Challenges Of Robotic Space Exploration

28 May 2003

The 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2003)

Peter R. Glück
Autonomy and Control Section
Jet Propulsion Laboratory
California Institute of Technology

Agenda

- JPL Missions and Software
- Control Systems
- Deep Space Requirements
- Deep Space Application Differences
- Coupling In Space Systems
- States and Models
- Deep Space Vehicle Control Software
- Deep Space Life Cycle Differences
- Challenges of Deep Space
- Mitigating the Risks
- Reducing the Cost
- Summary

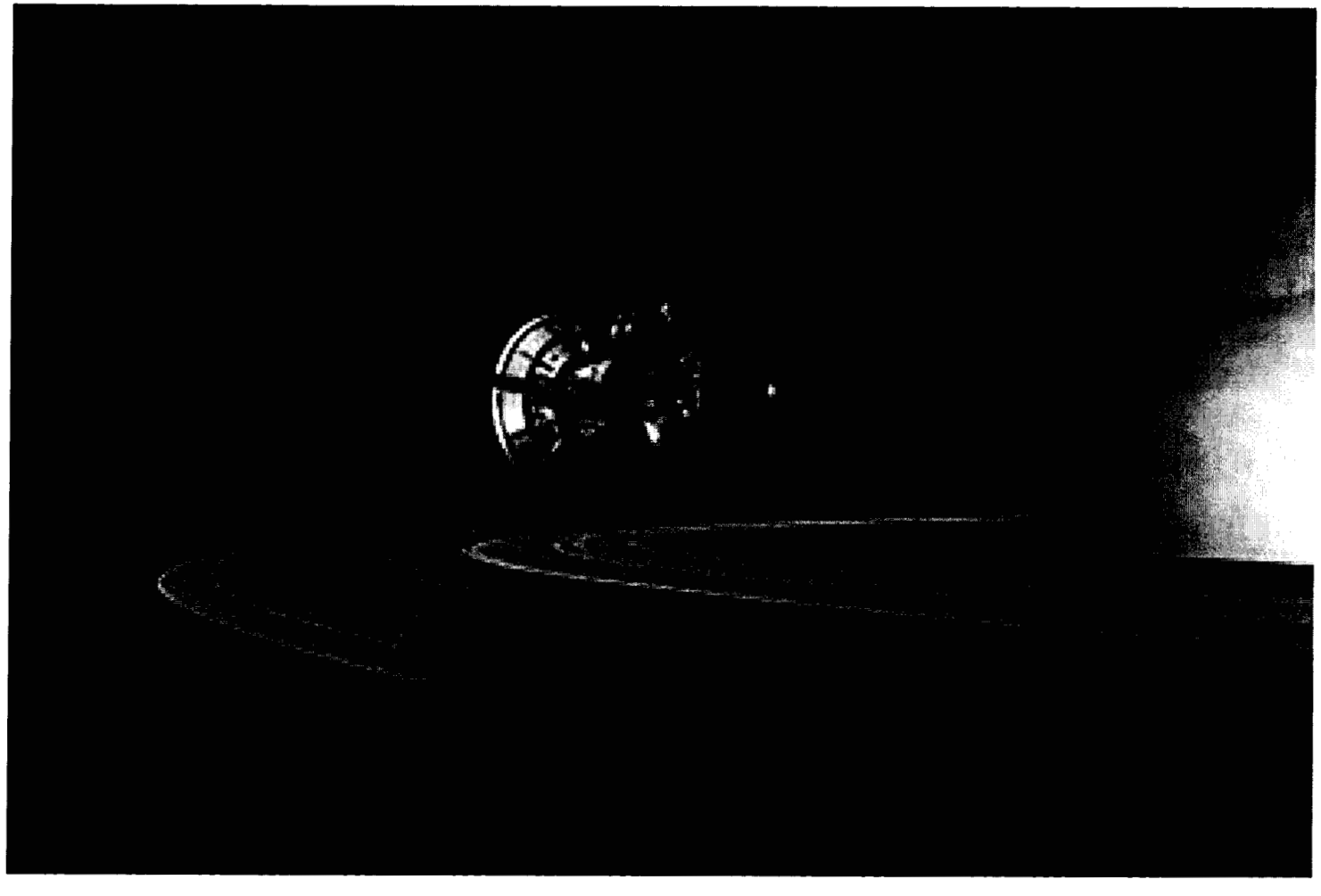
The JPL Mission

- **Expand the frontiers of space by conducting challenging robotic space missions for NASA**
  - Explore our solar system
  - Expand our knowledge of the universe
  - Further our understanding of Earth from the perspective of space
  - Pave the way for human exploration
- **Apply our special capabilities to technical and scientific problems of national significance**
- **Our mission is what we do to implement NASA's vision**

