

# Cal Poly Innovation Jam

---

Jet Propulsion Laboratory, California Institute of Technology  
March 30, 2012



**JPL Innovation Foundry**



# Aims, Goals, and Objectives

- Challenge Cal Poly science and engineering students to conceptualize future science measurements and technology needs for the coming decade
- Answer the technical question, can new FSW applications and methods be implemented / tested on CubeSats for later adoption by JPL
- Explore pathways and methods for open innovation / collaboration between JPL and university partners



# Concept Maturity Level Definitions

- CML 1 – “Cocktail Napkin”: Objectives and basic approach.
- CML 2 – Initial Feasibility: High-level physics, mass and cost assessments.
- CML 3 – Trade Space: Objectives and architecture trade space elaboration and evaluation of performance, cost and risks.
- CML 4 - Point Design within Trade Space: Subsystem-level design and cost estimates.
- CML 5 – Concept Baseline: Relationships and dependencies, partnering, heritage, technologies, key risks, mitigation plans and system make-buy approaches.
- CML 6 – Initial Design: Requirements and schedules to subsystem level, grassroots cost agreements, schedule, and V&V approach for key areas.

Taken from Pre-Project Principles and Practices, Rev. 0



# Innovation Jam Culture

- We are a team of peers – that includes YOU
- Foster trust and respect
  - Talk about crazy ideas not crazy people
  - What’s said in the sessions stays in the sessions
- Build on the ideas of others
- If you are in the room, you participate
- Make an effort to listen more than you speak
- Don’t ask for permission, plenty of forgiveness
- Encourage wild ideas, they inspire innovative ideas
- Discovery is key – learn something new today
- Have at least one “bad” idea today – and many more “good” ones!



# A Couple Quick Thoughts

- This is going to be like drinking from a fire hose, but remember this is just the start
- We are here to help and this is building a partnership – trust and respect are key
- You are not in competition with each other – at least for now! – so help each other
- You are building a network of JPLers who know more about your concept and can continue to be a good resource for you
- Use this time and this resource well



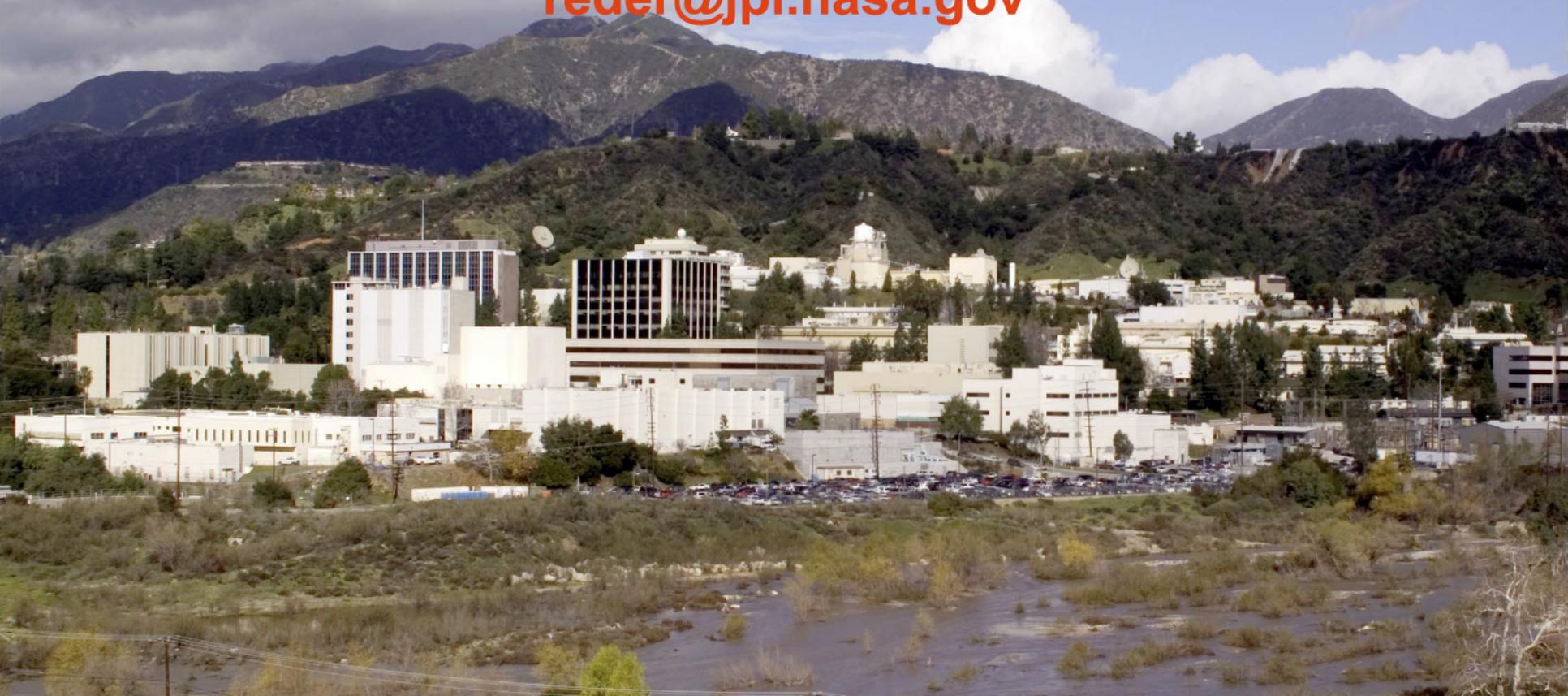
# Agenda

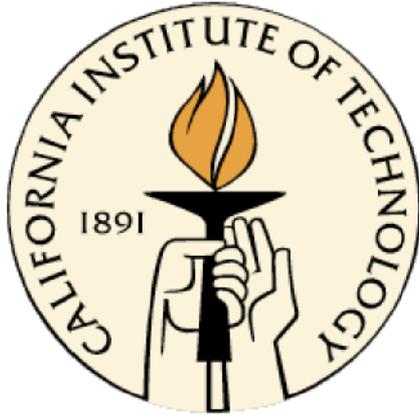
- Introductions and Overview 8:00 am
- Morning Presentations 8:20 am – 10:00am
  - Overview of JPL 8:20 am
  - Future Directions in Mission Computing 8:35 am
  - Flight Software Applications Group 8:55 am
  - Avionics at JPL Present and Future 9:25 am
- JPL Tour including SMAP C&DH testbed 10:00 am
- Lunch 12:00 pm
- Science and Technology Traceability Matrix 1:00 pm
  - Introduce the concept of “Science Value Matrix”
- Student Concept Presentations 1:30 pm
  - Each team will have 20 min for presentation and discussion
- Next Steps Toward Refining Concepts 5:00 pm

The JPL logo is rendered in a bold, red, sans-serif font. The letters are thick and blocky, with the 'J' and 'L' having a distinctive shape. The logo is centered at the top of the image, set against a background of a blue sky with scattered white clouds.

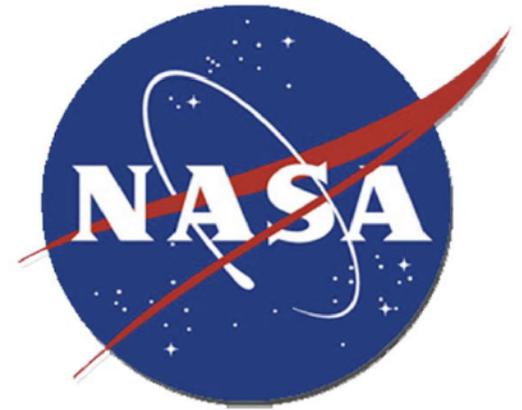
# JPL

[www.jpl.nasa.gov](http://www.jpl.nasa.gov)  
Leonard Reder  
[redler@jpl.nasa.gov](mailto:redler@jpl.nasa.gov)

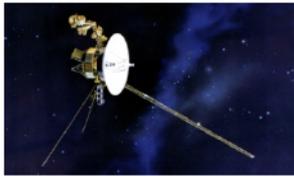




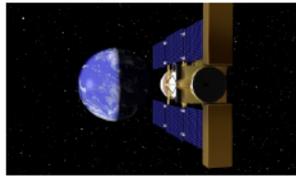
**JPL**



- **Spread out over about 176 acres**
- **Approximately \$1.6 B Business Base**
- **Over 5000 employees and contractors**
- **Under NASA contract but managed by CalTech**
- **Supported by proposals to do work**
- **About 15% of work is non-NASA (Military, Civil, etc.)**
- **JPL is a non-profit organization**
- **The lab has been around for about 75 years**



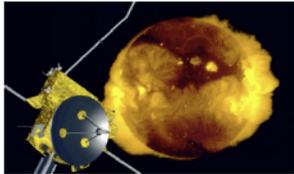
Voyagers 1 & 2



Stardust



Galaxy Explorer



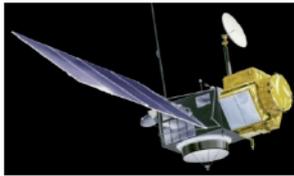
Ulysses



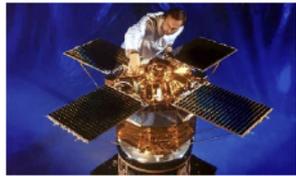
QuikScat



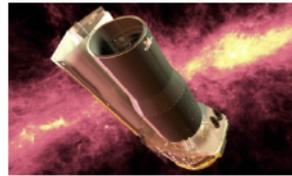
Spirit & Opportunity



Topex/Poseidon



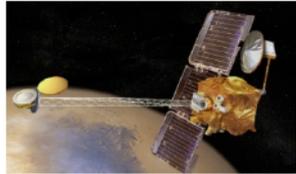
ACRIM



Spitzer Space Telescope



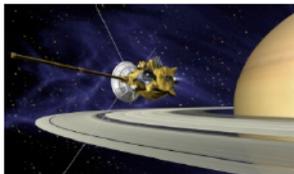
Mars Global Surveyor



Mars Odyssey



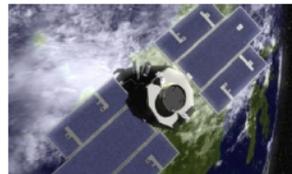
Mars Reconnaissance Orbiter



Cassini



Jason



CloudSat

On behalf of NASA, JPL is responsible for operating many spacecraft, 2 Mars rovers, and numerous science instruments that today are surveying the solar system and beyond.



# Mars Rovers



# Deep Space Network Antenna (70 meter diameter)

- Three locations
  - 120 deg. Apart
  - Goldstone, CA.
  - Madrid, Spain
  - Canberra, Australia
- 1966 support of Mariner 4
- Used to track Apollo and every major robotic planetary mission
- Other Science:
  - Interferometry



<http://deepspace.jpl.nasa.gov/dsn/>

# Keck Interferometer



- 2 Keck 10m telescopes with full adaptive optics
- Wavelengths 1.2  $\mu\text{m}$  to 10  $\mu\text{m}$   
(AO operates in visible)
- Science: nulling, differential phase, astrometry, imaging
- First fringes with Keck Telescopes March, 2001

# Interferometer Major Components



Beam Transport Optics  
(Coude Train)



Fringe Tracker (FATCAT)



Fast Delay Lines (FDL)



Angle Tracker (KAT)



Long Delay Lines (LDL)



# Smaller Projects



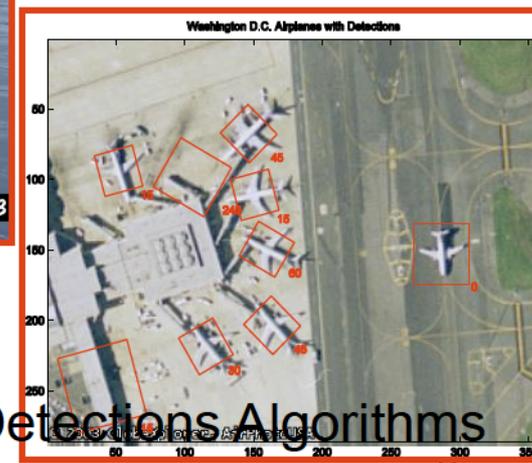
Lake Tahoe Buoy  
Satellite Sensor  
Calibration and  
Validation



Laser Spectrometer  
Absorption Instrument  
Ready for Flight out of  
Van Nuys Airport

