Jet Propulsion Laboratory
California Institute of Technology

Mission Data System

Proposal for MDS Project Engagement with the POST Project

Jet Propulsion Laboratory
California Institute of Technology

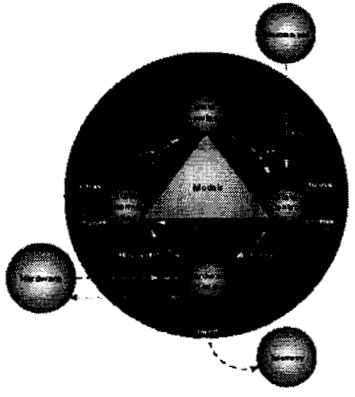
Legal Notice

Copyright 2003, by the California Institute of Technology. ALL RIGHTS RESERVED. United States Government Sponsorship Acknowledged. Any commercial use must be negotiated with the Office of Technology Transfer at the California Institute of Technology.

Content contributors:
Matthew Barry, PhD, Deputy Manager, IT Program Office
Daniel Dvorak, PhD, Deputy Architect, MDS
Kenny Meyer, Software Engineer, MDS
Jet Propulsion Laboratory
California Institute of Technology

Proposal Premises

- MDS process, tools, and tests are underway at JPL
- POST process, tools, and tests are underway at JSC
- MDS might benefit from additional adaptations and deployments in order to demonstrate technology maturation
- POST might benefit from additional tool and process support to the payload customer's hardware and software development activity



PAYLOAD OPERATIONS SUPPORT TEAM
• FIT TO FLY •

Outline

- **MDS Overview**
- Concept for POST Integration
- Technology Maturity
- Proposal

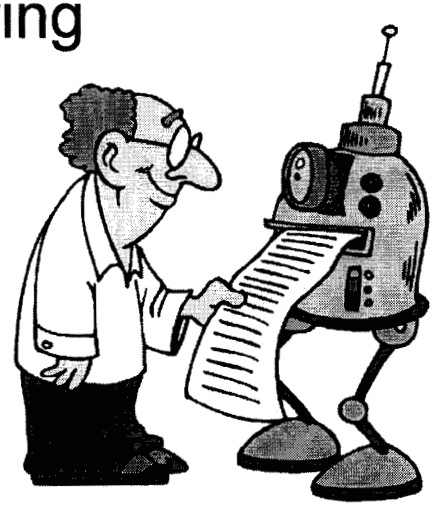
Pressures for Change

Challenging Future Missions

- ***In situ* exploration** of the solar system with vehicles on and under surfaces, in atmospheres, and on and around small bodies
- **Multi-vehicle coordination** for formation flying, rendezvous and docking, and so on
- **More complex observatories** with very demanding operational requirements
- An **interplanetary network** requiring multi-project coordination of resources scattered around the solar system

What We Need

- Highly reusable core software for flight, ground, and test
- Synergistic systems and software engineering
- Reduced development time and cost
- Improved development processes
- Highly reliable operations
- Increased functionality



Yes

The MDS Vision

A unified control architecture and methods
for flight, ground, and test systems
that enable missions
requiring reliable, advanced software

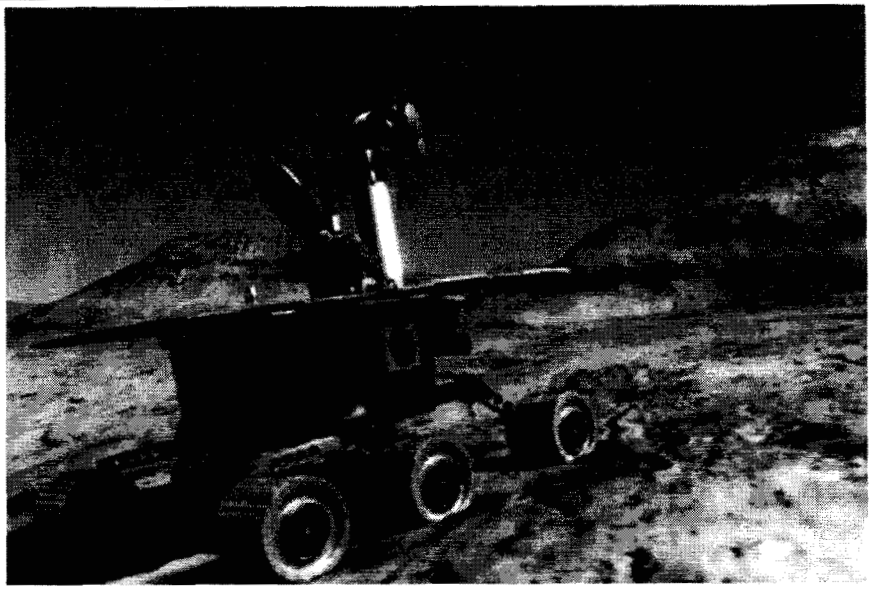
Jet Propulsion Laboratory
California Institute of Technology

Mission Data System

Problem Domain

Control of physical systems

- Mission information, control, and operations
- Broadly applicable to mobile and immobile robots that operate autonomously to achieve goals specified by humans
- Architecturally suited for resource-limited systems



Scope

- Flight systems
- Ground systems
- Test and simulation systems

Major Parts

MDS adopts a product line (multi-mission) approach that exploits commonalities across missions

- An information & control system architecture
- Reusable & adaptable framework software
- A systems engineering methodology