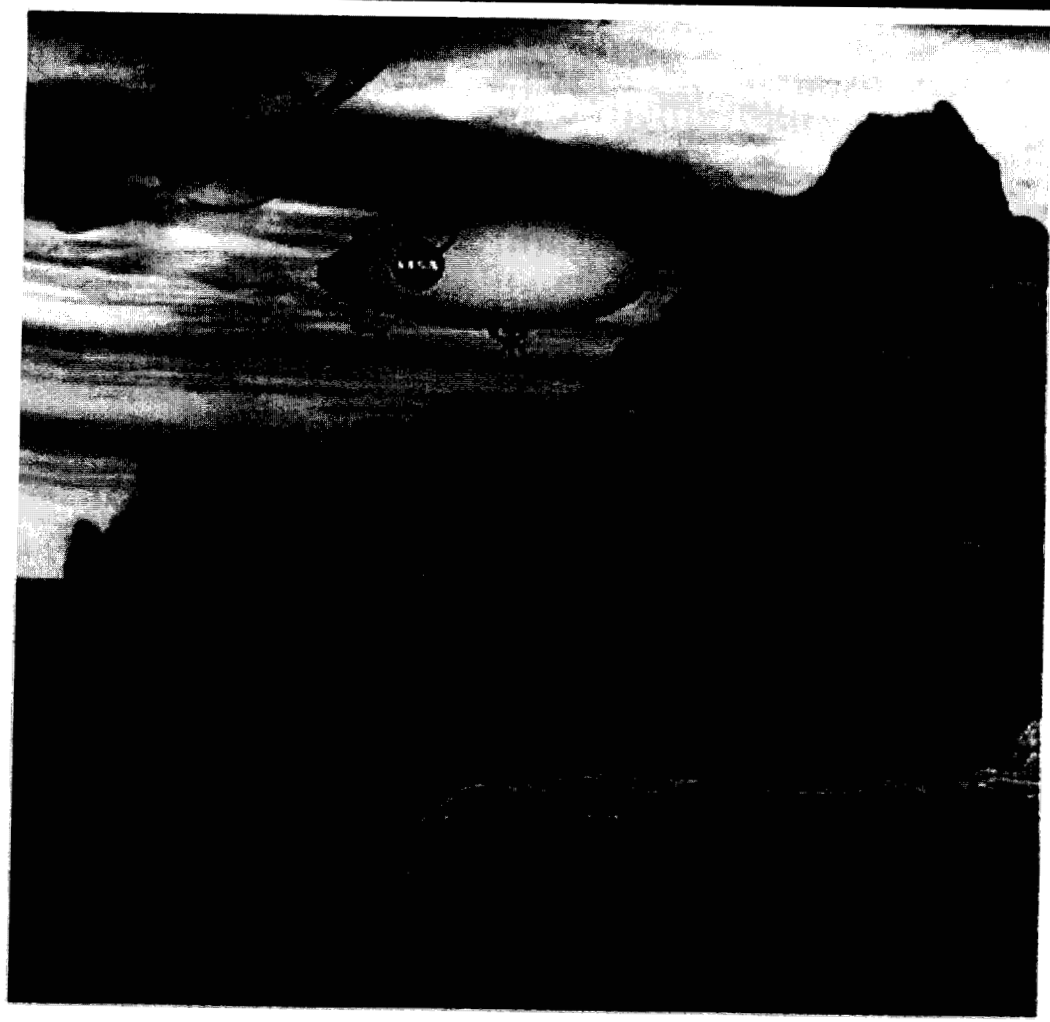
Characteristics of Deep Space Missions

- Explore in uncertain environments
- Conduct in situ science investigations
- Operate far from Earth
- Survive for decades (in some cases)
- Operate semi-autonomously