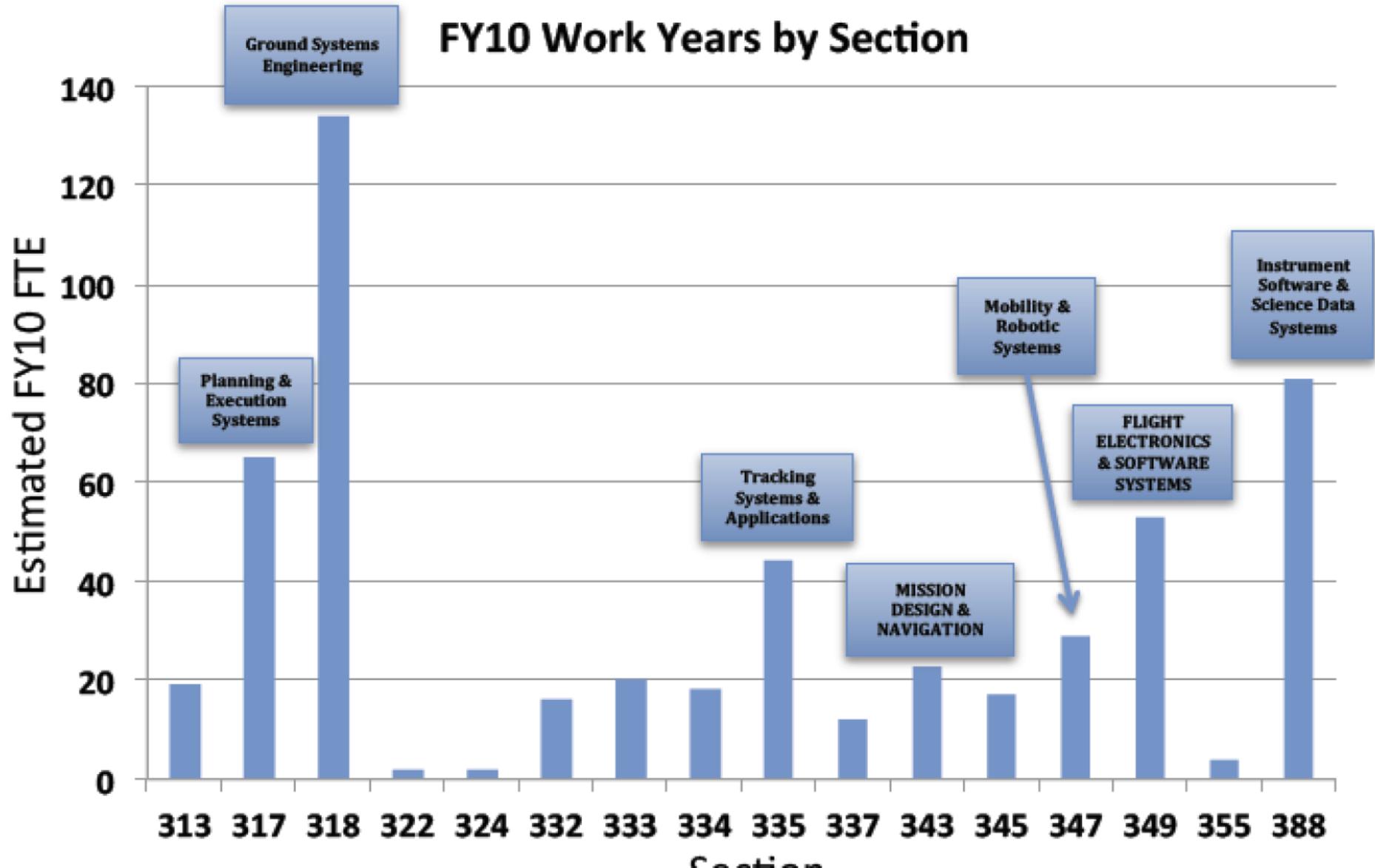


Rover Analysis, Modeling,  
and Simulation



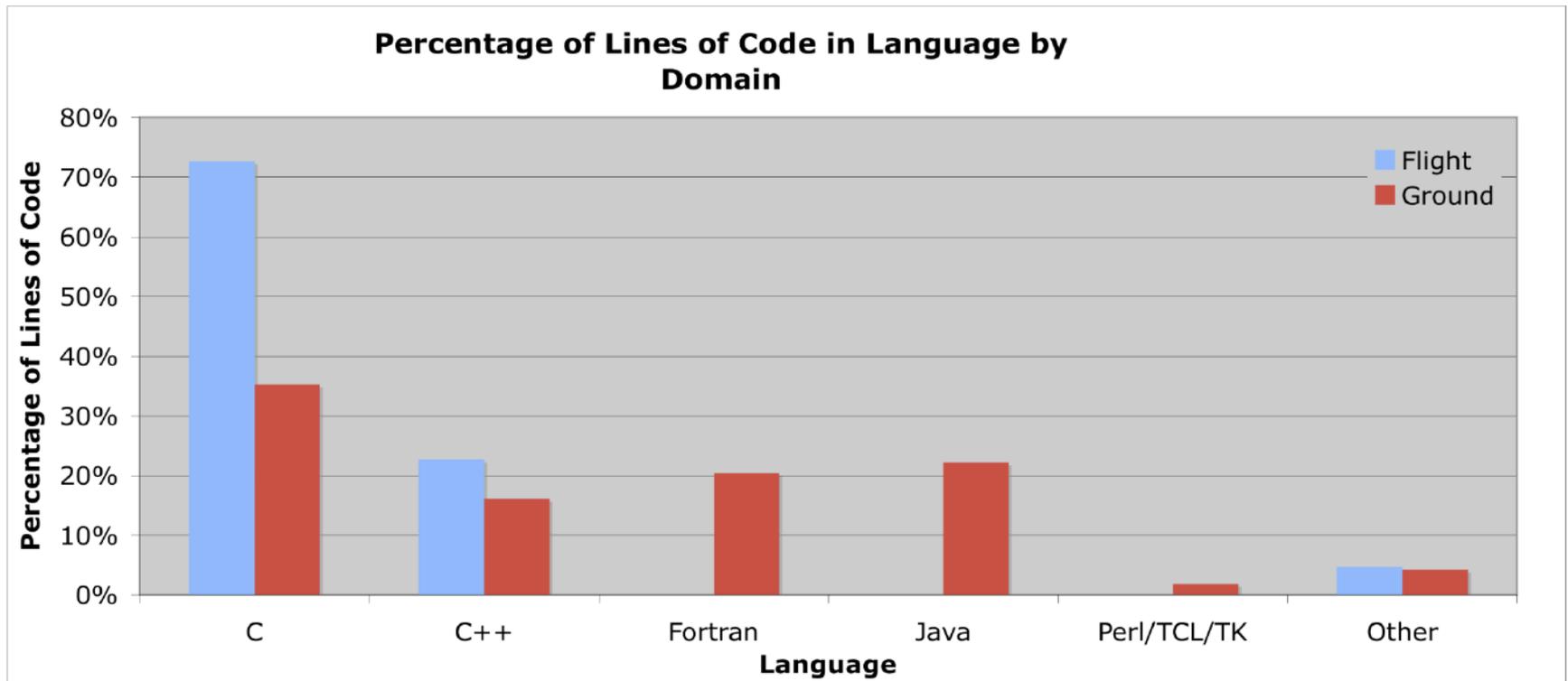
# Software: Basic Facts

- Software is spread across many sections in 3X



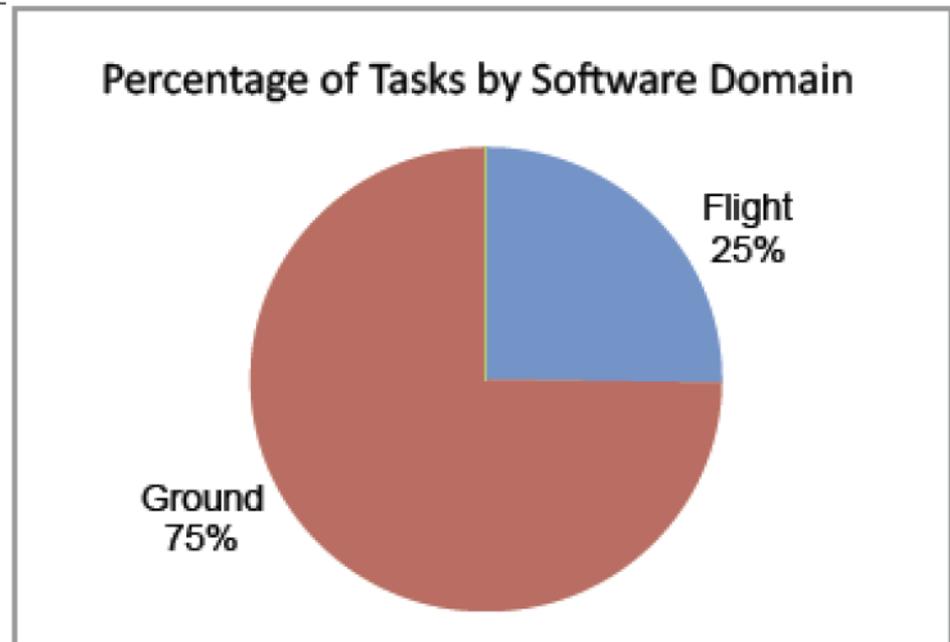
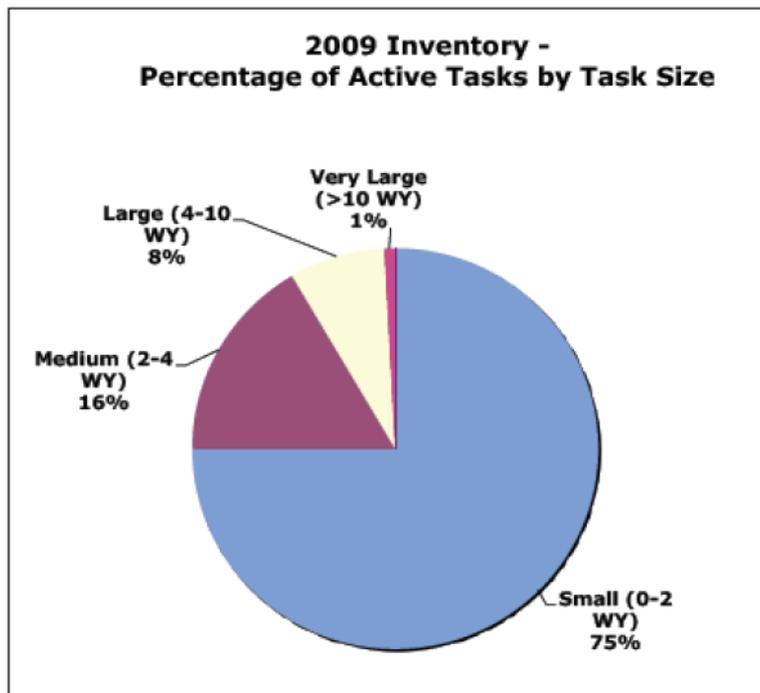
# Software: Basic Facts

- **Many languages are in use across the Lab**
  - C is the primary language for flight software
  - Java is the primary language for new code in development on the ground



# Software: Basic Facts

- **As of 2010, there are approximately 53 million SLOC in development or actively being maintained at JPL (36 million safety critical)**
  - 345 software tasks
  - 250 tasks are Class B and Class C code
  - 75% of tasks are Small (0 - 2 work years), 25% Code if Flight Software



# Why is FSW important?

## 1. Control the spacecraft

- ACS/GNC ~ Fly the spacecraft
- Misc. Sensors & Actuators

## 2. Data Acquisition

- Collects Raw science & Systems Data Products

## 3. Communications

- Flight/Ground Interfaces
- Uplink & Downlink

## 4. Provides Spacecraft Autonomy

- Sequencing is standard
- More exotic planners and agents

# Why is FSW important?

- Captures Mission system behavior
  - Flight software has become more pervasive is the system
- Glue that makes or breaks the mission!
  - Deployments are largely dominated by interfaces
- Always designed and implemented to achieve mission requirements

# Project Requirements Levels

Requirements level within the project shall be established using the following definitions:

- Level 1: Customer requirements** are the sponsoring organization or program derived and allocated requirements on the project
  
- Level 2: Project requirements** are those requirements that are both levied on the Project from the Program or institutional level as well as those that are derived at the project level
  
- Level 3: System requirements** are those requirements that are both allocated functions to each system from the controlling Project System as well as those that are derived at the system level
  
- Level 4: Subsystem requirements** are those requirements that are both allocated functions to each subsystem from the controlling system entity as well as those that are derived at the subsystem level
  
- Level 5 and below requirements** are defined similarly

# Requirements Level, Control & Location

## Element Controlling the Requirement

<u>Level</u>	<u>Element</u>	<u>NASA HQ</u>	<u>Project System</u>	<u>Systems (e.g. FS, MS)</u>	<u>Sub-systems (e.g. ACS, Nav)</u>
Level 2	<u>Project System</u>	Negotiated	Self Derived		
Level 3	<u>Systems (e.g. FS, MS)</u>		Allocated	Self Derived	
Level 3	<u>Inter-System</u>		IRD(s)		
Level 4	<u>Subsystems (e.g. ACS, Nav)</u>			Allocated	Self Derived
Level 4	<u>Inter-Subsystem</u>			IRD(s)	
Level 5	<u>Assembly</u>				Allocated
Level 5	<u>Inter-Assembly</u>				IRD(s)

Element Module Containing and Satisfying the Requirement

IRD = Interface Requirements Document

Allocating element maintains control of the requirement

# Innovative Technology (Avionics & Flight Software) Development Driven by New Cal Poly CubeSat Applications

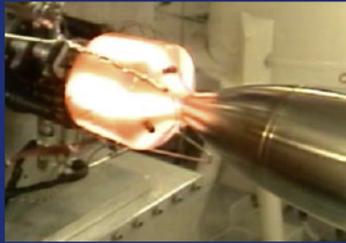
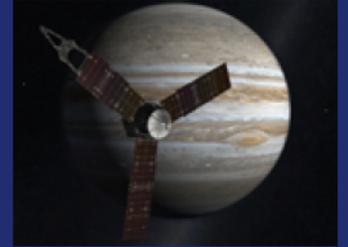
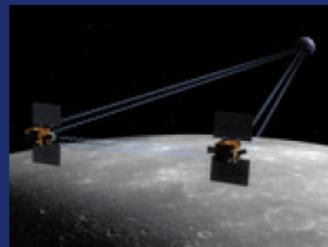
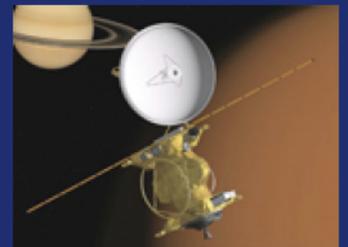
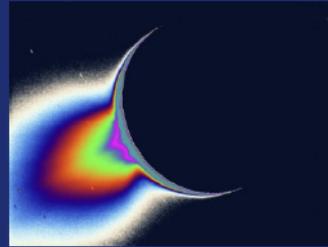
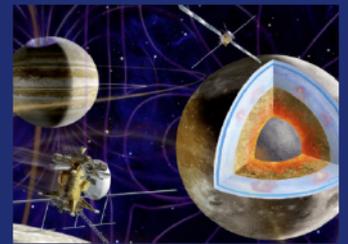
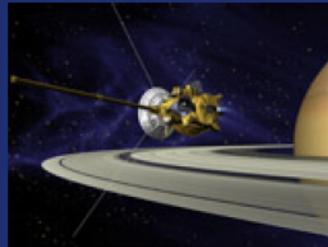
- Goals: **Educate** Cal Poly students about JPL and **explore collaboration opportunities** in new avionics and software demonstrations utilizing the CubeSat form factor.
- Objectives:
  1. Answer the technical question, can new technologies be implemented and tested on CubeSats for later adoption by JPL?
  2. Consider a broad range of student-developed CubeSat mission concepts as drivers?
  3. Can we identify a set of FSW/Avionics technologies that could be demonstrated utilizing the student concepts with justification of importance?

# Future Directions in Mission Computing

Dr. Larry A. Bergman, Manager, *Mission Computing & Autonomy Systems Research Program*

*Jet Propulsion Laboratory*

*March 30, 2012*



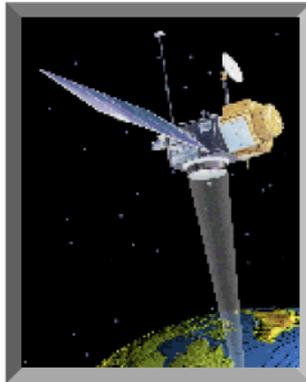
# Software, Computing and Networking on Space Missions

JPL

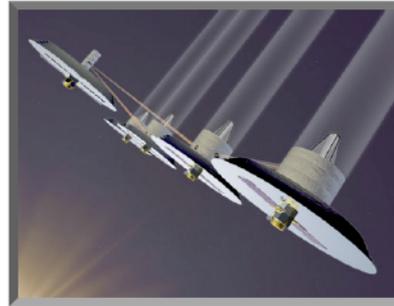
## Mars and Planetary Missions



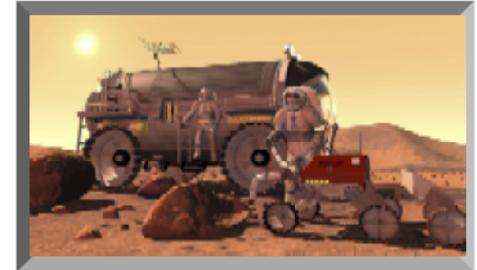
## Earth Observing Missions



## Astrophysics Missions



## Human Exploration Missions



### *Flight Systems*

Mission Planning & Execution -- Science Event Detection & Data Processing -- Model-based Fault Management -- Control Architectures -- Fault-Tolerant Computing Architectures -- Machine Vision for EDL -- Autonomous Navigation -- Autonomy for Surface Ops -- Human-Robotic Operations -- Software Reliability

### *Space Networking*

Space-based Protocols -- Content-based Data Compression -- Downlink Prioritization -- Delay- and Disruption-Tolerant Networking (DTN) -- Relay Operations -- Demand Access -- Multi-Platform Coordination



### *Ground Systems*

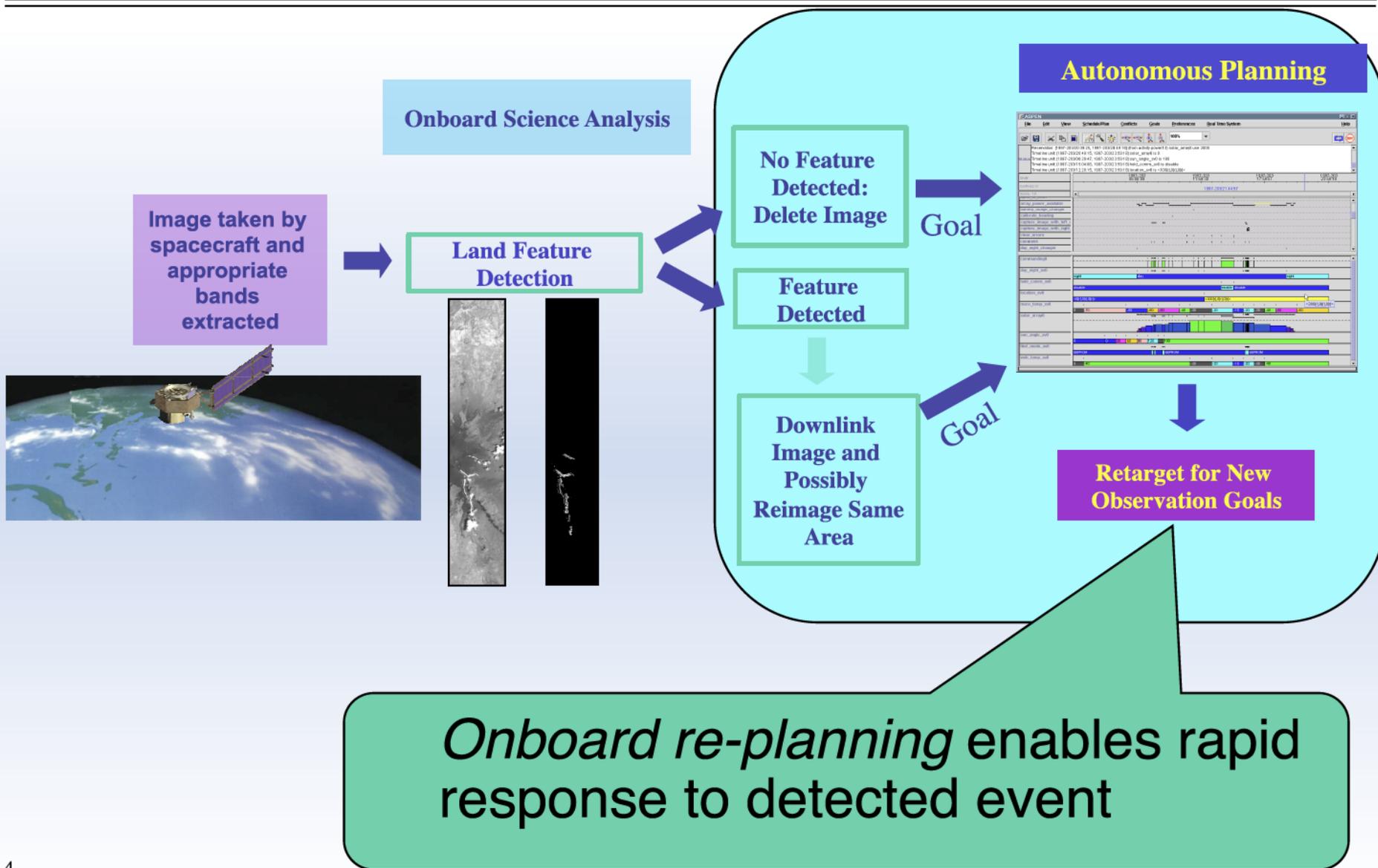
Ground System Automation -- Mission Planning & Execution -- Modeling & Simulation -- Design Exploration -- High-Performance Computing -- Data Analysis, Visualization & Management -- Software Engineering -- Information Security -- ISHM -- Mission Operations -- Virtual Environments

# AGENDA

---

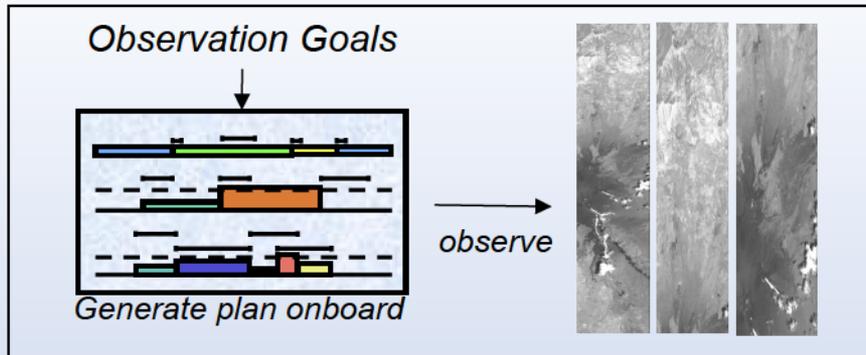
- Autonomy
- Delay Tolerant Networking (DTN)
- Flight Computing
  - Next generation processors
  - Software architecture