Jet Propulsion Laboratory
California Institute of Technology

State / Model / Goal Architecture

Estimators interpret measurement and command evidence to estimate state

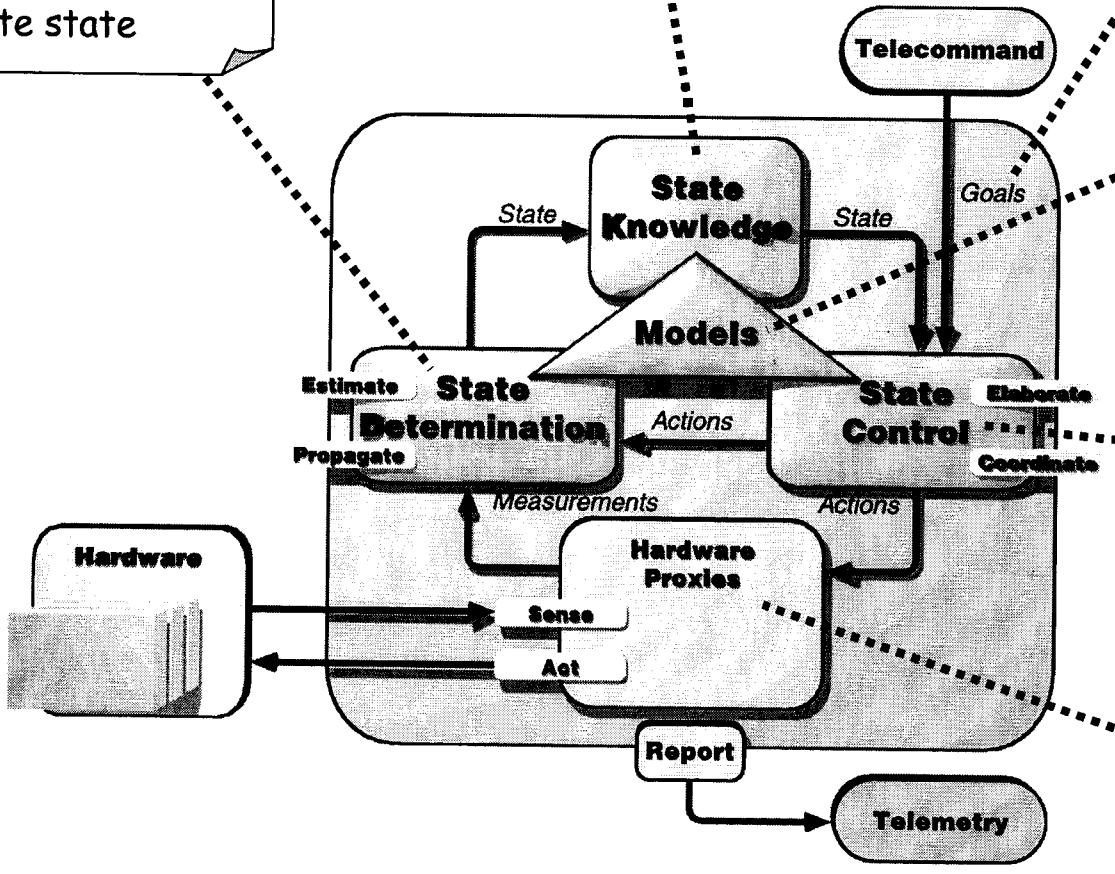
State variables hold state values, including degree of uncertainty

A goal is a constraint on the value of a state variable over a time interval

Models express mission-specific relations among states, commands, and measurements

Controllers issue commands, striving to achieve goals

Hardware proxies provide access to hardware busses, devices, instruments



JPL
Jet Propulsion Laboratory
California Institute of Technology

Bridging the Gap Between Systems and Software Engineering

Systems engineers follow a disciplined "state analysis", asking & answering questions such as these:

The state architecture bridges the gap through a shared set of architectural elements

Software engineers build the system by adapting a software framework having the <u>same</u> architectural elements

What do you want to achieve?
Move rover to rock. → ***Goal*** → Goal Network | Simulation | …

What's the state to be controlled?
Rover position relative to rock. → ***State Variable*** → State Knowledge | Hardware Proxy | Sequential Estimation

How do you know what that is?
Measure relative position with stereo camera. → ***Measurement*** → Measurement Package | Data Catalog | Graph Library

What does the stereo camera measure?
Distance to terrain features, light level, camera power (on/off), camera health. → ***Measurement Model*** → State Models | Coordinate Systems | SI Units | Naming

How do you control light level?
Wait until the sun is up. → ***State Model*** → Time Services | Linear Algebra | Event Logging | …

Where is sun relative to horizon?
…

Etc.

Cost Estimation

- With traditional systems engineering, software cost estimation has been more difficult because:
  - Requirements are expressed as "shall statements"
  - Work Breakdown Structure on prior projects typically mixes several engineering costs

- Systems engineering in MDS expresses requirements in terms of a small vocabulary:
  - Goals
  - Goal networks
  - Temporal constraints
  - State variables
  - State values
  - State effects models
  - Measurements
  - Measurement models
  - Commands
  - Command models
  - Estimators
  - Controllers
  - Hardware adapters
  - Resource allocations

These terms provide a much more detailed basis for cost estimation

Verification & Validation

- ***Leverage the state architecture***
  - Unifying concept of 'state' facilitates direct comparison between flight, ground, and simulation
  - Goals provide runtime pass/fail status (self checking!)
  - Models can be inspected/validated separately from application
  - Can simulate & test at 3 levels: state, functional, bit
  - Standard representation of state knowledge with built-in uncertainty
- ***Leverage the component architecture***
  - Check a deployment's as-built configuration against architectural rules
- ***Leverage the framework***
  - Frameworks are an important target for V&V, distinct from adaptation
  - Instrument the framework, enable/disable as desired
  - Design away some sources of error
- ***Leverage previous adaptations***

Elevating into Architecture

- Things that are "buried in the analysis" or "buried in the code" are hard to review, analyze, modify, and manage
- In many areas MDS makes things *explicit* and brings them out into the light of day. We call this "elevating into architecture". Examples:
  - State variables and behavior models are explicit design elements
  - State estimation is cleanly separated from control
  - State timelines organize both past and future state information
  - Components communicate only via connections
  - Architecture configuration is explicit, inspectable, and verifiable
  - Units of measurement are explicit and enforced
  - Initialization dependencies are explicit and enforced
  - Operation based on explicit, unambiguous constraints on state & time
  - Resources (power, etc) are managed as state allocations
- All this *explicitness* offers many opportunities for improving dependability through disciplined analysis and verification

Jet Propulsion Laboratory
California Institute of Technology

MDS Software Framework Packages

Application Services
Level 5

- Goal Elaboration Language
- State Query
- Data Visualization
- Simulation
- Component Scheduler

State Services
Level 4

- State Knowledge
  - -state variable
  - -state value
- Goal Achiever:
  - -estimator, controller
  - -Measurement, command
- Goal Network
- Hardware Adapter
- Graph State Variable

Complex Services
Level 3

- Components & Connectors
- Value History
  - -sampled history
  - -time-interval history
- Data Catalog
  - -collection, entry, event
  - -data product
- Data Transport
  - -sender, receiver
  - -session, request

Simple Services
Level 2

- Embedded web Server & client
- Event Log Facility
- Naming Services
- Time Services
- Data Mgmt Policy
- CCSDS File Delivery Protocol

Primitive Services
Level 1

- Initialization & Finalization
- Data Serialization
- Exception Classes
- Standard Utility classes
- Sequential Estimation
- Graph Library
- Physics Library:
  - -SI Units
  - -Coordinate systems
  - -Position, velocity & acceleration
- Math Library
  - -Linear algebra
  - -Probability dist.
  - -Polynomials
  - -6-DOF classes