Cassini-Huygens Mission

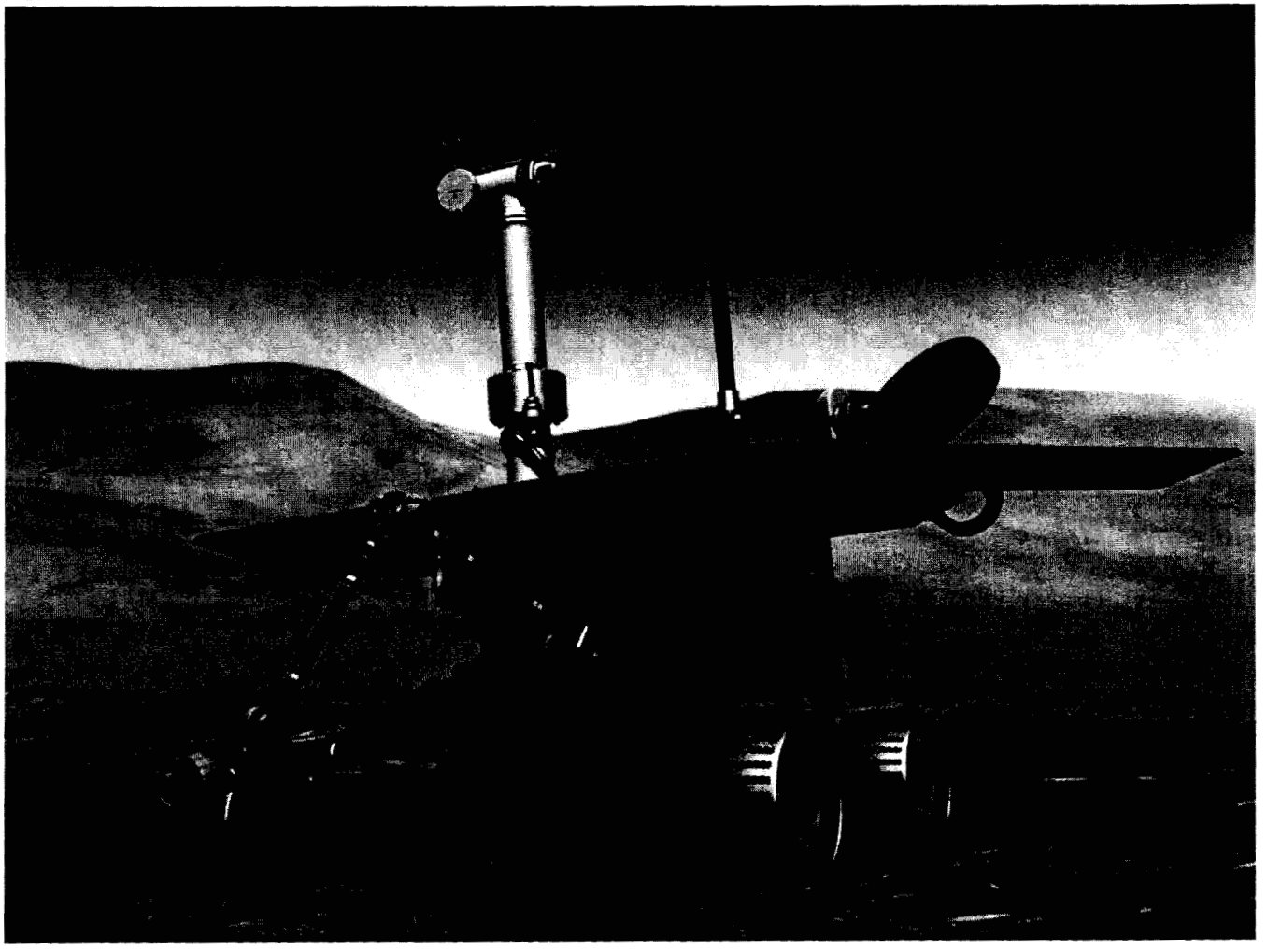


Titan Explorer



Hydrobot in Europa Ocean

Mars Exploration Rover



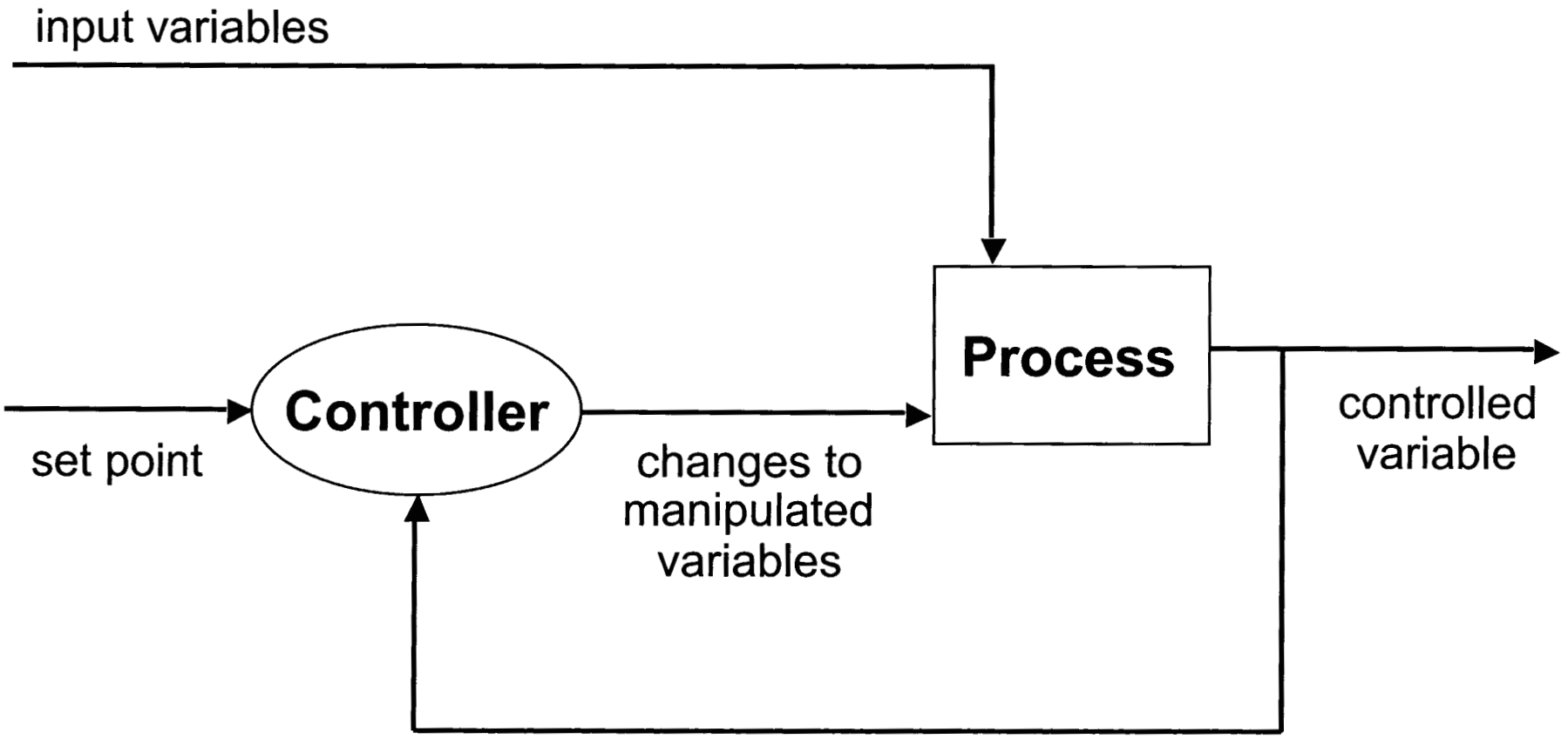
Types of JPL Software

- **Business Software**
  - Oracle, web-applications, MS Office
- **Engineering Software**
  - Drawing, CAD, CAE, software development tools, MS Office
- **Mission Software**
  - Ground Data System software
    - Command generation and transmission
    - Telemetry reception, distribution and analysis
  - Flight System Software
    - Vehicle (spacecraft) control software
    - Payload software

Control System Domain

- **Characteristics:**
  - Interacts with world via imperfect sensors & actuators
  - Designed for continuous operation
  - Real-time closed-loop control
  - Embedded systems, often
- **Examples:**
  - Petroleum refining
  - Pharmaceutical manufacturing
  - Nuclear power plant
  - Spacecraft control

Feedback Control System



* Diagram from "Software Architecture: Perspectives on an Emerging Discipline", Shaw & Garlan, 1996