# Autonomous Science Experiment (ASE) on EO-1 Mission Scenario



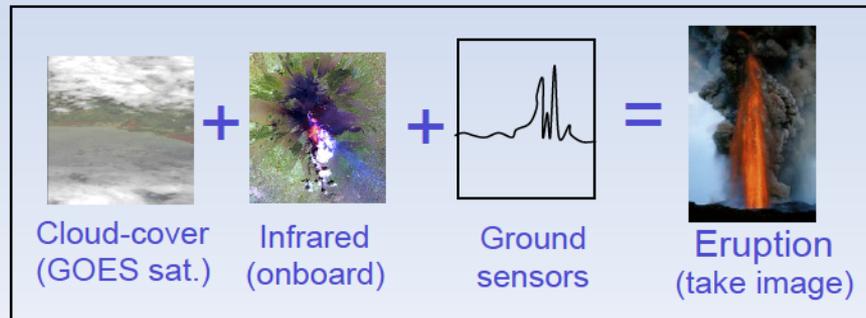
# Onboard Science Autonomous Science Experiment (ASE)

## Command



## Detect Events

(onboard recognizers, all source triggers, change detection)



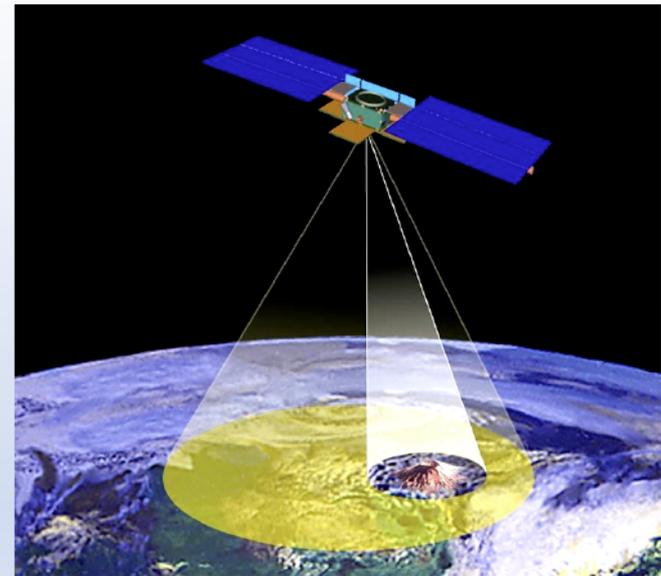
## Respond

Retask

Take new observation

Alert

Send alert and event images



Dynamic Retasking Currently  
Operating on EO-1

- Over 1,500 autonomously acquired images
- ASE retasked EO-1 to observe forest fires and floods based on multi-source event triggers
- Observations triggered from multiple data sources, including onboard recognizers, ground sensors, and data from other satellites

# Dynamic Event Detection on MER



- **Cloud and dust devil campaigns are conducted regularly on MER.**
  - These phenomena occur year-round, but vary seasonally
    - No dust devils were observed for the first Earth year of rover operations, although there was evidence that dust devils had swept over the rovers
    - During cloud campaigns, between 8 and 25% of the images contain clouds.

- **Approach**

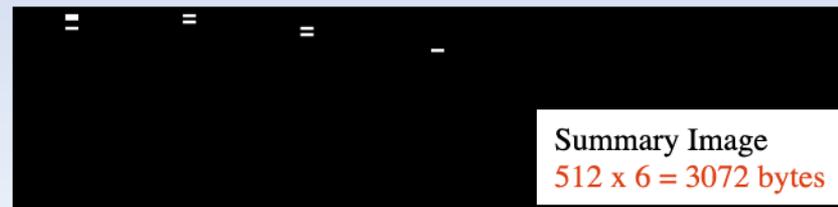
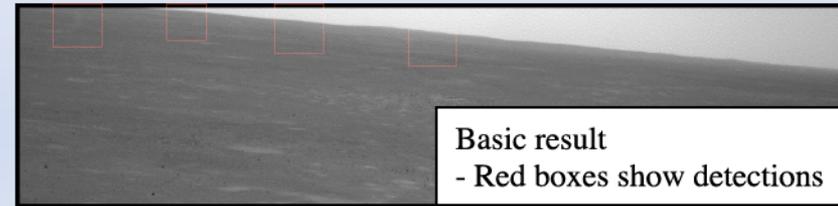
- Collect images more frequently
- Analyze onboard to detect events
- Only downlink images believed to have events

- **Benefit**

- Even < 100% accuracy can dramatically increase event data returned to Earth

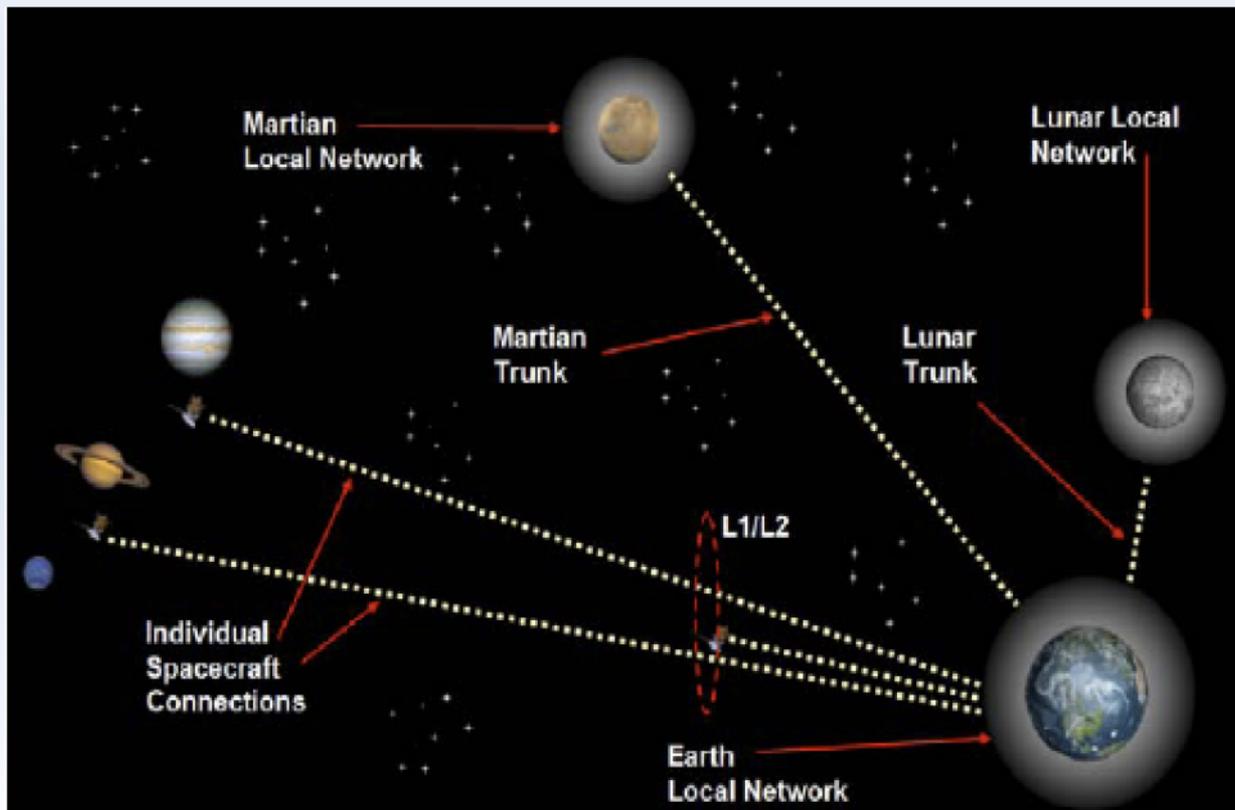
- **Status**

- Uploaded to rovers in 2006 as part of R9.2
- Fully checked out and operational



# Delay / Disruption Tolerant Networking (DTN)

- Disruption Tolerant Networking (DTN) protocols to enable NASA transition from point-to-point to networked communications.



- Enables coordinated platform science and makes space communications more efficient
- Flight validated on EPOXI spacecraft in November 2008-2010

# DTN Flight Validation Activities

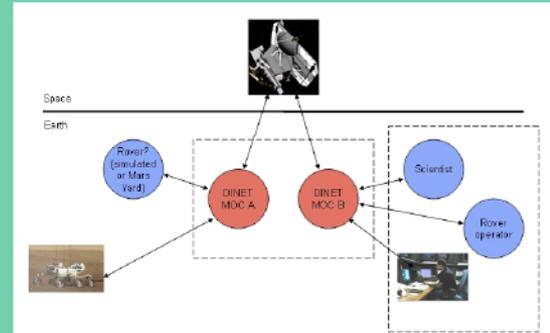
## Validation Experiment

## Topology

### DINET 1: Core DTN Flight Validation

- Priority
- Dynamic Routing
- Automated Forwarding
- Custody Transfer
- Delay-Tolerant Retransmission
- Flow & Congestion Control

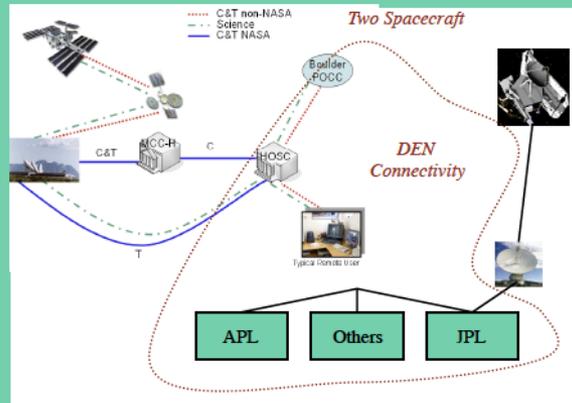
*Single Spacecraft  
One DEN Node*



### DINET 2: Flight Validation of DTN Enhancements for Mission Ops

- DTN with unacknowledged CFDP overlay
- Bundle Security Protocol (BSP)
- Dynamic Contact Graph Management
- Demo Multiple ION Network Nodes
- Extended Priority System
- Potential use by EPOXI ops team to transfer science files using DTN

*Two Spacecraft*



### Future Demonstrations

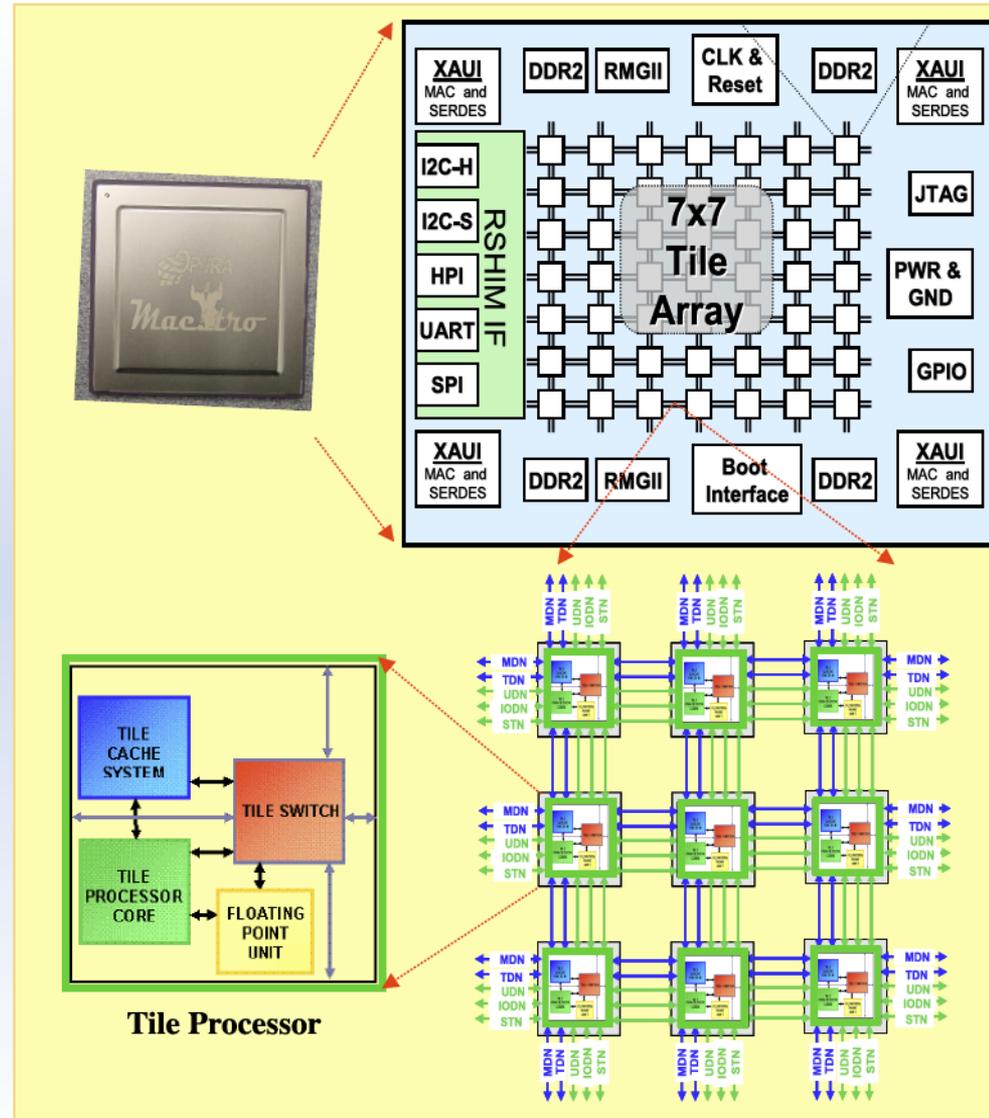
- Asynchronous Messaging Service (AMS)
- Space Network Time Protocol (NTP)
- One Way Light Time Calculator

*Provides additional features  
enabling autonomous  
operations of DTN functionality*

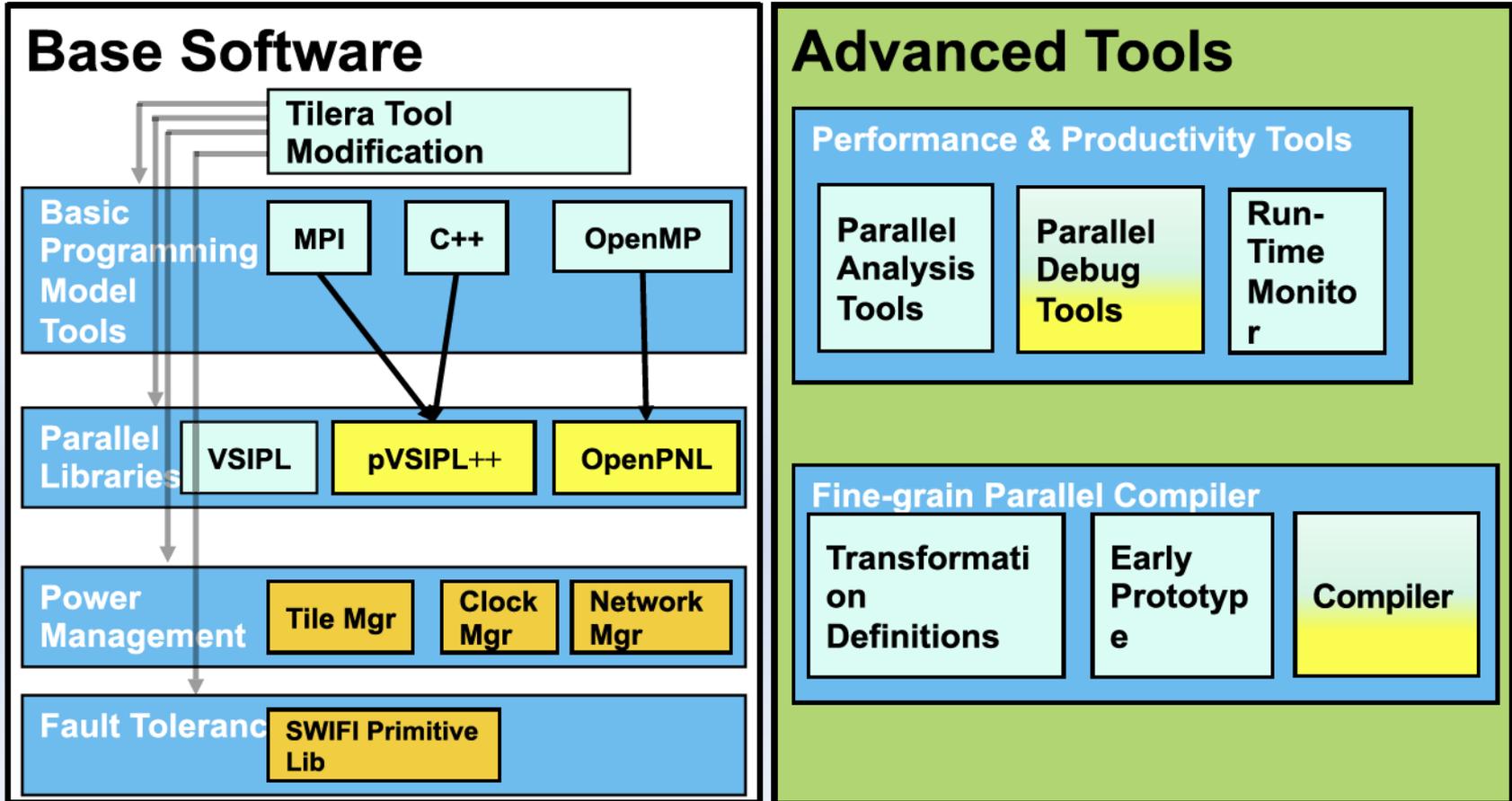
# Maestro Flight Processor

## Key Features

- Highest Performance Rad Hard General Purpose Processor
  - Up to 350 MHz, 44 GOPS, 20 GFLOPS
- Tiled Architecture
  - 49 tile, 2-D Processor Array connected by low-latency high bandwidth register-mapped networks
- Tile Processor
  - Main Processor: 3-way VLIW CPU
    - 64-bit instruction bundle
    - 32-bit integer operations
  - Static Switch Processor
  - Floating Point Co-Processor (IEEE 754 single and double precision)
- Memory
  - L1 cache: 2 cycle latency, 8KB I & 8 KB D
  - L2 cache: 7 cycle latency, 64 KB
  - Tiles can access each others L2
  - Off-chip Main Memory: 88 cycle latency
- I/O Interfaces
  - Four XAU1
  - Four DDR2



