 (thread)	D16	Jno cdh_b.FSW.payload.mwr_cmd (thread)	corrupt_cmd

Error propagation with AADL

- Table 1 lists a subset of the information provided in the detailed output generated by the “Fault Coverage” tool. There are four total “out propagations” occurring in the system, taking into account the binary relationship defined by the AADL Error Annex dependency rules.
- The “Propagation” column in Table 1 lists all the “out propagations” by name, found in the error model. The propagations in red indicate that they are not handled by the destination. The first row shows the only binary pair in this listing where the propagation is unhandled. With the issue uncovered, a solution can be implemented. A mechanism is required to handle the incoming propagation “corrupt_seq.” Simply, an “in propagation” of the same name must be declared in the appropriate error model and applied to a transition.
- Executing the “Fault Coverage” tool a final time produces the desired result. The percentage of actual propagations unhandled by the destination becomes 0%.

Conclusions

- The paper has shown how AADL can be applied to space missions.
- A case study described the Juno mission to Jupiter:
  - Some of the analyses that were performed for the Juno mission included end-to-end data flow and data latency that revealed where command errors can occur.
  - Data generation and memory analysis revealed the scenario when data overflow would occur which could have prevented loss of science data. The particular value of these analyses to Juno was to model the science collection and data downlink rate.
  - Furthermore, analyses results show how some the Juno command errors might have been avoided if the AADL model had been in place before the Juno instruments checkout activities.
  - Analyses results show the potential that AADL has in order to model flight and ground systems architecture applied to space operations. This work could be extended to model missions such as Mars 2020 or Europa.

Conclusions

- The Architecture Analysis and Design Language (AADL) is a Society of Automotive Engineers (SAE) standard notation (AS5506/1) for the modeling and analysis of real-time systems. AADL has been extended to model fault management behavior through the AADL Error Annex, also an SAE standard

- The Architecture Analysis and Design Language and the AADL Error Annex can be used to assure dependability in the software fault management system in an avionics, real-time embedded systems

- AADL's applications to software assurance. The approach needs to be supported by a standard, repeatable framework. The foundation for this framework is in the ability to model the fault management system integrated in the hardware and software avionics real-time system. Models can assist in assuring both functional and quality attribute requirements such as reliability. NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP)

References

- AADL, http://www.aadl.info/aadl.currentsite/
- Evensen, Kenneth D., Michela Muñoz Fernández. "Assuring Software Fault Management with the Architecture Analysis and Design Language." *AIAA* Infotech@Aerospace. 19 - 21 June 2012. Garden Grove, CA.
- Model-Based Engineering with AADL, Peter H. Feiler, and David P. Gluch. SEI, Carnegie Mellon. 2013.
- Dave Gluch, Model-Based Software Assurance with the SAE Architecture Analysis & Design Language (AADL), September 2008.
- SysML and AADL, patterns for integrated use. J. Hugues, P. de Saqui-Sannes, ISAE.
- INCOSE Systems Engineering Vision 2020. INCOSE-TP-2004-004-02. September 2007.

References

- SysML, http://www.sysml.org
- OMG. http://www.omg.org
- NASA Ground Systems Development and Operations Program. http://go.nasa.gov/groundsystems
- [10]Ariane 5 Software problem. http://www.vuw.ac.nz/staff/stephen_marshall/SE/Failures/SE_Ariane.html
- [11]NASA Juno website: http://www.nasa.gov/mission_pages/juno/multimedia, Juno mission status updates: http://missionjuno.swri.edu/news

Questions?