Deep Space Requirements on Vehicle Control Software (1 of 4)

- **Initiate and maintain control of vehicle attitude**
- **Deploy vehicle assets if necessary**
  - Solar arrays
  - Antennae
  - Payload booms
- **Monitor vehicle health for serious equipment failures**
- **Autonomously execute critical mission activities**
  - Trajectory corrections
  - Orbit insertions
  - Entry, descent, and landing
  - Surface excursions
  - Scientific observations

Deep Space Requirements on Vehicle Control Software (2 of 4)

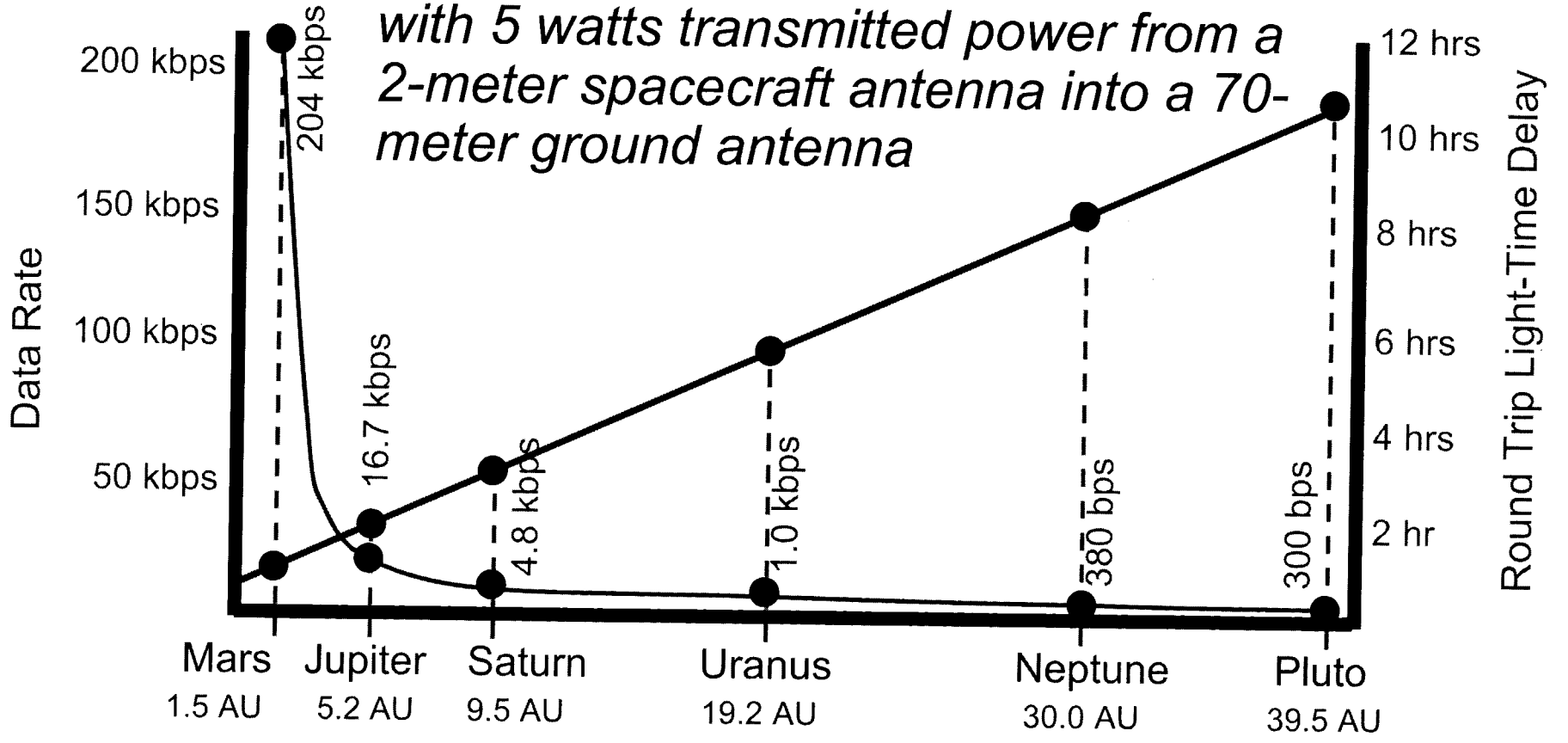
- **Protect vehicle assets from damage or exposure**
  - Optics
  - Thermal control surfaces
  - Delicate structures
  - Deployables
- **Notify operators of unrecoverable errors**
- **Keep the vehicle "safe" for weeks without interaction**
  - Maintain power with solar arrays
  - Protect consumable resources (e.g., fuel, cryogenic coolants, switch cycles)
  - Maintain an attitude conducive to Earth communications
  - "Worry" about not hearing from Earth (did my receiver fail?)