# Maestro Software Architecture



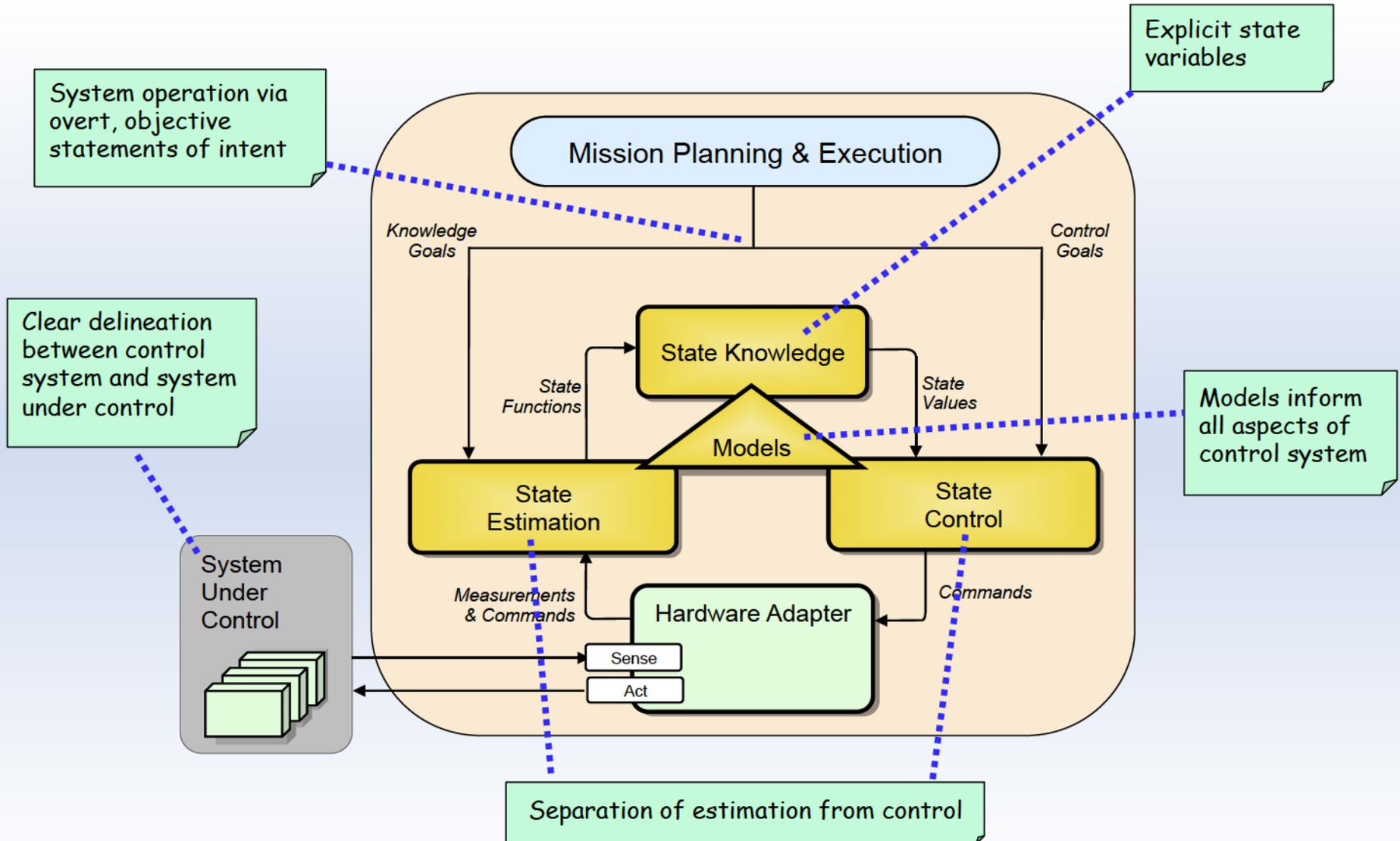
**Initial S/W Suite Developed and Validated**

Future Mission Needs

Future Research Areas

# Mission Data System (MDS) Control Architecture

## *Unified architecture for flight & ground*



# Possible Software Research Areas

---

- Artificial Intelligence / Onboard Autonomy
- Delay Tolerant Networking (DTN)
- Multicore Flight Processor & Software
- Mission Data System (MDS) control architecture and supports autonomy
  - State Analysis
  - Goal-Based Mission Operations
- Decentralized Coordinated Control of Satellites and UAVs
- Mars Sandbox at Cal Poly

# Questions?

---

Contact Info:

**Larry Bergman**

Manager, Mission Computing and Autonomy Systems  
Research Program Office (436)

4800 Oak Grove Drive – MS 321-B60

Jet Propulsion Laboratory

Pasadena, CA 91109

Email: [Larry.A.Bergman@jpl.nasa.gov](mailto:Larry.A.Bergman@jpl.nasa.gov)

Tel: (818) 393-5314



# Developing Flight Software Applications

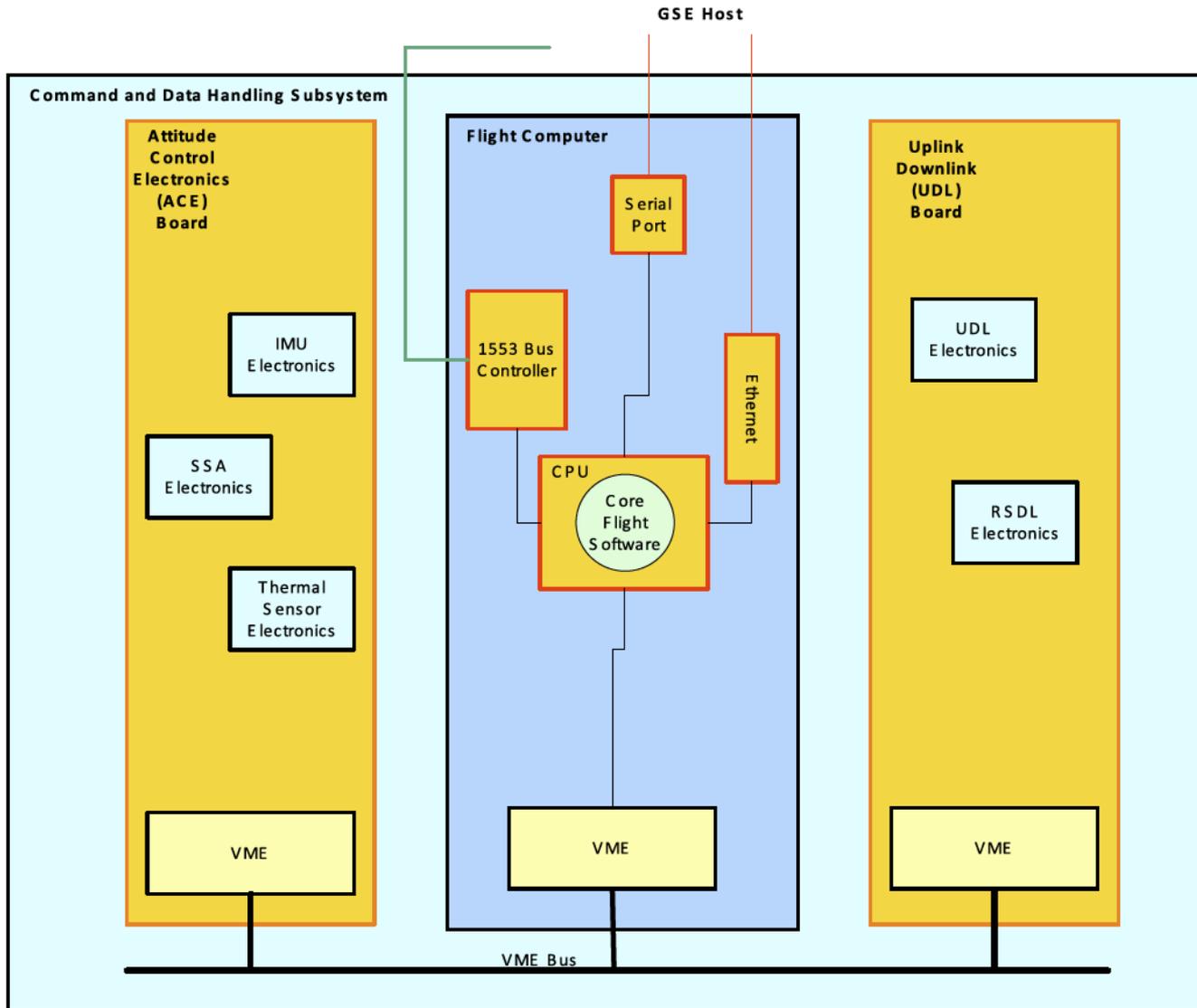
- Flight Software Architecture
  - Chassis View
  - Context Diagram
  - Module Hierarchy
  - Module Data Flow
  - Inter-task Communication
  - Task Diagram
- State-machine Modeling
  - STAARS Process
  - State-machine Autocoder
  - Building rapid executable prototype models
  - Examples
- Component Modeling
- MSL Auto-code Tools
- Auto-code Generation Process