OS Services
Level 0

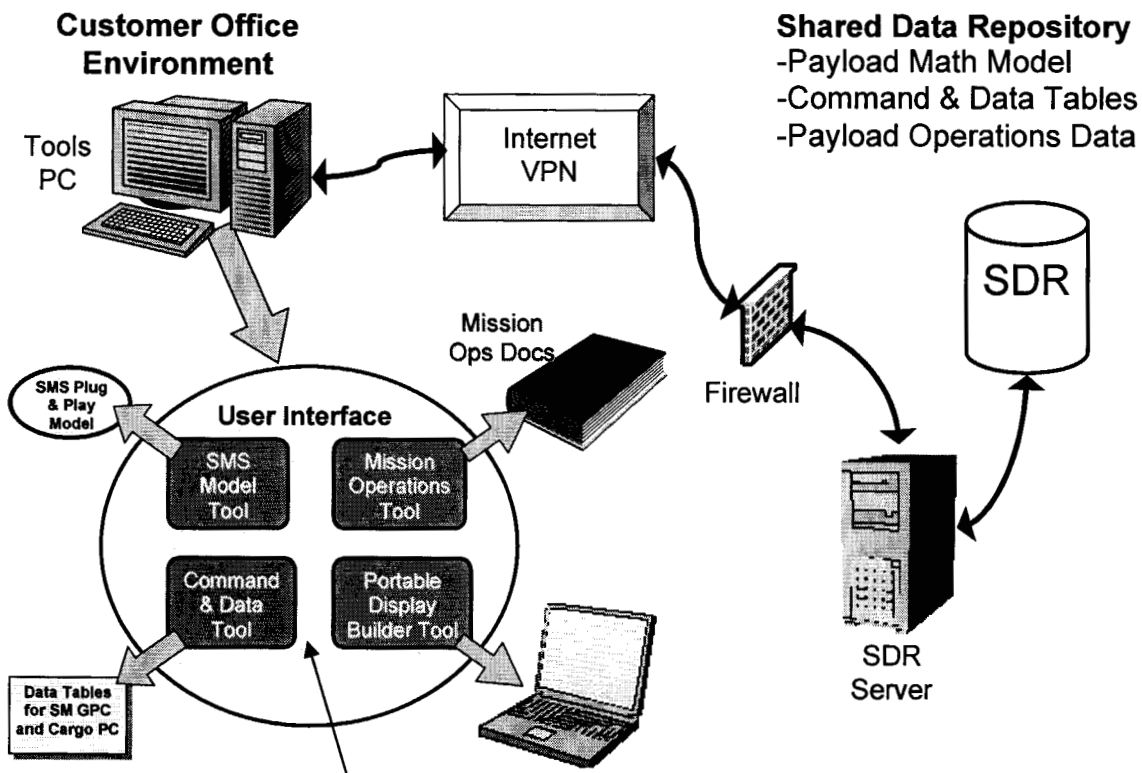
- C++ Standard Library
- Unit Testing Package
- Adaptive Communication Environment → Real Time Operating System

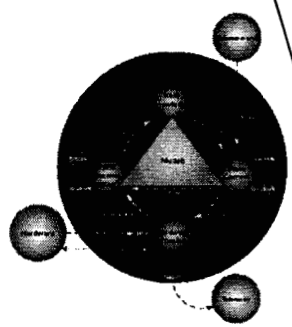
Outline

- MDS Overview
- **Concept for POST Integration**
- Technology Maturity
- Proposal

Jet Propulsion Laboratory
California Institute of Technology

MDS in the POST Environment

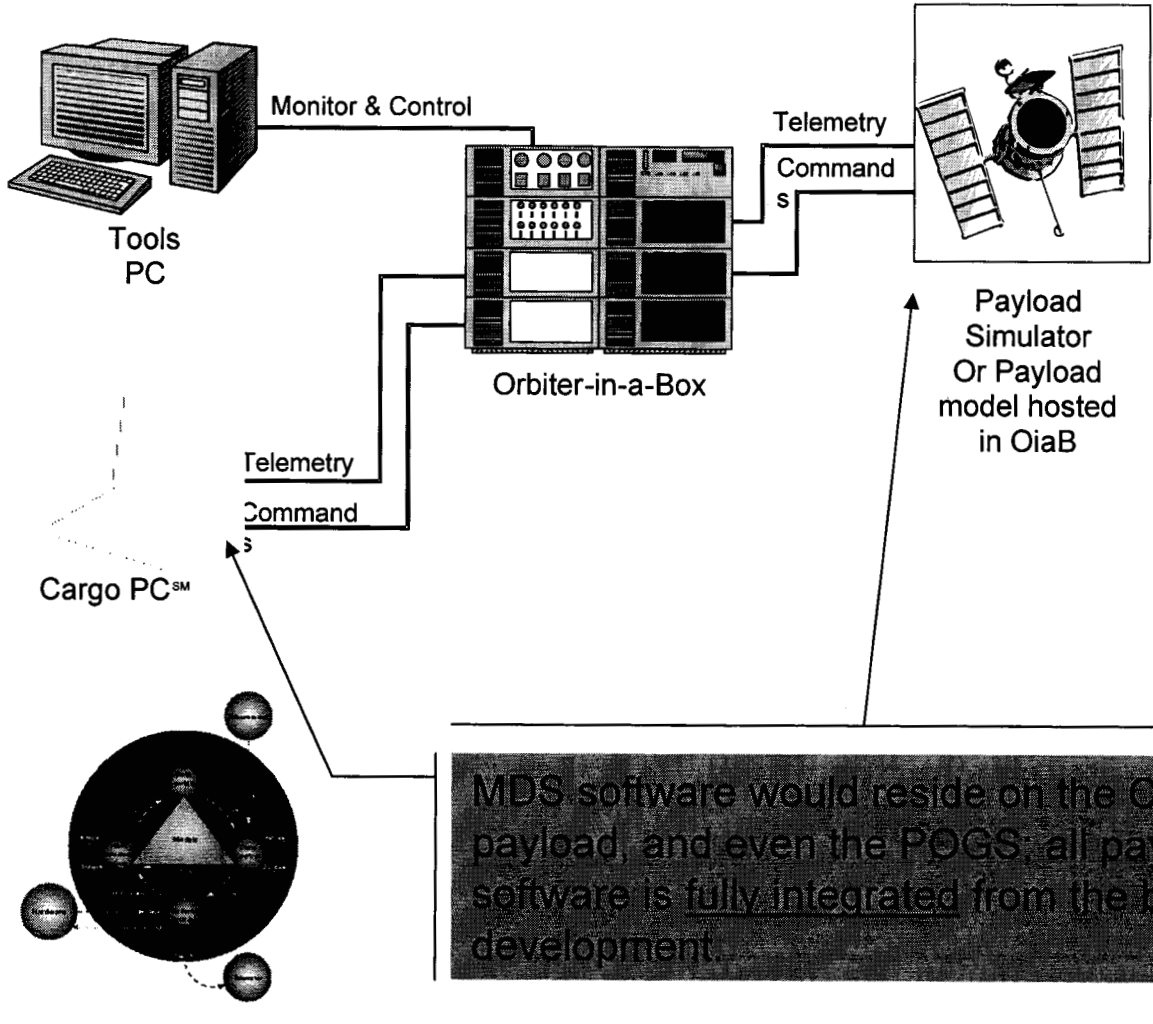




MDS complements the existing POST tools, and supplements them with an innovative state analysis process for hardware and software engineering; the on-going design content can live in the SDR