Deep Space Requirements on Vehicle Control Software (3 of 4)

- **Infrequent, scheduled communication**
  - The 'network' is not continuously available due to DSN constraints and spacecraft activities that compete for power and pointing.
- **Distance and Time Delay**
  - With a ~10 hour round-trip light-time delay to Pluto, it's impossible for operators on Earth to react to events in a timely manner.
- **Distance and Communication Rate**
  - With a data rate of ~300 bits/sec from Pluto, it isn't feasible to send all of the raw science data; prioritization/summarization is needed.
- **Distance and Pointing**
  - When transmitting, need to point antenna at where Earth will be when the signal arrives, not at where it is now. Vice versa for receiving.

Distance, Data Rate, Time Delay

Effect of distance on data rate for X-band RF communication with 5 watts transmitted power from a 2-meter spacecraft antenna into a 70-meter ground antenna



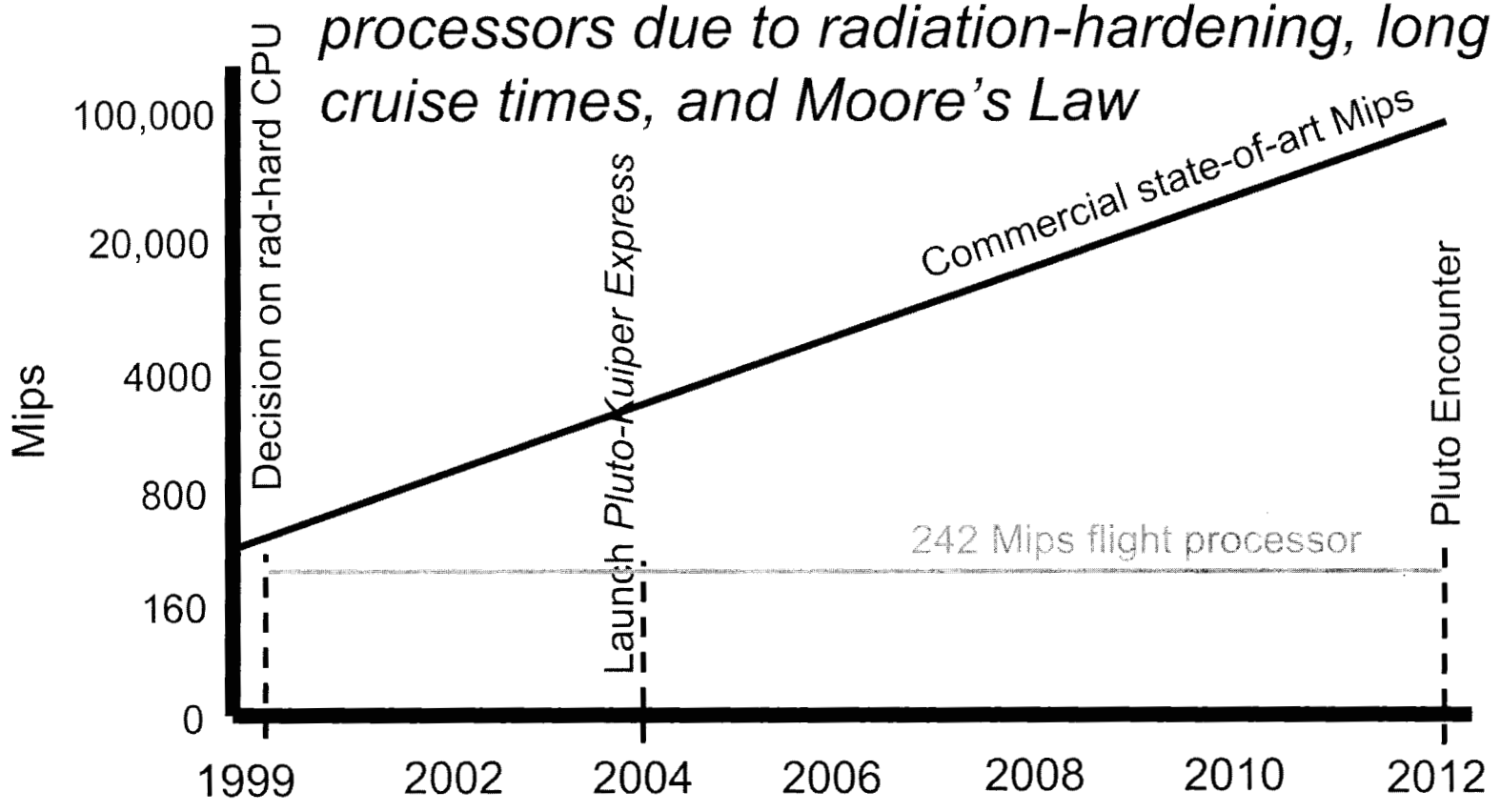
AU = Astronomical Unit = mean Earth-Sun distance