# Flight Software Architecture

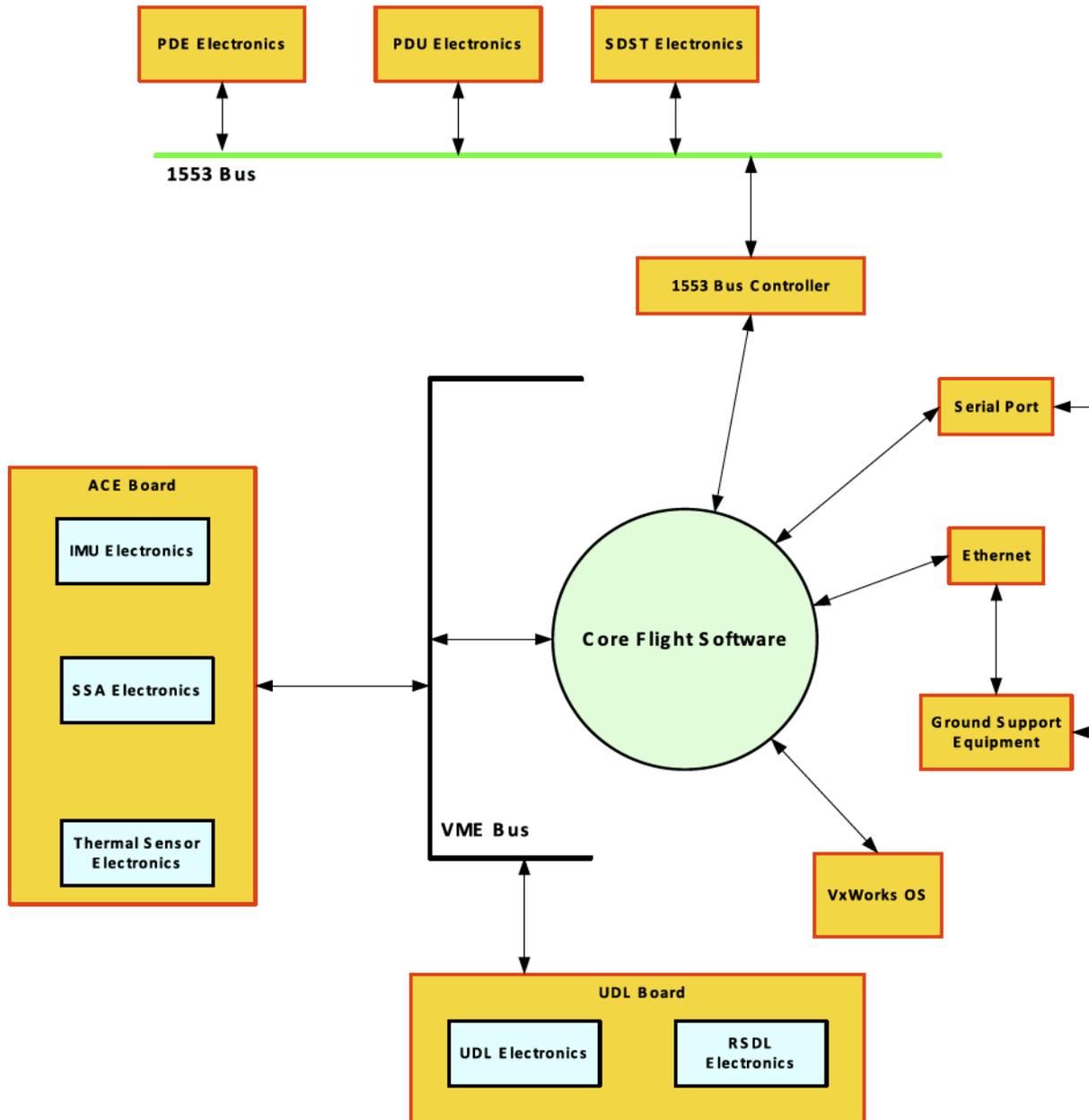


# Flight Software: C&DH Chassis View



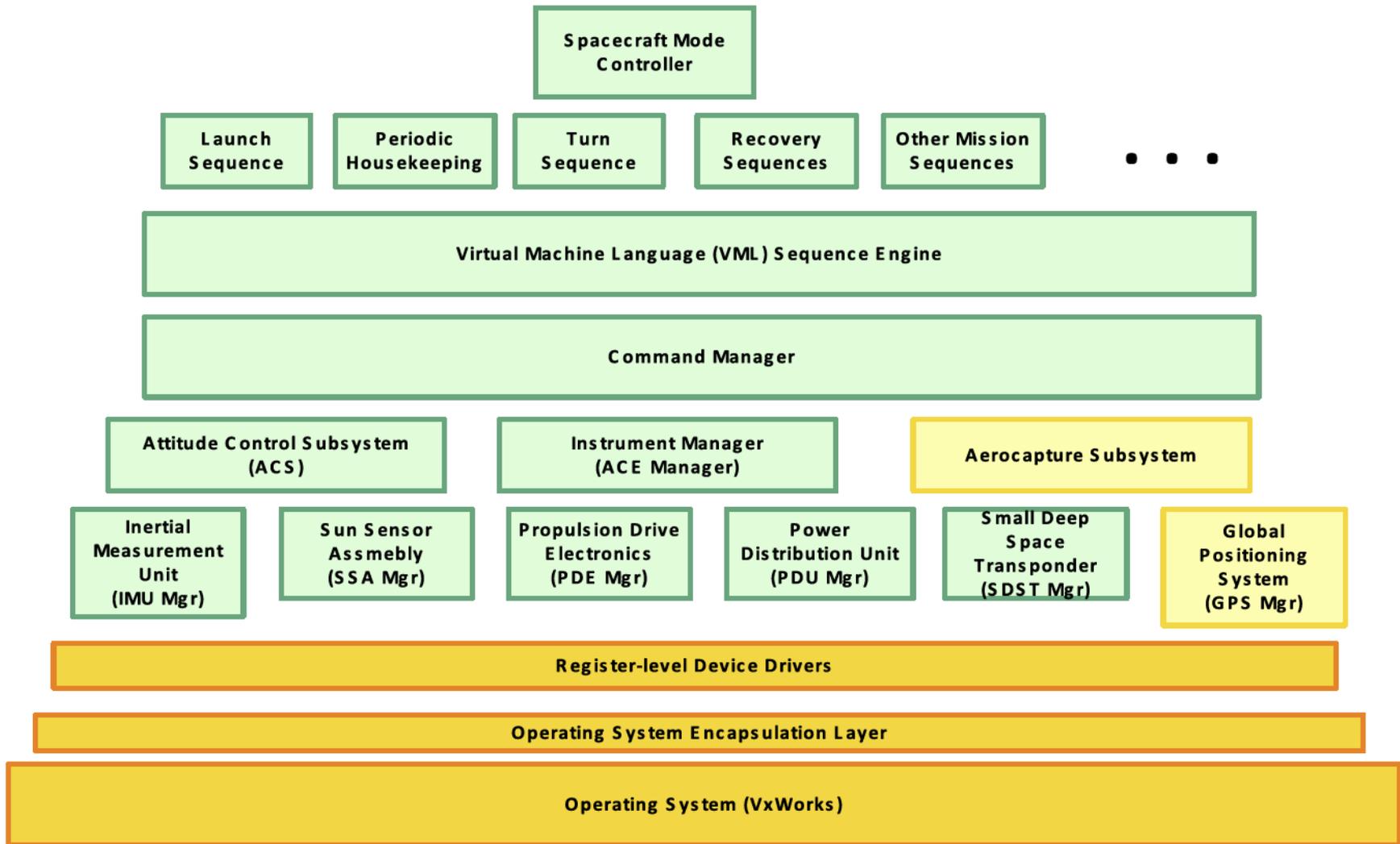


# Flight Software: Context Diagram



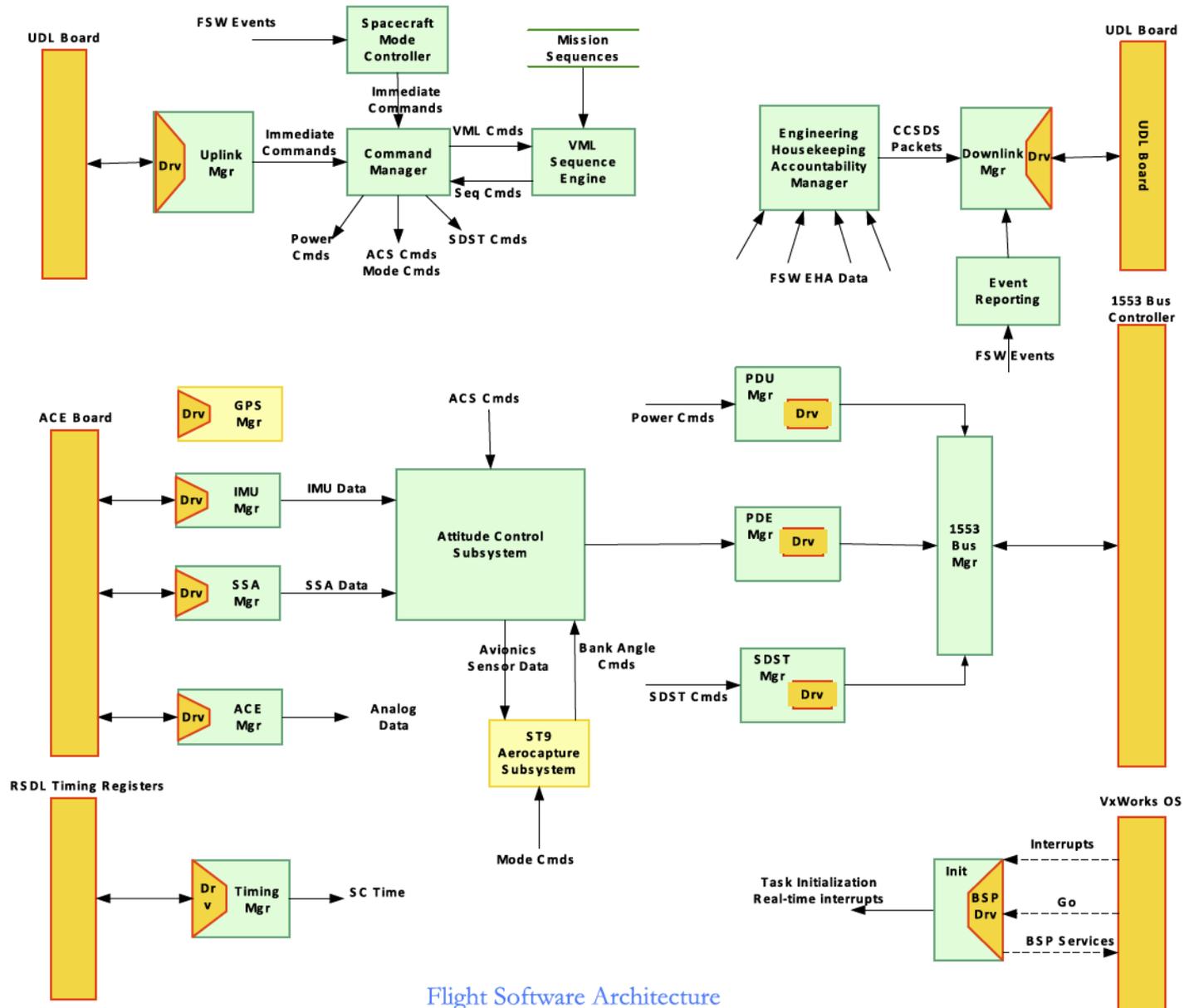


# Flight Software: Module Hierarchy



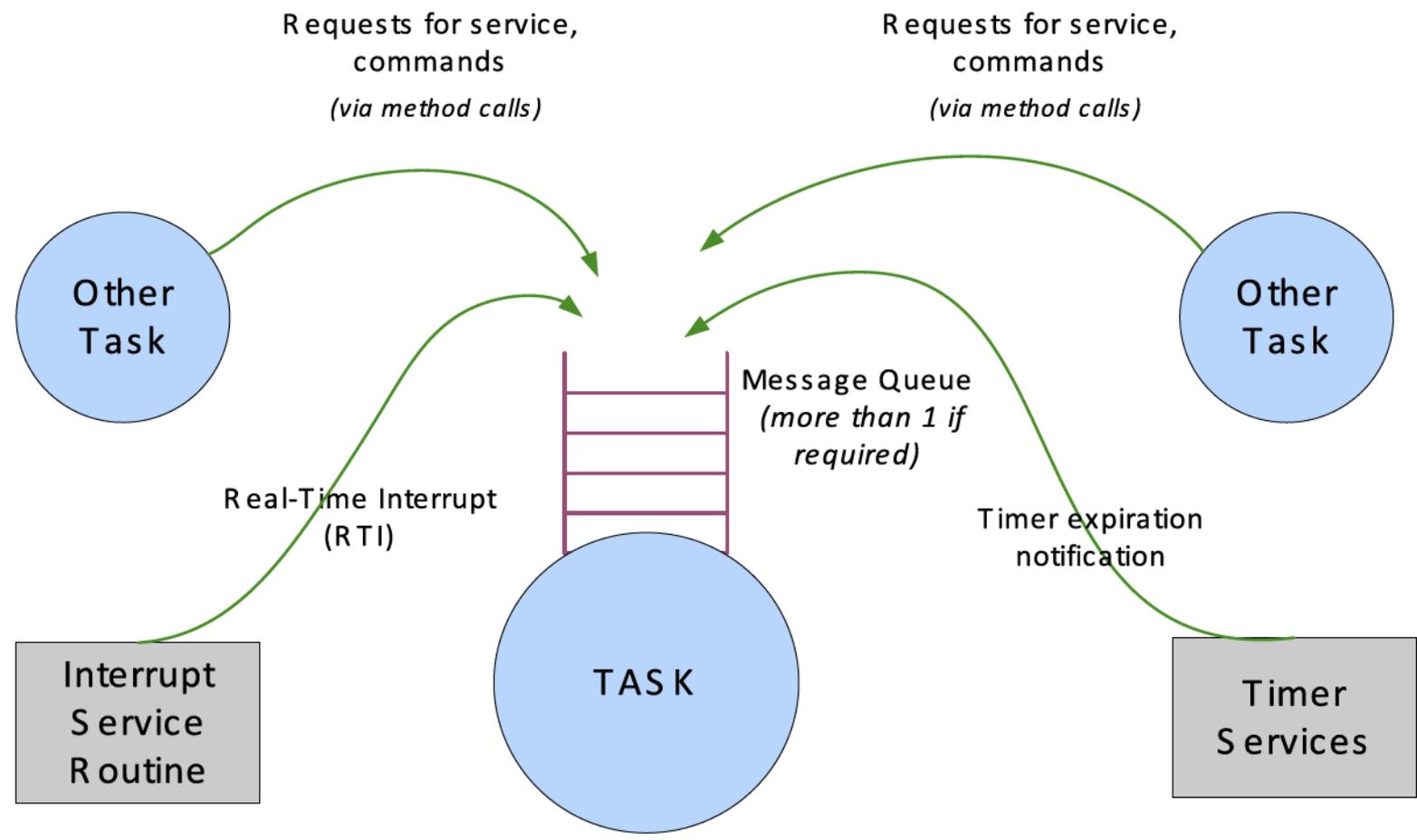


# Flight Software: Module Data Flow



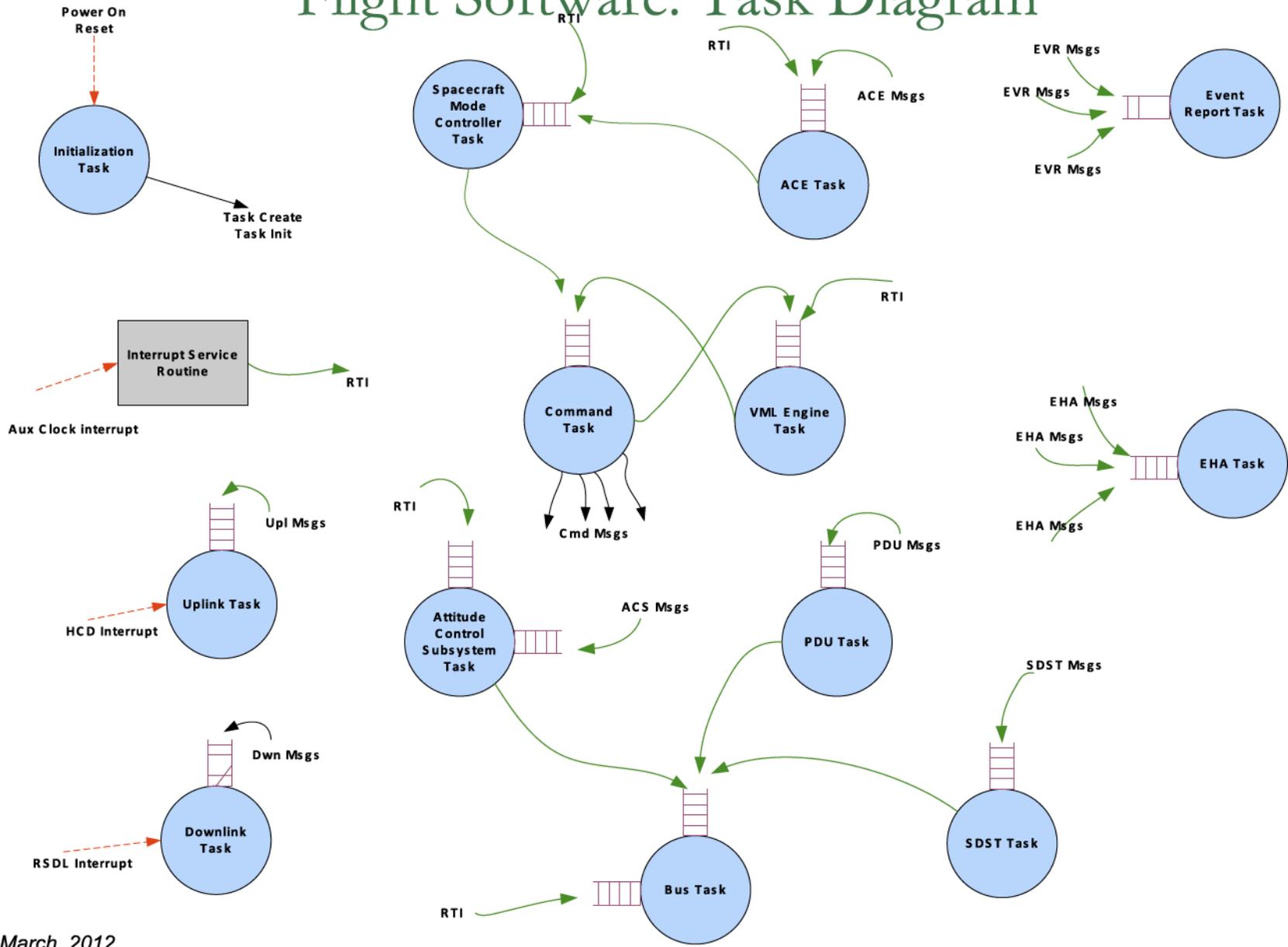


# Flight Software: Inter-task Communication





# Flight Software: Task Diagram





---

# State-machine Modeling

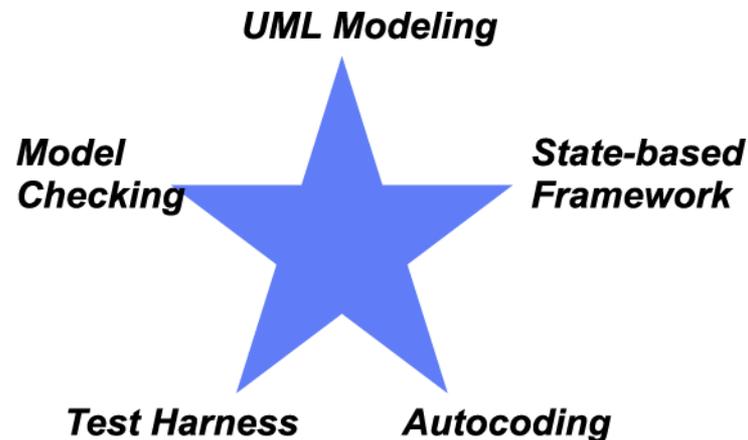


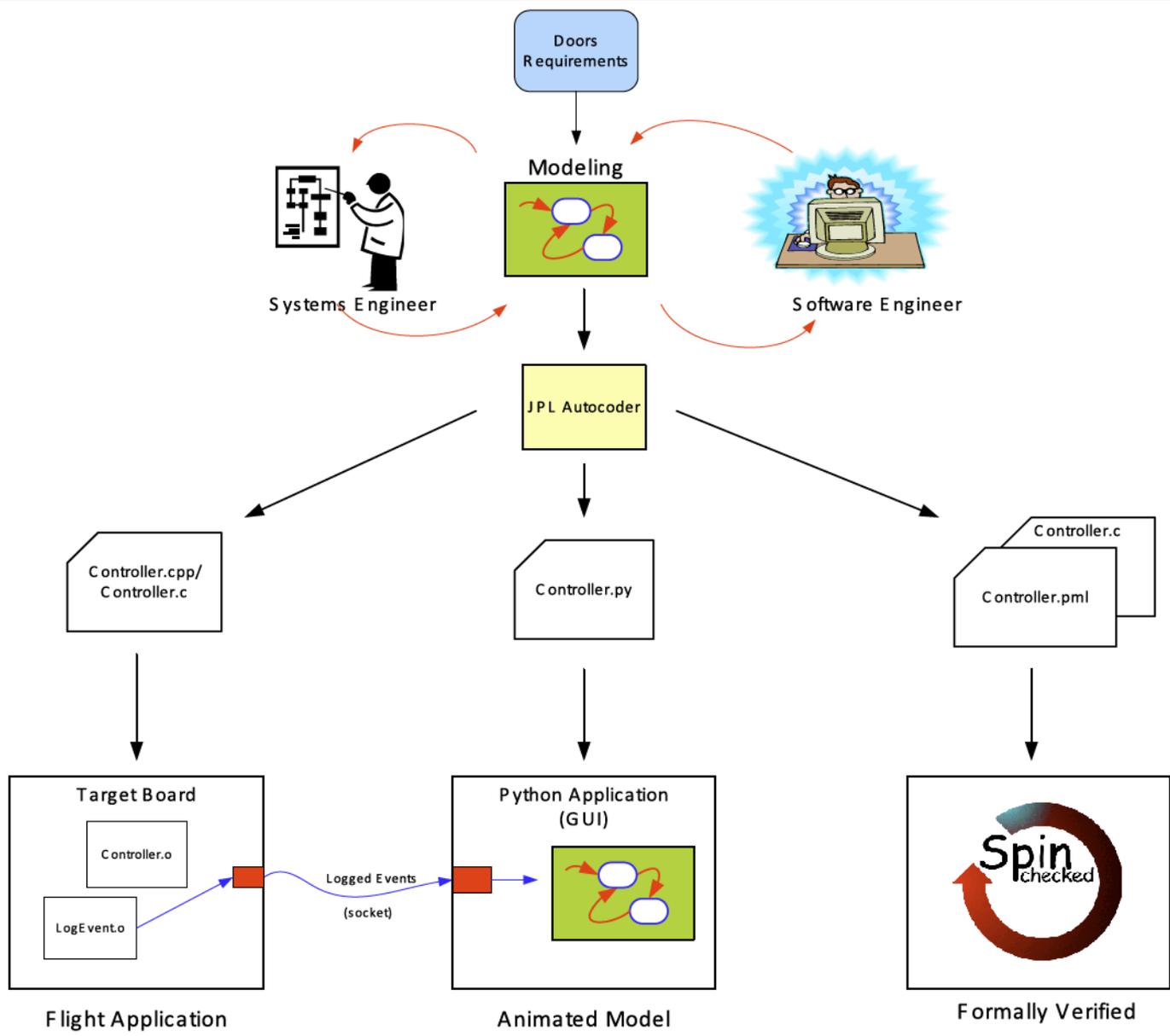
# What's the Problem?