Jet Propulsion Laboratory
California Institute of Technology

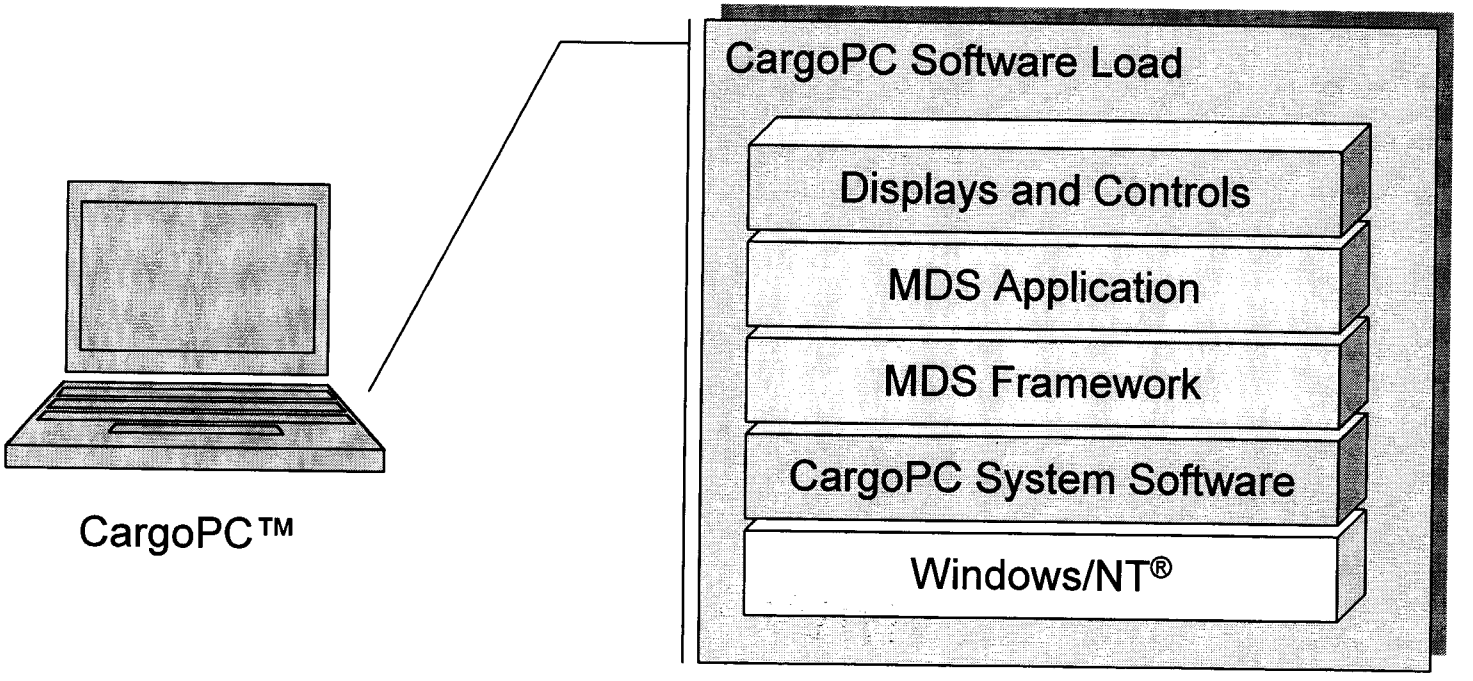
MDS in the POST Environment



MDS software would reside on the CargoPC, the payload, and even the POGS; all payload control software is fully integrated from the beginning of development.

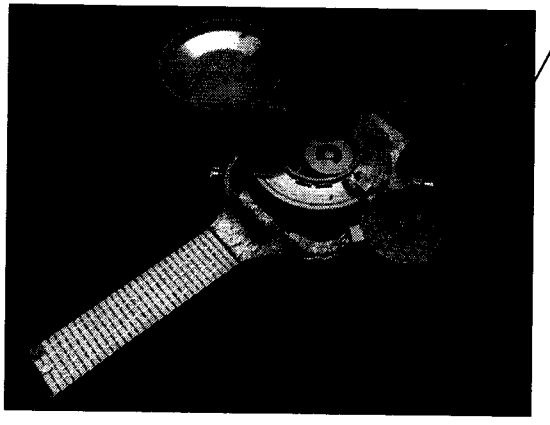
Jet Propulsion Laboratory
California Institute of Technology