Deep Space Requirements on Vehicle Control Software (4 of 4)

- **Special Relativity and Time Dilation**
  - Though spacecraft velocity is a tiny fraction of lightspeed, navigation must take relativistic effects into account.
- **Limited Flight Processor and Memory**
  - Radiation-hardened flight processors are years behind mainstream commercial processors. Flight software must be frugal with CPU cycles and memory.
- **Cruise Time and Moore's Law**
  - The disparity between flight and ground processing abilities grows with every year of cruise time.
- **Limited Resources and Tight Coupling**
  - In a resource-limited system, 'everything affects everything'.

Computing: Flight vs. Ground

Gulf between flight processors and ground processors due to radiation-hardening, long cruise times, and Moore's Law



When spacecraft reaches Pluto in 2012, running a 242 Mip processor, desktop computers will be running at 100,000 Mips!

Deep Space Application Differences (1 of 2)

Attribute	Consumer	Terrestrial Embedded	Flight System for Earth Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
Consequence of Software Crash	User inconvenience	User inconvenience; possible loss of business in severe cases	Loss of service and/or replacement satellite required	Loss of science data and/or loss of mission if mission critical events are compromised
Duration of Unattended Operation	Minutes	Hours	Hours	Weeks
Test Method	Beta-test with volunteer users on common platforms	Test on plentiful hardware	Formal test on rare, expensive and complex hardware	Formal test on rare, expensive and complex hardware
Development Timescale	Months	Months to a year	Years	Years
User Interaction Timescale	Seconds	Seconds	Hours	Days
User Feedback Delay	Milliseconds	Milliseconds	Seconds	Minutes to hours
Autonomous Operation	Low	Low	Moderate	High

Deep Space Application Differences (2 of 2)

Attribute	Consumer	Terrestrial Embedded	Flight System for Earth Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
Interfaces	Simple: PC, keyboard, mouse, monitor	Simple: Keypad, transceiver, microphone, speaker	Complex: (usually) single payload [transceiver], vehicle control sensors and actuators [star trackers, gyros, sun sensors, thrusters power switches, heaters, temperature sensors], communications hardware [RF transmitters & receivers, data buses], launch vehicle	Complex: **multiple** payloads [cameras, spectrometers, magnetometers, radars], vehicle control sensors and actuators [star trackers, gyros, sun sensors, thrusters power switches, heaters, temperature sensors], communications hardware [RF transmitters & receivers, **recorders**, data buses], launch vehicle
Safety Considerations	None	None	Capability to achieve and maintain a safe state for several hours	Capability to achieve and maintain a safe state for several **weeks**

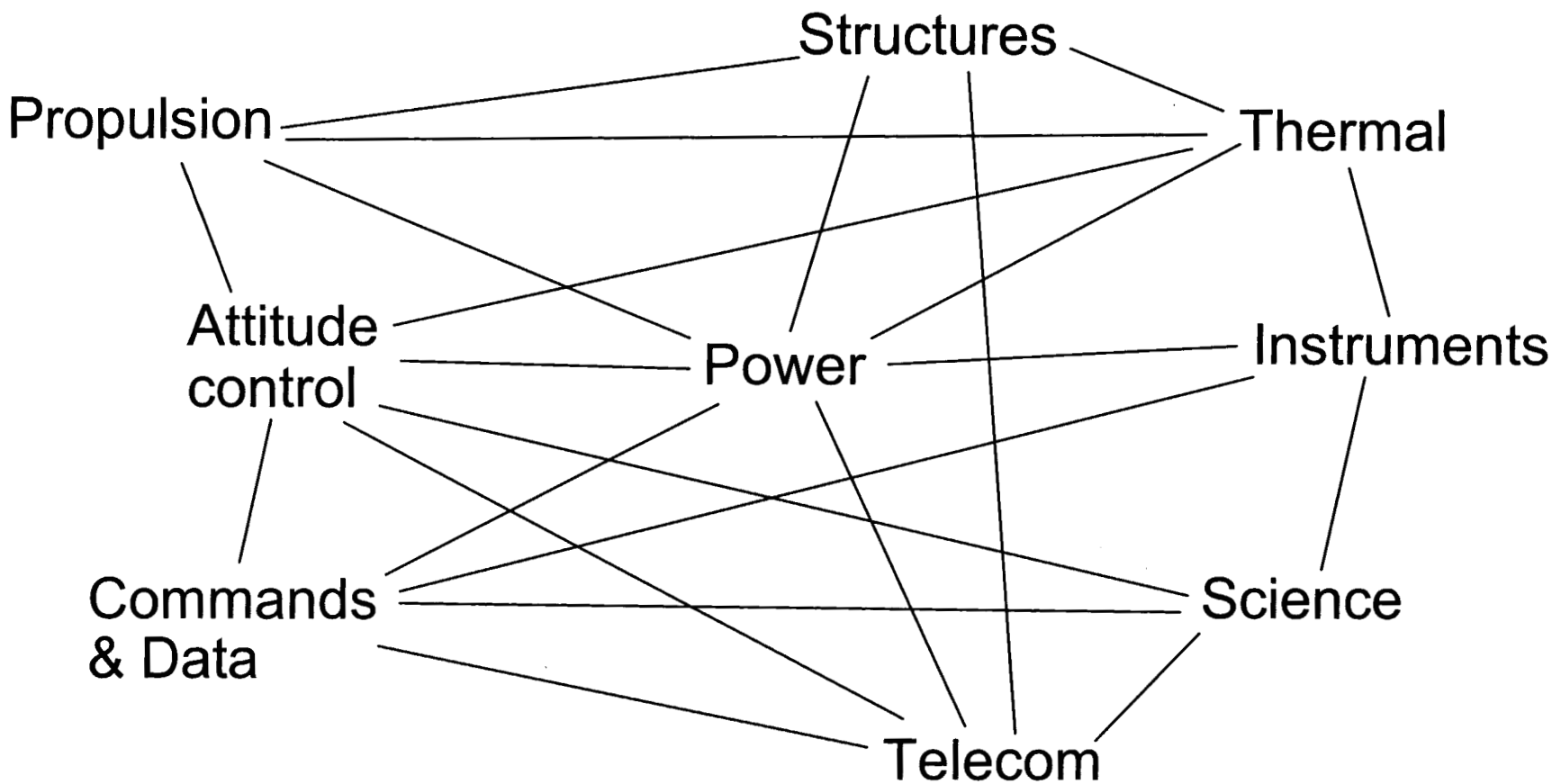
Coupling In Space Systems (1 of 3)