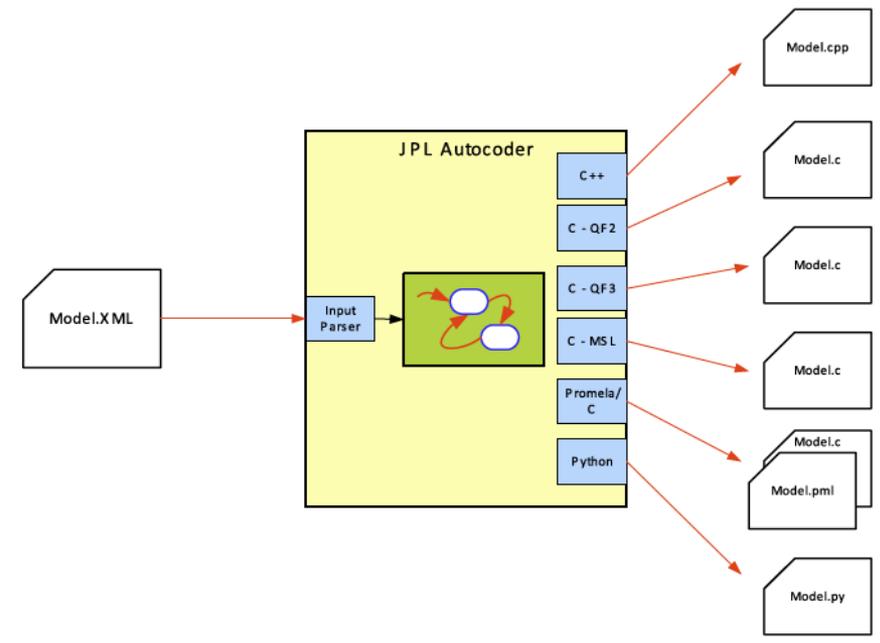
- **Defects** in the software implementation from:
  - Improperly specified Requirements
  - Misunderstood Requirements
  - Software Design
  - Manual Coding
- **Immature Requirements**
  - Need to flesh-out early Requirements
  - Need to build rapid proto-type models.
- **High cost** of software development
  - Increase efficiency of software developers
  - Increase the maintainability of the software product
  - Increase the reliability of the software product

- What is STAARS?
  - A JPL home-grown model-based engineering process for software development.
  - **STAARS** (**ST**ate-based **A**rchitecture and **A**uto-coding for **R**eal-Time **S**ystems)
  - Combines 5 tools that can be customized for each project





- **UML Modeling**
  - Explicitly capture the intent of the requirements
  - Formally capture the behavior in a model
  - Create a crisp notion of **state**
- **State-based Framework**
  - Supports the UML standard
  - Allows developers to think and work with higher constructs – states, events and transitions
- **Auto-coding**
  - Light-weight Java program
  - Reads in the Model which is stored in a non-proprietary data format (XML)
  - Converts the input model into an internal data structure
  - Has multiple back-ends to support different project requirements
- **Test harness**
  - Ability to run the model stand-alone – module test environment
- **Model checking**
  - Automatic generation of Verification models
  - Exhaustively explore the state-space of the model
  - Checks for various correctness properties within the model



## Doors Requirements

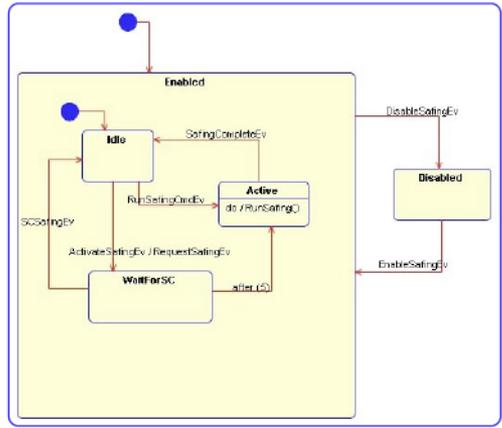


## Dynamic Behavior Model



## Auto-coded Flight Software

- The instrument shall provide an instrument safing request response.
- The instrument safing request response shall first request that the spacecraft instruct the instrument to run its instrument safing response.
- If, after TBD period following an instrument request for safing the spacecraft fails to instruct the instrument to run its safing response, the instrument shall autonomously run the instrument safing response.
- Each instrument fault monitor shall provide a means to disable or enable each individual type of notification to the fault handler of persistent fault symptoms.
- The enabling of any instrument fault response shall cancel any outstanding requests for that response (which may have occurred while response was disabled).



```

QSTATE Safing::Idle(QEvent const *e) {
string stateName = objName + " Idle";
switch (e->sig) {
case Q_ENTRY_SIG:
    LogEvent::log(stateName + " ENTRY");
    return 0;
case Q_EXIT_SIG:
    LogEvent::log(stateName + " EXIT");
    return 0;
case ActivateSafingEv:
    LogEvent::log(stateName + " ActivateSafingEv");
    QF::publish( Q_NEW(QEvent, RequestSafingEv)
);
    Q_TRAN(&Safing::WaitForSC);
    return 0;
case RunSafingCmdEv:
    LogEvent::log(stateName + " RunSafingCmdEv");
    Q_TRAN(&Safing::Active);
    return 0;
}
return (QSTATE)&Safing::Enabled;
}

```

## Test Harness

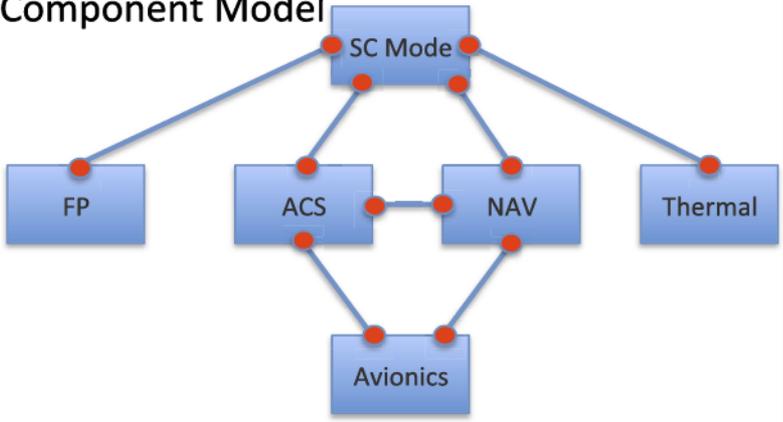




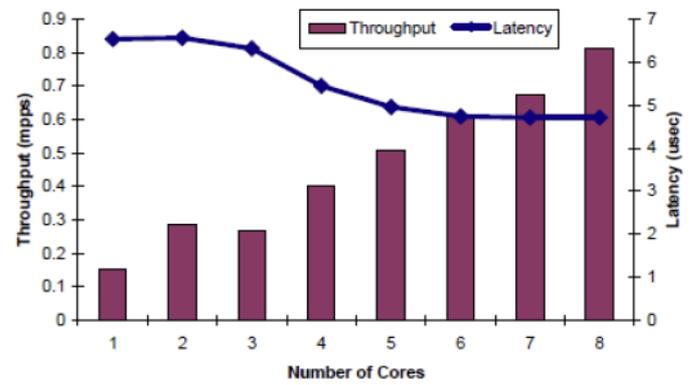
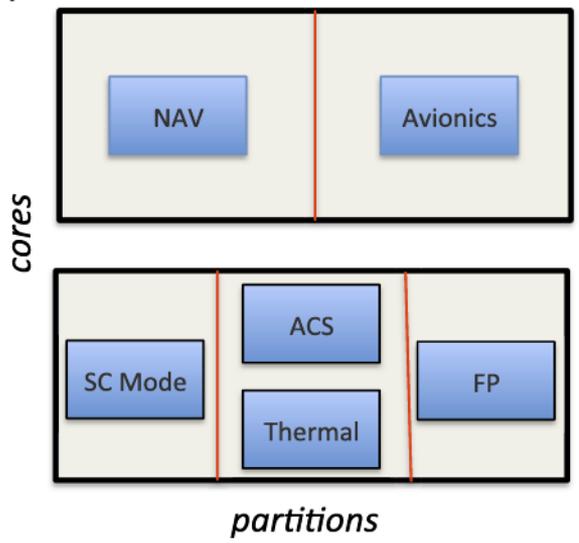
---

# Component Modeling

### Component Model



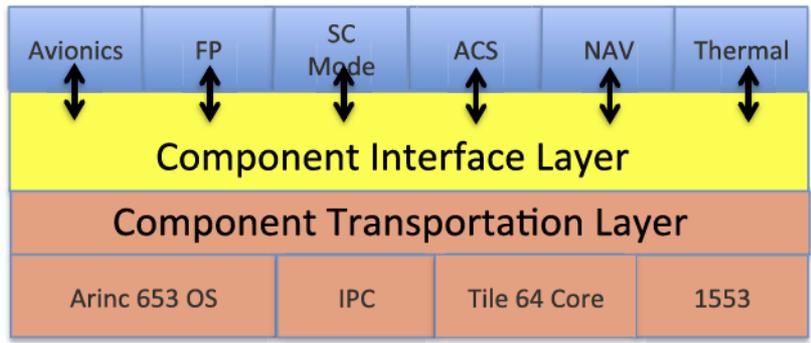
### Deployment Model



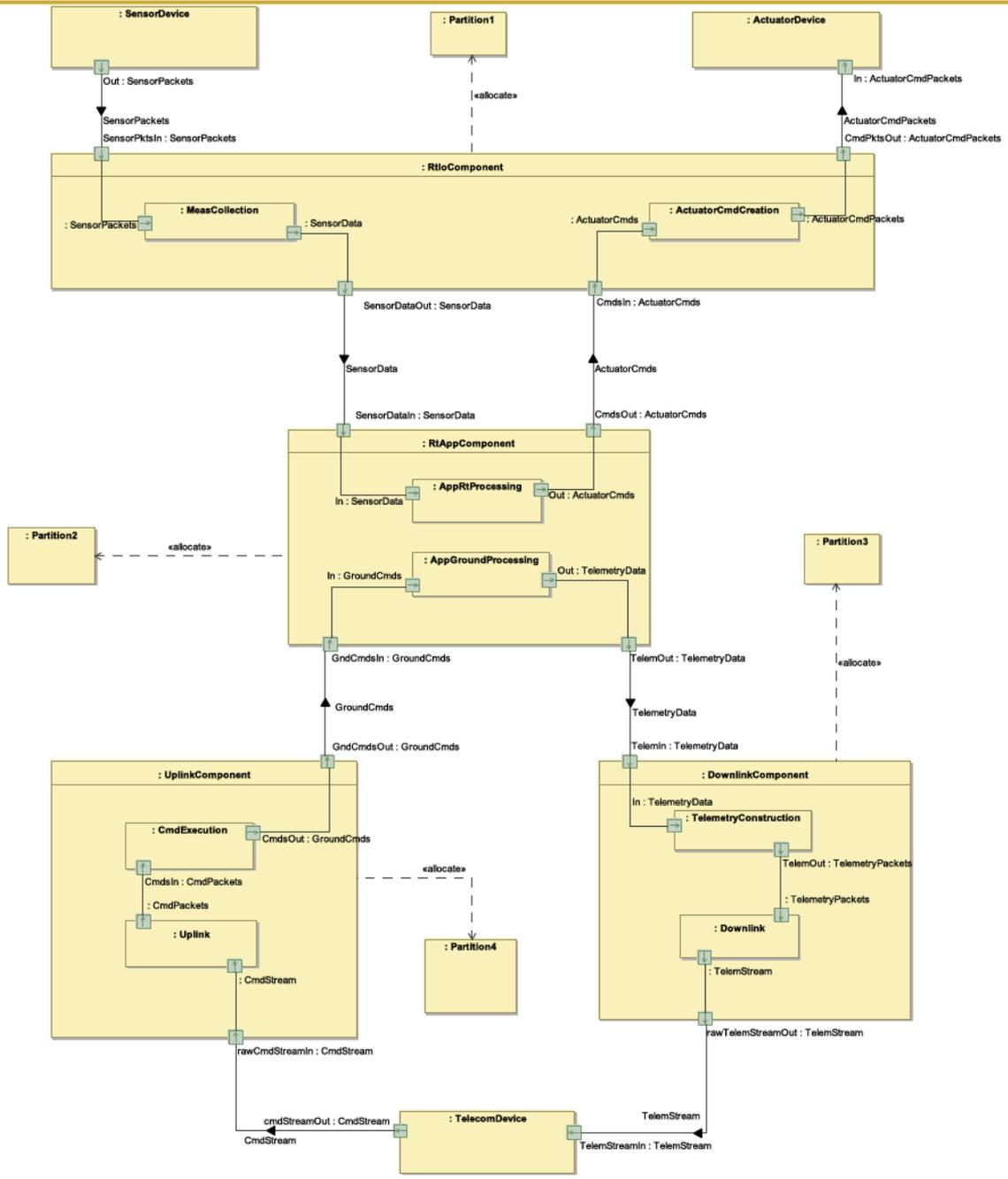
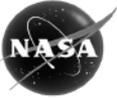
Single Modeling Artifact

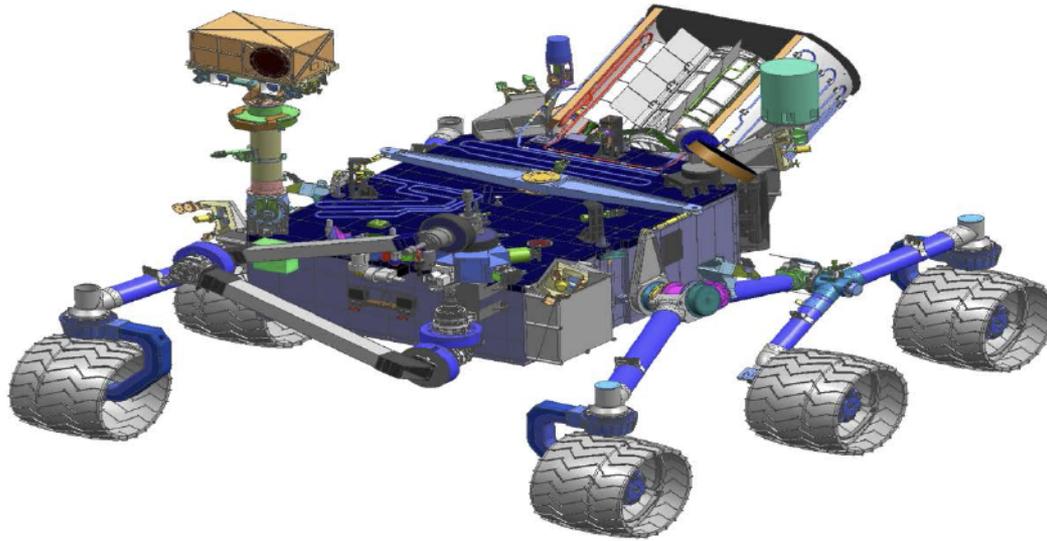
Analysis

Deployment









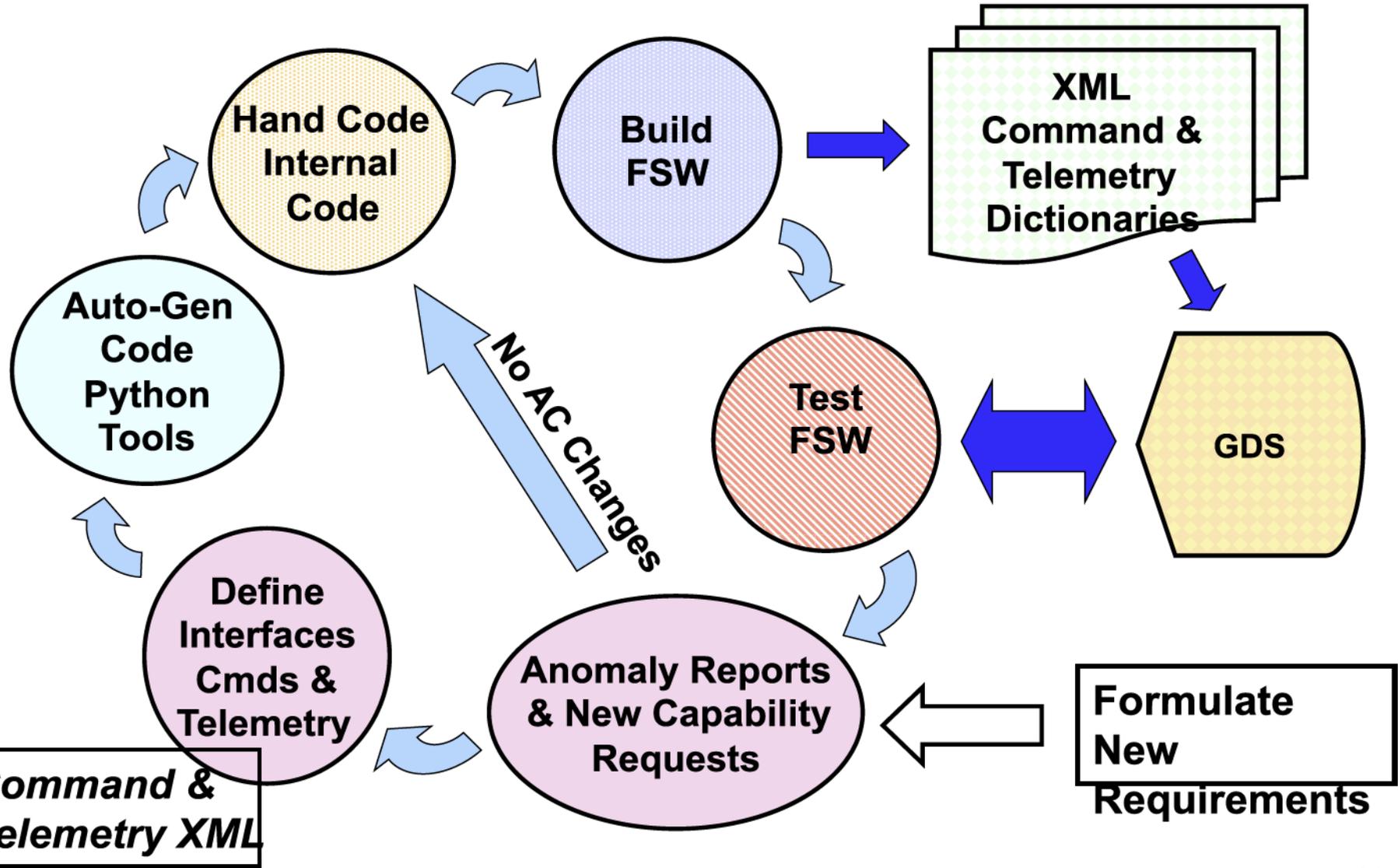
## Embbed System

- VxWorks/C Impl
- Multi-threaded
- Module to module message based

- Provide formal interface specifications (XML)
- Python tools generate tested and well understood patterns
- Rapidly implement new framework code via automation