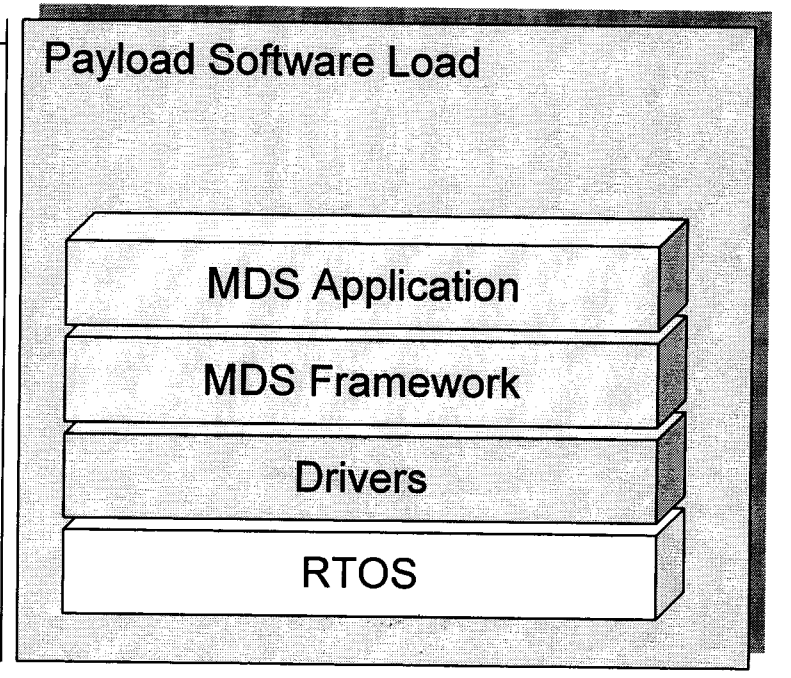
MDS Software Fit Within CargoPC



Jet Propulsion Laboratory
California Institute of Technology

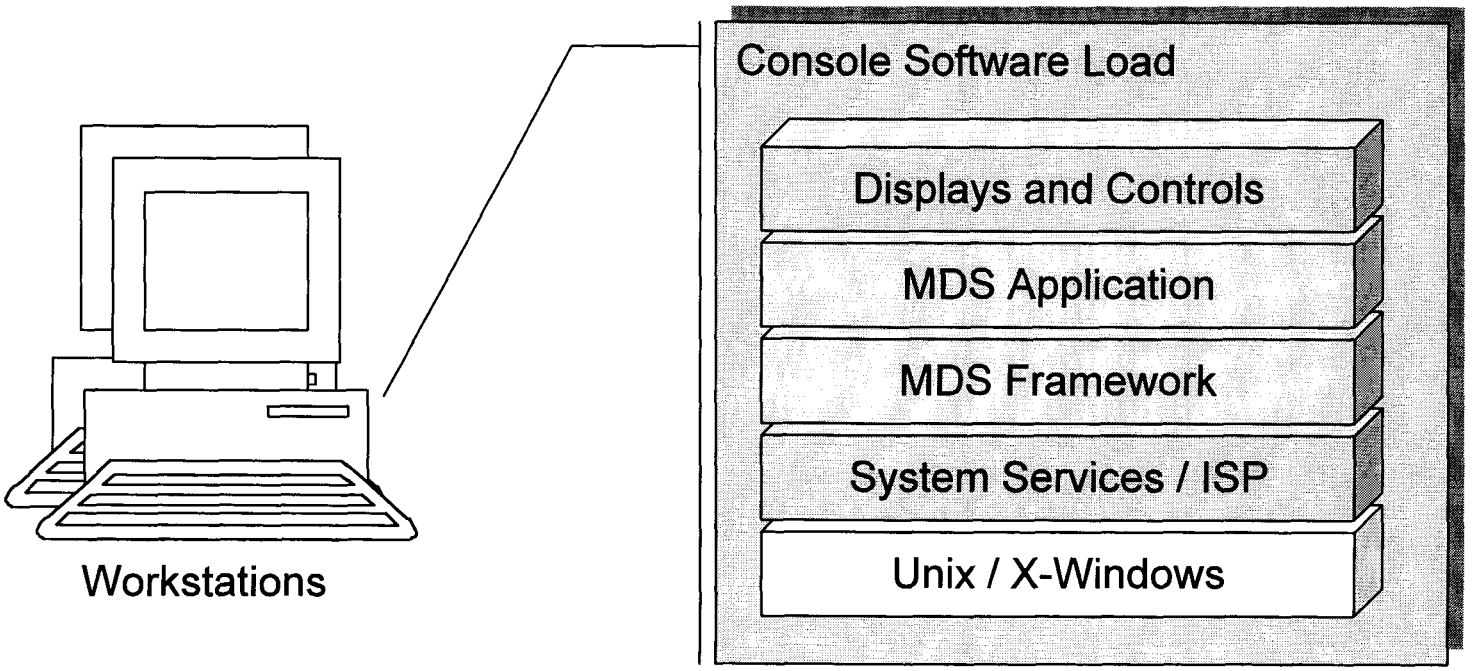
MDS Software Fit Within Payload





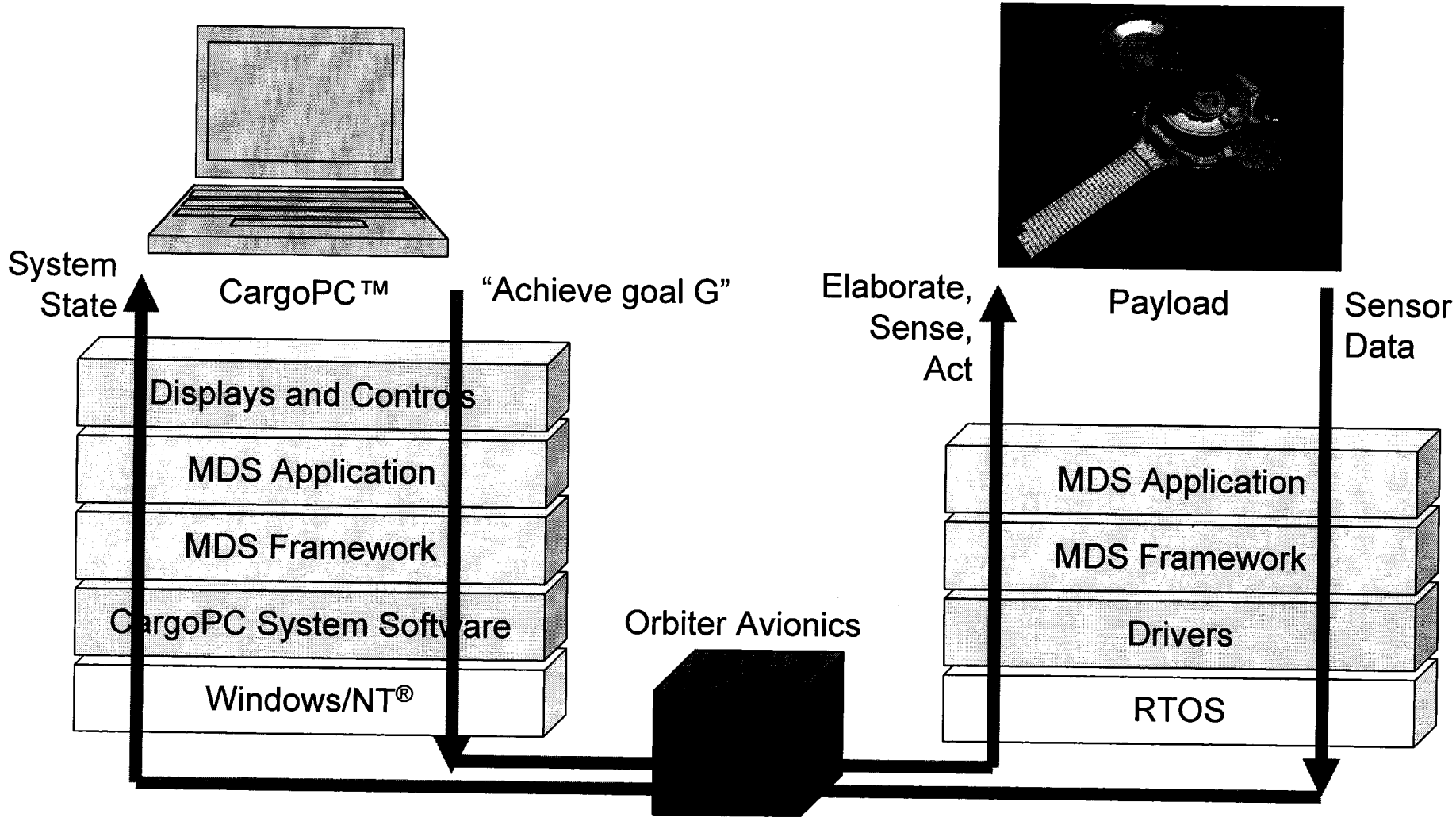
Jet Propulsion Laboratory
California Institute of Technology