- **The world of side-effects**
  - Turning on a disk drive has the following side effects:
    - It reduces available power
    - It causes heating
    - It causes vibration
    - It causes electromagnetic radiation
    - It imparts rotational torque
    - It stabilizes orientation around axis of rotation
  - In a server room on Earth, these side effects are negligible
  - In a spacecraft, every one of these side effects is significant and must be managed!

Coupling in Space Systems (2 of 3)

- **Complex couplings arise from physics and design**
  - Amount of mass launched determines a big mission cost
  - Therefore, minimize size of batteries, size of solar panels, amount of memory, articulation mechanisms, shielding, smaller antenna, low-power transmitter, etc
  - That means:
    - Slower CPU and busses and less memory
    - Can't drive and transmit concurrently
    - Can't run heaters while firing thrusters
    - Can't independently point camera and antenna
    - Lower signal-to-noise ratio, so lower data communication rates, so science downlink is limited
    - Must hold reserve power for surviving the night

Coupling in Space Systems (3 of 3)



** Some domains of concurrent design in JPL's Project Design Center*

Couplings in a Mars Rover

- **Consider a home heating system:**
  - It is designed as a thermal subsystem that has complete responsibility for estimating and controlling temperature
- **However, on a Mars Rover, thermal control:**
  - Competes for power with driving, science, telecom, etc
  - Is affected by heating from nearby powered-on devices
  - Is affected by position of Sun relative to rover
  - May produce electromagnetic interference that precludes use of certain science instruments
  - If temperature sensor fails, must rely on thermal model
  - If heater fails, must turn on nearby devices for heating effect
- **Observation: The very concept of self-contained thermal control falls apart because it rests on an assumption of loose coupling**

Example Spacecraft States

- **Dynamics**
  - Vehicle position & attitude, gimbal angles, wheel rotation, …
- **Environment**
  - Ephemeris, light level, atmospheric profiles, terrain, …
- **Device status**
  - Configuration, temperature, operating modes, failure modes, …
- **Parameters**
  - Mass properties, scale factors, biases, alignments, noise levels, …
- **Resources**
  - Power & energy, propellant, data storage, bandwidth, …
- **Data product collections**
  - Science data, measurement sets, …
- **Data management policies**
  - Compression/deletion, transport priority, …
- **Externally controlled factors**
  - Space link schedule & configuration, …

Example Spacecraft Models

- **Relationships among states**
  - Power varies with solar incidence angle, temperature, & occultation
- **Relationships between measurement values and states**
  - Temperature data depends on temperature, but also on calibration parameters and transducer health
- **Relationships between command values and states**
  - It can take up to half a second from commanding a switch to full on
- **Sequential state machines**
  - Some sequences of valve operations are okay; others are not
- **Dynamical state models**
  - Accelerating to a turn rate takes time
- **Inference rules**
  - If there has been no communication from the ground in a week, assume something in the uplink has failed
- **Conditional behaviors**
  - Pointing performance can't be maintained until rates are low
- **Compatibility rules**
  - Reaction wheel momentum cannot be dumped while being used for control

Deep Space Vehicle Control Software Architecture (1 of 2)

- **Real-time**
  - 1 to 10 Hz control of spacecraft attitude
  - Processor interrupts or polling for
    - Attitude sensor updates
    - Data bus events
    - Uplink/downlink servicing
  - Priority-based multi-tasking
    - High-priority control and safety tasks get time when they need it
    - Low-priority data processing tasks take remaining time available
    - Task communication via interprocess communication mechanisms