# Auto-Code Generation Process





Jet Propulsion Laboratory  
California Institute of Technology



# Avionics Technology: Present and Future

Yutao He

Advanced Computer Systems and Technologies Group

Flight Electronics and Software Section

Jet Propulsion Laboratory

March 30, 2012



# Outline



- Overview of Avionics
- Present Avionics Technology
- Future Avionics Technology

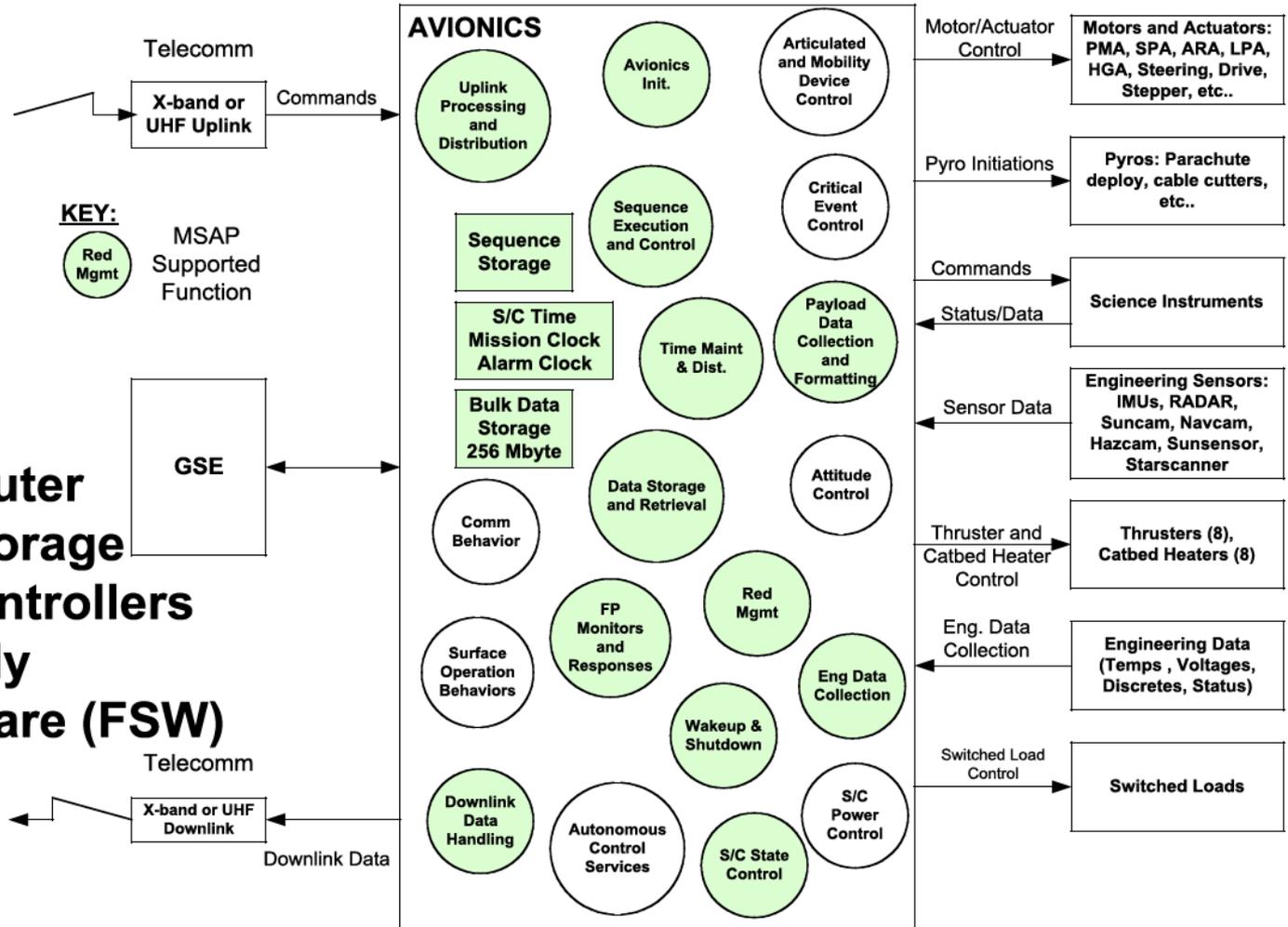
# What is Avionics?

## *Avionics*

1. *The science and technology of electronics and the development of electronic devices as applied to aeronautics and astronautics.*
  2. *The electronics systems, equipment, and other devices so developed.*
- Avionics include the following components:
    - **Command and Data Handling (C&DH) equipment**
      - Computers, memory devices, peripheral interfaces and system buses
      - CDH software, fault protection software
    - **Power System Equipment**
      - Power switches, DC to DC converters, valve drive and pyro drive electronics, power bus regulation electronics, power sources, energy storage
    - **Attitude Control Sensors**
      - Star trackers, gyroscopes, sun sensors, accelerometers
      - Software for attitude control, guidance and navigation
    - **Telemetry electronics**
      - Temperature monitors, voltage monitors
    - **Motor drive and control electronics**
    - **Software** for low level hardware interfaces, device drivers



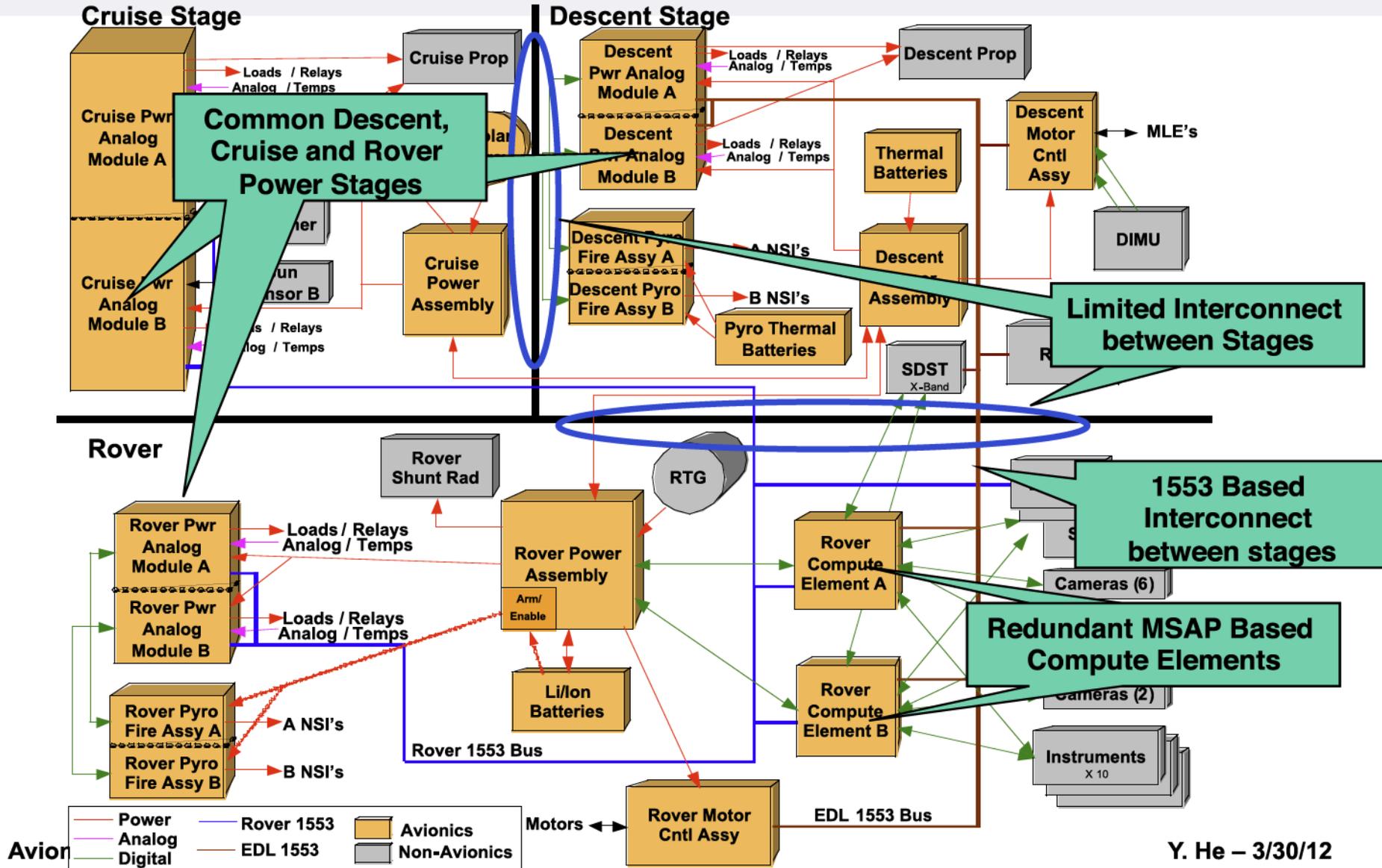
# Avionics Functional Block Diagram



- Flight computer
- On-board storage
- Interface Controllers
- Power supply
- Flight Software (FSW)



# Present - MSL Avionics Architecture



# MSL Avionics Hardware

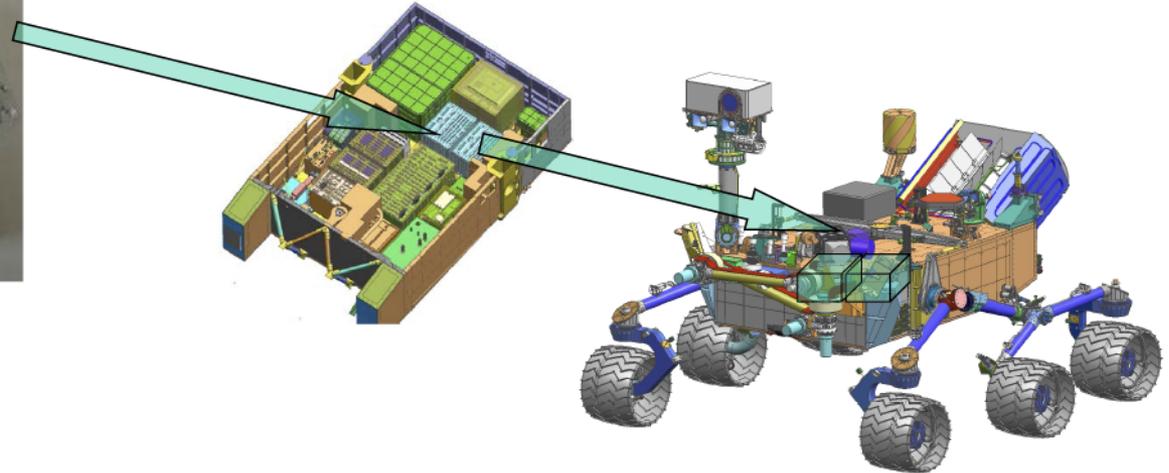




# Avionics - System Context



RCE Integrated Chassis



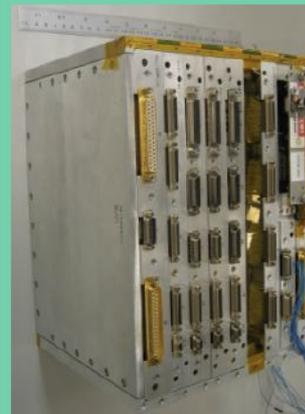


## State-of-the-art COTS: Dell OptiPlex XE Workstation



- 3 GHz Dual-Core Intel Core 2 processor
- 8 GB DDR3 SDRAM
- 500 GB Hard drive
- Many fancy interfaces
  - SATA 3
  - USB 2.0
  - 1394 FireWire
  - Ethernet ports/DVI ports
  - UARTs
- Mass: 6.48 Kg
- Power: 300 W
- Volume: 85.2mm x 290mm x 324mm

## State-of-the-art Avionics: MSL C&DH



- 133 MHz Rad750 PowerPC processor
- 128 MB SDRAM with EDAC
- 8 GB NVM
- Typical avionics interfaces
  - LVDS
  - RS422
  - 1553B
  - UARTs
  - Ethernet ports
- Mass: 10 Kg
- Power: 40 W
- Volume: 209mm x 188.3mm x 280mm



The best way to invent the future is to invent it.

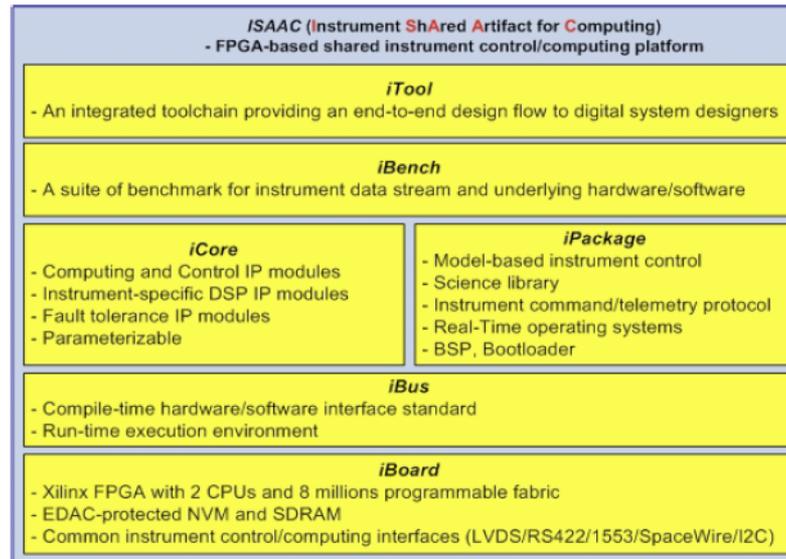
*- Alan Kay, 2003 ACM Turing Award Winner*

- Reusable FPGA Computing Platform – **ISAAC**
- Scalable Unified 10-Gbps Spacecraft interconnect – **NEXUS**
- Low-Mass Avionics for Mars Fetch Rover – **MATCH**
- Power-Efficient Adaptive Avionics – **PEAC**
- Multicore-based computing
- Distributed Motor Controller (DMC)

- Future space-borne instruments as laid out in the Earth Science NRC Decadal Survey roadmap feature a wide range of requirements for digital systems
  - Simple microcontroller-based as in CLARREO UV-Visible-IR Interferometer
  - Complex distributed systems with high-speed interconnect and high-performance on-board data processor as in DESDynI SweepRadar
- Challenges to instrument digital system designers
  - Each instrument has its unique requirement
  - Requirements will change during the development phases
  - Algorithm design demands new technology and needs to consider the implementation technology up front
  - Designers use different tools, languages, and methodologies
- Traditional approaches will incur significantly high non-recurring cost and risk and face daunting challenges in meeting performance and function requirements

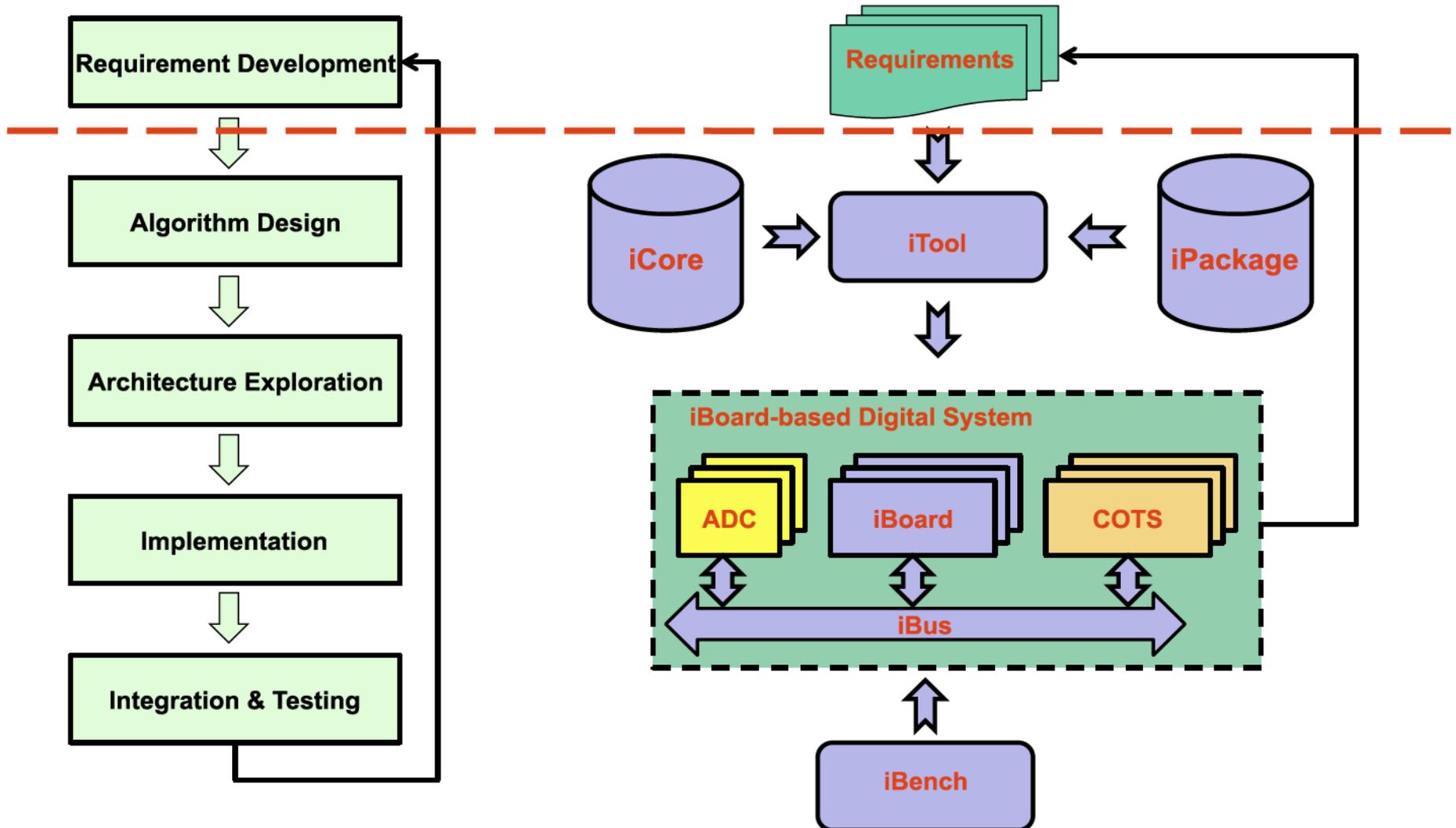


- **ISAAC (Instrument ShAred Artifact for Computing)**
  - Provide a set of six **modular** and **reusable** components under a **common framework** that can be used to **configure** a complete instrument control and computing system on a **per-application-basis** to meet various space instrument requirements as defined in the NRC Decadal Survey Report.