MDS Software Fit Within MCC/PCC



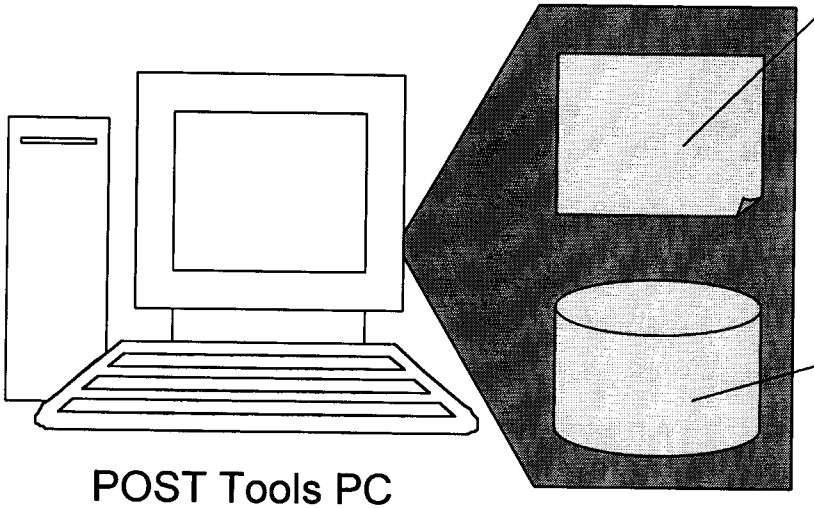
Jet Propulsion Laboratory
California Institute of Technology

Goal-Driven Control Loops



Jet Propulsion Laboratory
California Institute of Technology

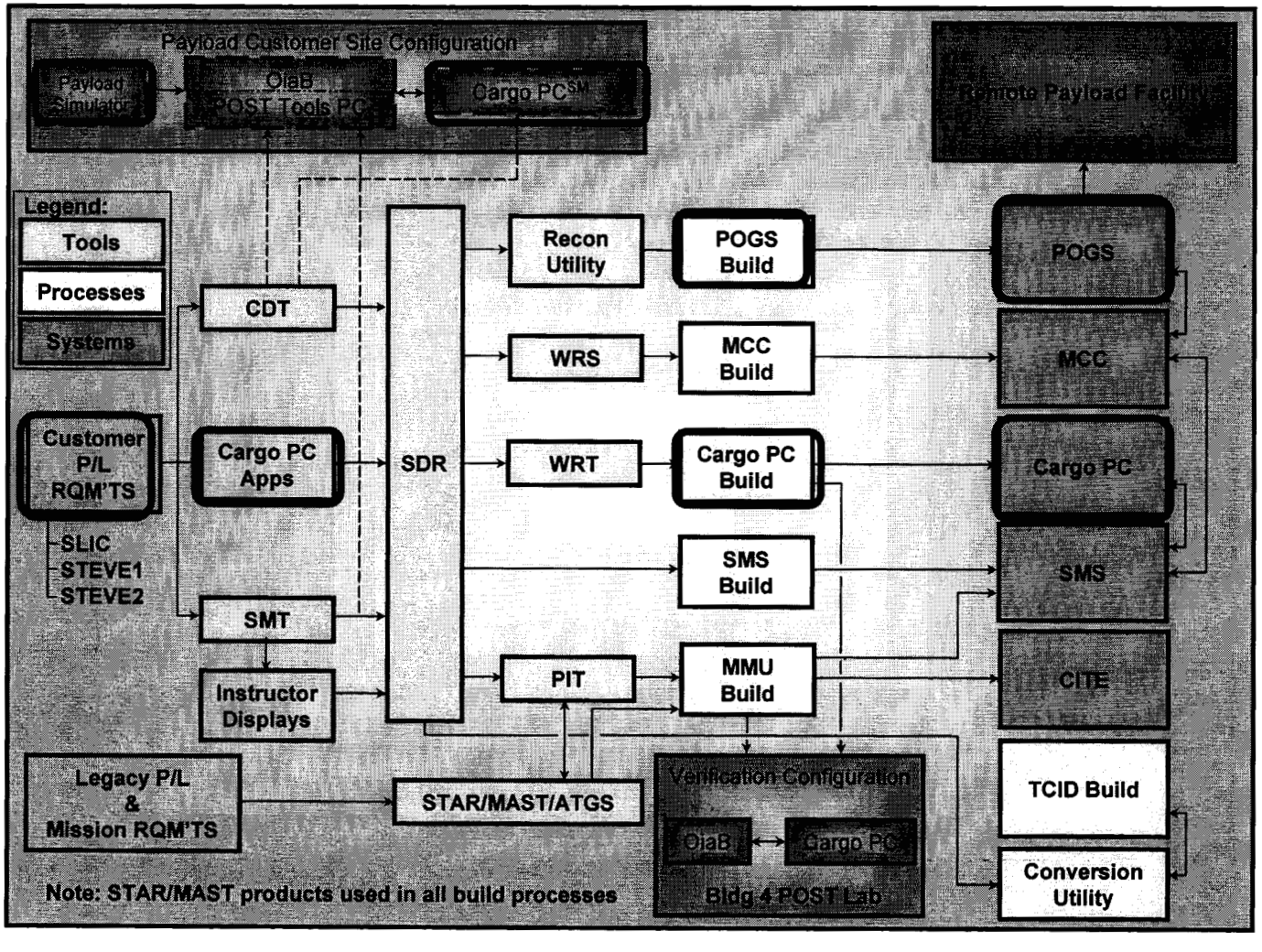
MDS Software Fit into PTPC



MDS Software Framework, providing several layers of architecture support and APIs; to be used by programmers along with Visual Studio or other tools

MDS State Database, capturing all of the results of the MDS state analysis process; to be used to support all development, verification and validation activities; browser GUI, hosted by PTPC database server

Potential MDS Footprint



MDS Footprint

Outline

- MDS Overview
- Concept for POST Integration
- **Technology Maturity**
- Proposal

Technology Readiness Levels

- NASA Management Instruction NMI-7100
  - Integrated Technology Planning
- The TRL concept offers a taxonomy for describing maturity
  - Categorization and evaluation of proposals
    - Research and technology development
    - Mission trade studies
  - Risk assessment and mitigation

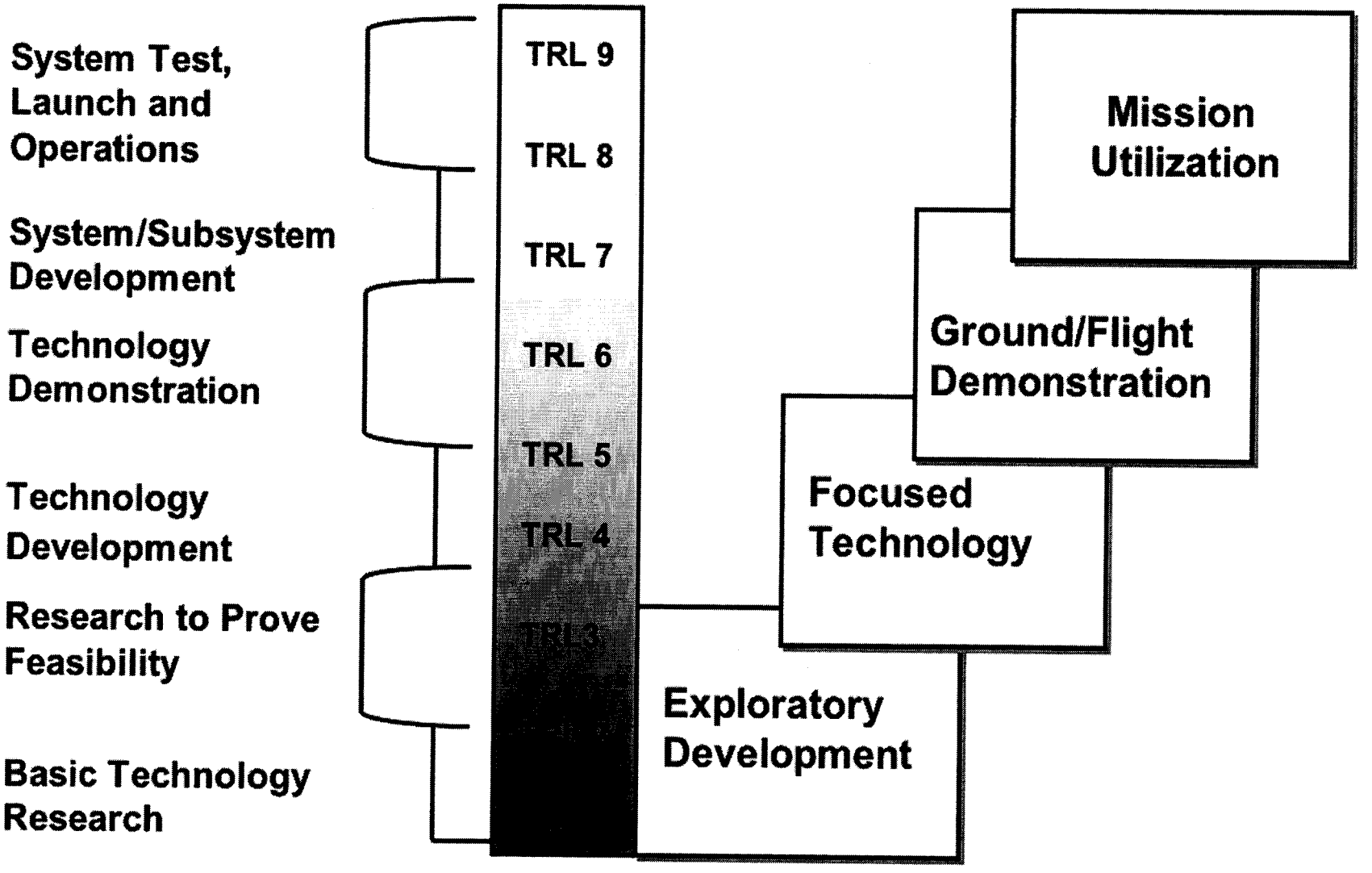
TRL Definitions

TRL 9	Actual system flight proven; successful mission operation
TRL 8	Actual system flight qualified through test and demonstration
TRL 7	System prototype demonstration in space environment
TRL 6	Model or prototype demonstration in relevant environment
TRL 5	Validation in relevant environment
TRL 4	Validation in lab environment
TRL 3	Experimental critical function; proof of concept
TRL 2	Technology concept formulation
TRL 1	Basic principals observed

Reference: John Mankins, "Technology Readiness Levels," NASA OSAT, 1995.

Jet Propulsion Laboratory
California Institute of Technology

TRL Model



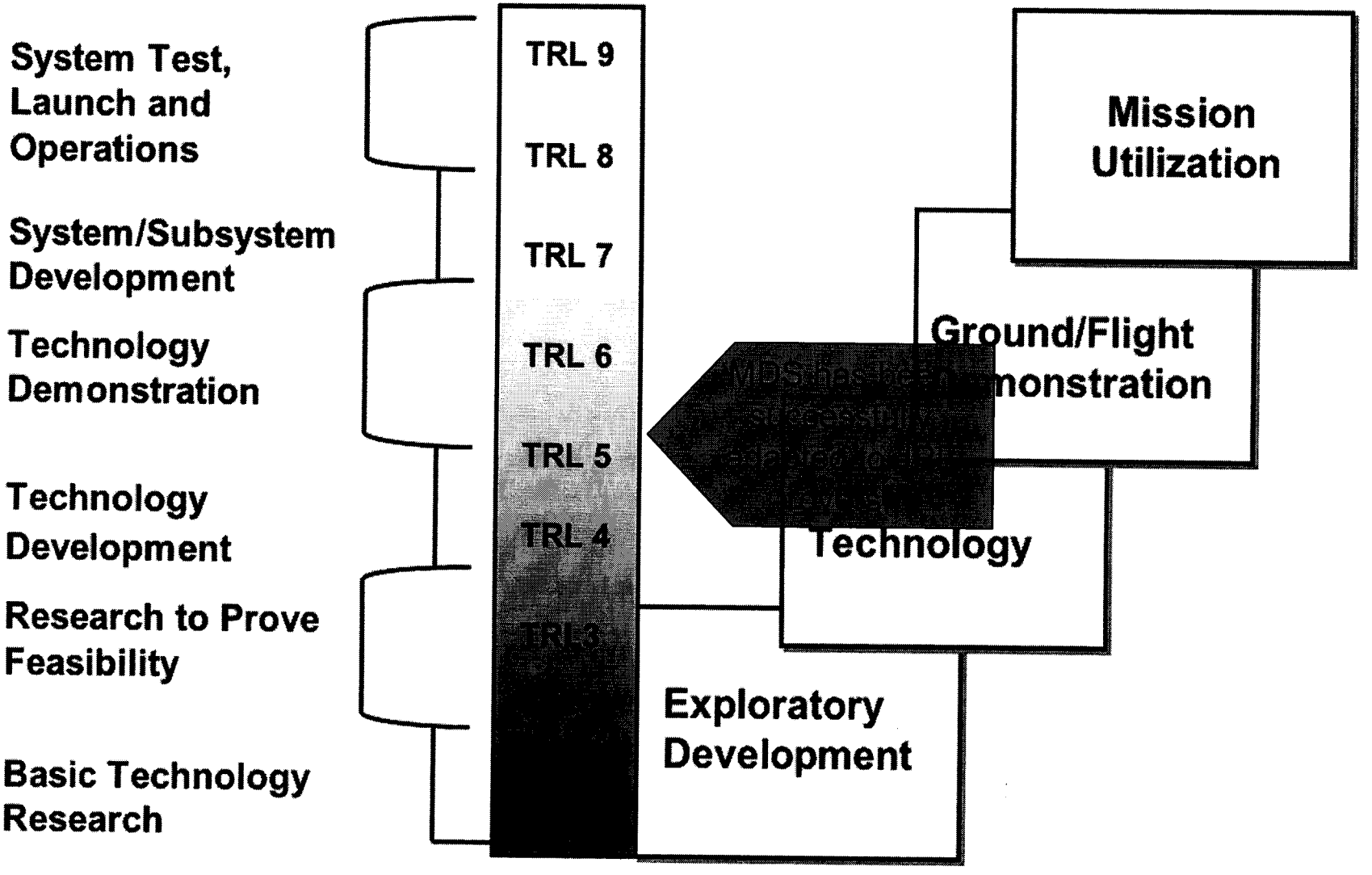
Reference: John Mankins, "Technology Readiness Levels," NASA OSAT, 1995.