# ISAAC – Operational Concept





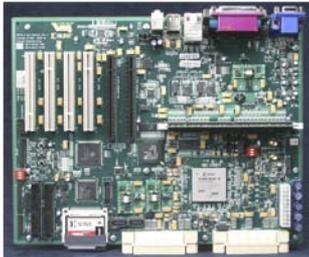
# ISAAC – In Action



**FY' 08**

## iBoard 1

- Xilinx ML410 Board
- Virtex 4 FX60 FPGA
- Prototype for other ISAAC Components



**FY' 09**

## iBoard 2

- First custom board
- Virtex 5 FX130T FPGA
  - Path-to-flight
- Targeted to DESDynI SweepSAR, GEO-CAPE panFTS, SWOT Radiometers



**FY' 10**

## iBoard 3

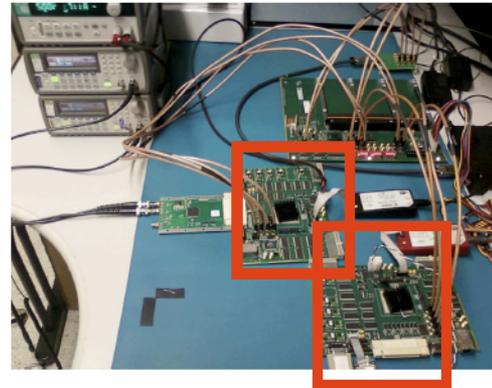
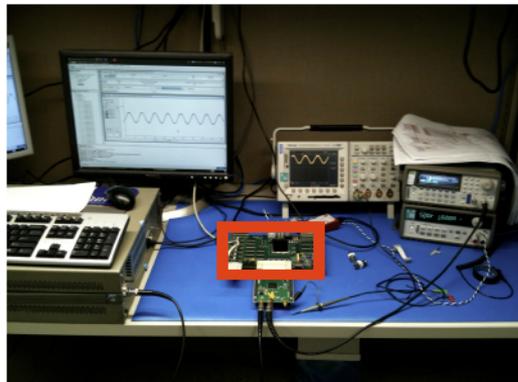
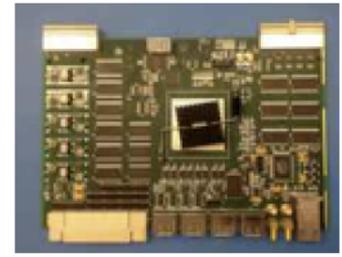
- Second custom board
- Virtex 5 FX130T FPGA
  - Path-to-flight
- Targeted to Radar Testbed, and Europa Penetrating Radar



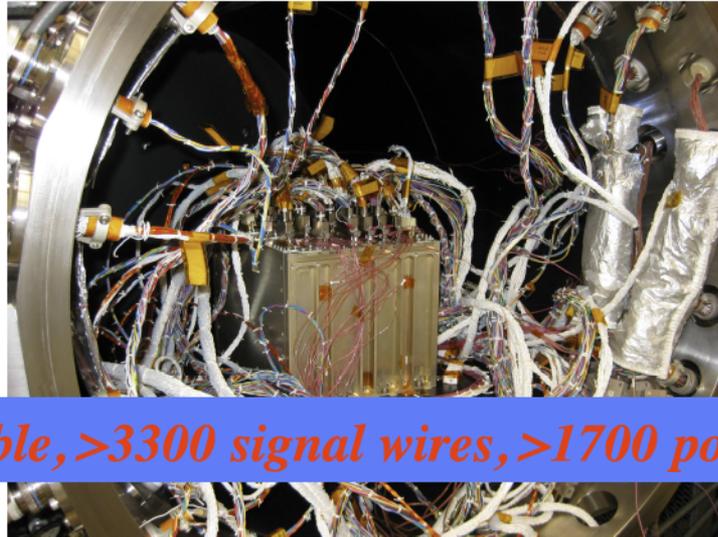
**FY' 11**

## iBoard 4

- Third custom board
- Virtex 5 FX130T FPGA
  - Path-to-flight
- Targeted to new DESDynI SweeSAR Digital Beamforming



## The Problem



*Miles of cable, >3300 signal wires, >1700 power wires*

## And what should we do in the future?

- High-speed instruments of a Gbps-bandwidth
  - SARs and hyper-spectral Imagers;
- On-board multi-core computing paradigm;
- Guaranteed real-time determinism with a few microsecond latency/jitter for tight control loops;
- Fractionated spacecraft and sample return missions that requires separability and scalability
- More suitable physical layer technology such as wireless and fiber optics.

- *NEXUS* (**NEXt bUS**)
  - Develop a common highly-capable, highly-scalable next generation avionics interconnect with the following features:
    - Transparently compatible with wired, fiber-optic, and RF physical layers
    - Scalable fault tolerant (microsecond detection/recovery latency)
    - Scalable bandwidth from 10 Kbps to 10 Gbps
    - Guaranteed real-time determinism with microsecond latency/jitter
    - 20% - 50% wire mass reduction
    - Low power (< 100mW per Gbps)
    - Light-weight



# NEXUS – Approach & Prototype



- NEXUS – next generation avionics bus
- High bandwidth: 1 to 10 Gbps
  - Low Latency: 1 to 10 us
  - Low Jitter: < 100 ns
  - Guaranteed determinism
  - Scalable and inherent fault tolerance
  - Parallel traffic classes of different QoS levels

NEXUS-SYS (System Layer)

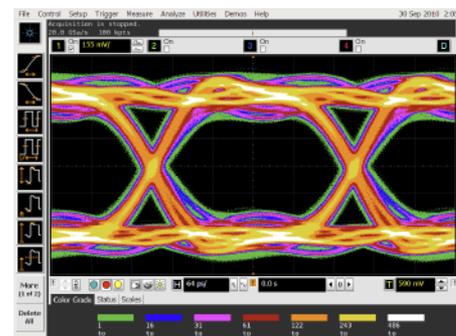
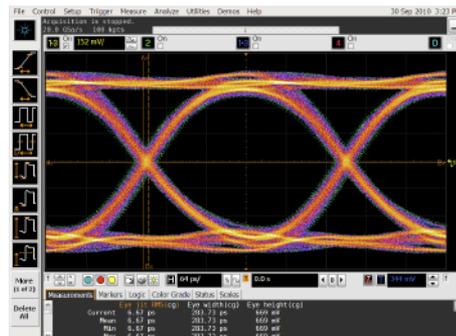
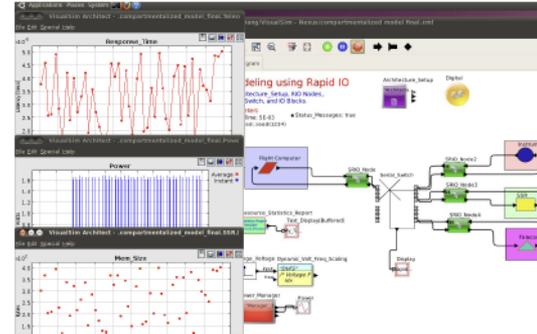
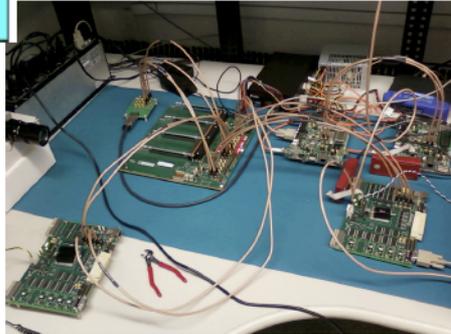
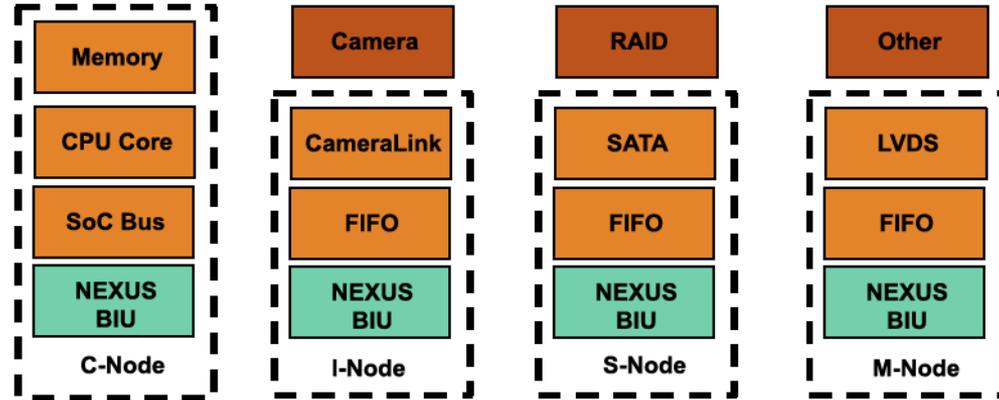
NEXUS-NET (Network Layer)

NEXUS-PA (Physical Abstraction Layer)

Wired Copper PHY

Wireless PHY

Wired Fiber  
Optical PHY



# MATCH – Motivation

## Mars Fetch Rover

### **Driver - Low Mass:**

- (1) MER Baseline: ~16 kg
- (2) Minimal Mass: < 10 kg

### **Core C&DH Functions:**

Computing/Control, Storage, Interfaces

### **Engineering Telemetry Collection:**

Cruise, Lander, and Rover stages

### **Motor Control and Drive:**

Brushed, brushless, and stepper motors

### **Low Power Sleep Mode**

### **General:**

- The mission architecture consists of Cruise stage, Lander stage, and Rover stage
- Technology Cut-off: TRL 6 by Sept. 2012
- Redundancy: Single string



# MATCH – Technical Approach

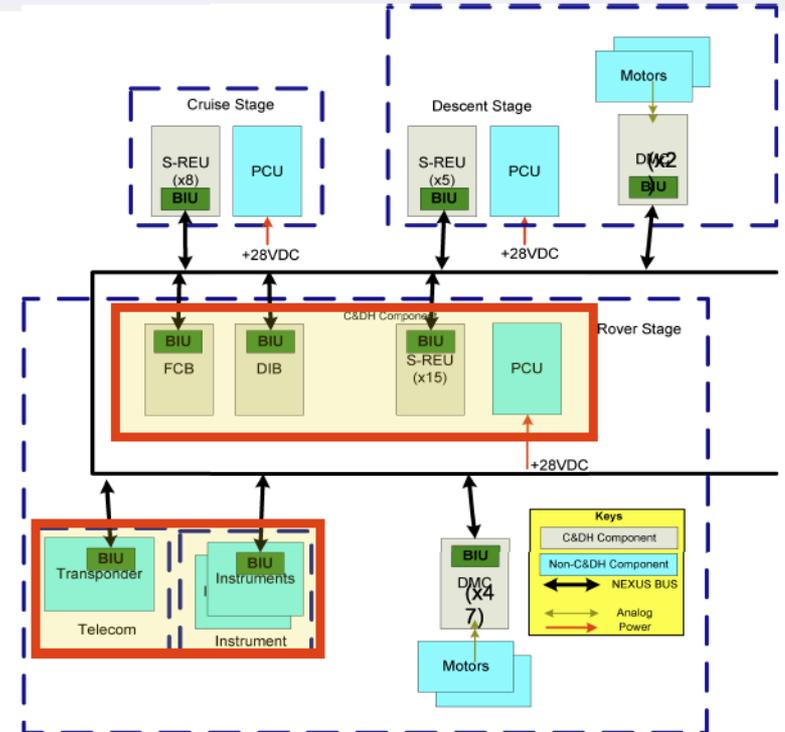


## Minimize the mass by

- Taking advantage of the key state-of-the-art SoC/FPGA technology
  - Highly integrated on-chip processing, storage, programmable fabric, and standard I/O drivers/receivers
- Increasing per board function density
- Removing the backplane with a high-speed backbone bus NEXUS
- Use cold-capable ASIC SiGe-REUs for collection of engineering telemetry
- Use distributed motor controller (DMC2)



## MATCH (Micro Avionics TeCHnology)



	Avionics Technology				
	MER	MSL	BroadReach Avionics	MRO-Lite	MATCH
<b>Mass (kg)</b>	16.03	21.85	15.28	18.51	8.05
<b>Power (w)</b>	31.84	69	55.6	60.3	14.3

## Limitations of Current Avionics Technology

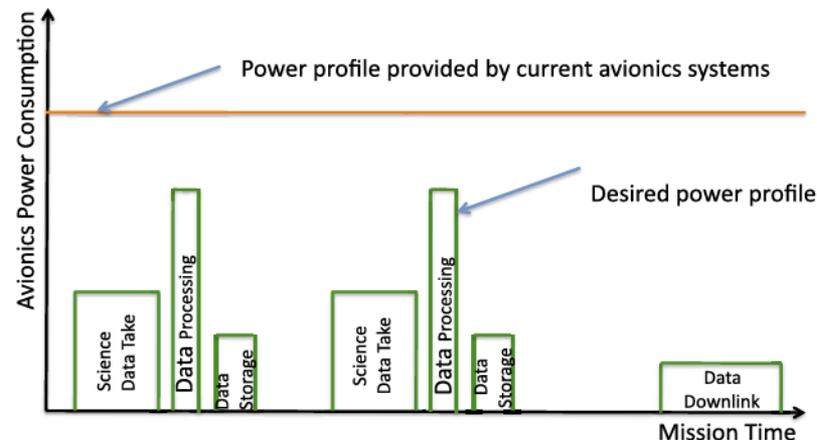
### 1. Lack of adaptive, fine-grained, and multiple avionics power operational modes

- Almost all existing avionics designs support merely on/off power modes at the box-level. Once powered on, it is never powered off, wasting significant amount of energy even during off-duty cycle

### 2. Lack of effective usage of inherent low-power features of modern space-grade microelectronics devices

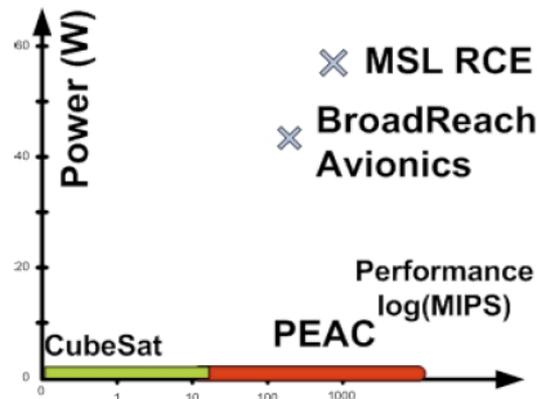
- Clock-gating and multiple-power-mode for microprocessors and variable-voltage-scaling for other parts

### 3. Lack of finer-granularity match between the mission operational mode (typically *bursty*) and the avionics operational mode (typically *flat*)



# PEAC – Objectives

- Develop and demonstrate a power-adaptive avionics technology  
**PEAC (Power Efficient Adaptive Computing)**
  - That will enable swarm of small spacecraft and deployable mini-payloads requiring low-power (tens of milli-watts to one-watt) typically powered only by primary batteries
- Develop a set of building blocks and a framework that can be easily configured and integrated to provide efficient power management and control of an avionics system
- Reduce the risk of low-power/mass avionics system for future small spacecraft missions



- Partition an avionics system into two separate *power domains* (PDs)
  - Always-on domain
    - Is always powered and provides minimal functions such as timer and CCSDS Critical Relay Command Handling, consumes minimal power of less than 100  $\mu$ W
  - At-times-changed domain
    - Is managed at runtime with multiple power profiles adapted to various mission operational modes
- Manage PDs via a centralized *PEAC-runtime* executive
  - Provides on-demand and fine-grained power scheduling and system control
  - Is capable of trading performance and power consumption
  - Stays in Always-on domain
  - Is protected by fault tolerance