Technology Infusion Platform

- As an architecture, MDS defines elements and interfaces and relationships that help structure system designs
- This includes for:
  - Goal planning and scheduling
  - Real-time scheduling
  - Commanding
  - State estimation
  - Fault diagnosis
  - Data management
  - Onboard data processing
  - Data transport
- The interfaces are technology-neutral; MDS doesn't prescribe a single solution; Different technologies can be used.

JPL
Jet Propulsion Laboratory
California Institute of Technology

MDS Current TRL



Outline

- MDS Overview
- Concept for POST Integration
- Technology Maturity
- **Proposal**

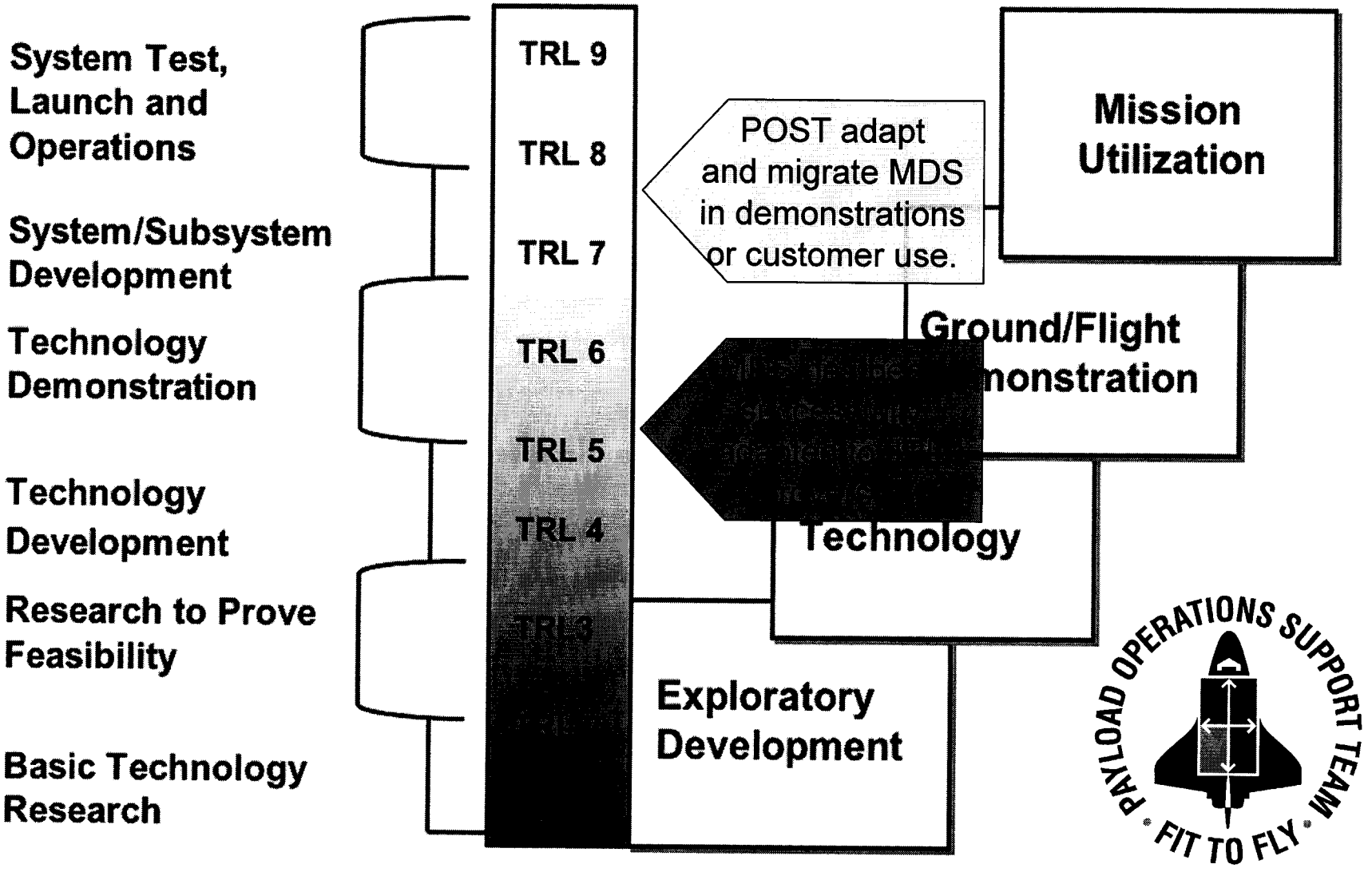
MDS Summary

- MDS is a significant program at JPL aimed at improving how we engineer, build, validate, and operate our missions
- MDS is a product-line approach based firmly on:
  - software architecture
  - systems engineering process
  - framework software
- MDS is baselined for the 2009 Mars Science Laboratory mission

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

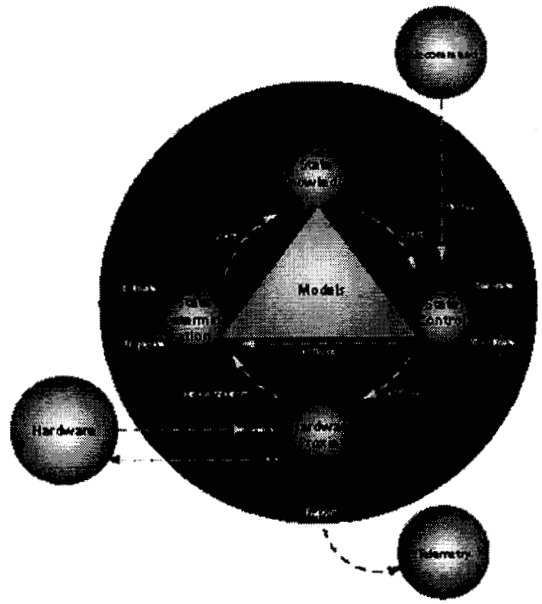
Jet Propulsion Laboratory
California Institute of Technology

POST Contributions to MDS



MDS Contributions to POST

- Multi-mission product line package supported by
  - System engineering methodology
  - Unified information and control system architecture
  - Reusable and adaptable software framework
- Unified architecture for Payload, POGS and CargoPC
- Software filling gap between CPCSS and PDB
- New features with little impact to current tool



MDS Collateral Potential

- The JPL relationship with CMU-West introduces:
  - Training in MDS approach to system engineering
  - State-of-the-art early design evaluation technology
  - Novel human-computer interface design technology
  - "Collaborative computer" technology

Carnegie Mellon

JPL
Jet Propulsion Laboratory
California Institute of Technology

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

Photo of MERBoard from Space.com credit IBM.

Proposal

- POST project consider merits of incorporating MDS products into POST tool suite
- USA, MOD and JPL approach SSPO for funding project
  - Identify integration points and requirements
  - Identify POST process changes
  - Training for all involved
  - Integration and adaptation of tools
  - End-to-end demonstration test
- Compare results with current POST end-to-end test results