Deep Space Vehicle Control Software Architecture (2 of 2)

- **Robust**
  - Limited data sharing
    - Use of global variables across tasks is discouraged
    - Use of pointers across tasks is discouraged
    - Data usually passed between tasks by value
  - Protect shared information
    - Semaphores and task locks
  - Memory partitions via the operating system
    - Limits overrunning data buffers and corrupting other tasks
  - Self-monitoring
    - Must recover control quickly in the event of lock-ups or crashes
  - Limited hardware access
    - Centralized and controlled access through a single interface

Deep Space Life-Cycle Differences (1 of 4)

Life-Cycle Phase	Consumer	Terrestrial Embedded	Flight System for Earth Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
Concept	Product idea and/or customer feedback	Product idea and/or customer feedback	Government or commercial objective	Scientific objective
Requirements	• Developed through developer and customer discussions over days or weeks	• Developed through developer and customer discussions over days or weeks	• Rigorously developed to meet high-level sponsor or institutional criteria over many months • Must consider payloads, Earth-orbital environment, mission duration, frequency of contact, and operations staffing and budget	• Rigorously developed to meet high-level sponsor or institutional criteria over many months • Must consider **multiple payloads, multiple environments [launch, cruise, planetary], critical mission activities**, mission duration, frequency of contact, and operations staffing and budget

Deep Space Life-Cycle Differences (2 of 4)

Life-Cycle Phase	Consumer	Terrestrial Embedded	Flight System for Earth Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
High-Level Design	• Developed from requirements over days or weeks • Some prototyping	• Developed from requirements over weeks • Some prototyping	• Developed from requirements over months • Little prototyping • Formally reviewed by peers, management and customers prior to detailed design	• Developed from requirements over months • Little prototyping • Formally reviewed by peers, management and customers prior to detailed design
Detailed Design	• Weeks by a few individuals	• Months by a few individuals to small teams	• Months to years by small teams with detailed documentation and peer reviews	• Years by small to large teams with detailed documentation and peer reviews
Implementation	• Weeks by a few individuals	• Months by a few individuals to small teams	• Months to years by small teams with detailed documentation	• Months to years by small to large teams with detailed documentation

Deep Space Life-Cycle Differences (3 of 4)

Life-Cycle Phase	Consumer	Terrestrial Embedded	Flight System for Earth Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
Unit Test	• Days to weeks	• Weeks	• Weeks to months, usually with similar or emulated hardware	• Weeks to months, usually with similar or emulated hardware
Hardware Integration	• Days to weeks on mature platforms	• Weeks on (typically) mature hardware	• Months on often-new prototype or engineering model hardware	• Months on often-new prototype or engineering model hardware
System Test	• Not applicable (standalone application)	• Weeks on (typically) mature hardware	• Months on the vehicle in various environments (static testing, dynamic testing, thermal-vacuum testing)	• Months on the vehicle in various environments (static testing, dynamic testing, thermal-vacuum testing)
Deployment	• Via internet or disk with user interaction	• Via download or programming support equipment with technician interaction	• Installed prior to launch • Requires built-in patch and/or load capabilities through RF link with operator interaction	• Installed prior to launch • Requires built-in patch and/or load capabilities through RF link with operator interaction

Deep Space Life-Cycle Differences (4 of 4)

Life-Cycle Phase	Consumer	Terrestrial Embedded	Flight System for Earth Orbit	Flight System for Deep Space
Example	*Web Browser*	*Cellular Phone*	*GPS Satellite*	*Deep Space One*
Operation	• Real-time interaction with user via keyboard/mouse and monitor	• Real-time interaction with user via keypad, microphone, and speaker	• Real-time interaction with user via RF link • Non-real-time execution of timed commands • Analysis of health and safety information by user • Real-time transmission of engineering and payload information	• **Delayed interaction** with user via RF link • Non-real-time execution of **sophisticated command sequences** • Often **unobservable execution of critical events** [orbit insertion; entry, descent and landing; night-side science] • **Storage of engineering and scientific observations** • Analysis of health and safety information by user • **Playback** and transmission of scientific information to customer

Challenges of Deep Space (1 of 2)

- **Robustness**
  - Vehicle must survive hostile environments for weeks relying only on the onboard intelligence bestowed by its creators
- **Precision**
  - Critical encounter events such as orbit insertion, landing, or flyby must occur at **exactly** the right time for **exactly** the right duration
  - The penalty for failure is **loss of mission**
- **Complexity**
  - Embedded software is inherently complex since it involves interaction with the real world and the concept of time
  - Deep-space software complexity is amplified by the many interfaces, environments, and scenarios which the software must accommodate

Challenges of Deep Space (2 of 2)

- **Mars '98 mission failures**
  - Both software related
    - But both also system engineering failures
  - One due to failure to keep units straight in **ground software**
    - Mars Climate Orbiter became mistargetted and failed to achieve orbit
  - One due to failure to faithfully run planned tests on **vehicle flight software**
    - Mars Polar Lander erroneously shut down landing rockets 40 meters above the surface of Mars
  - **Both failures were preventable with proper, rigorous application of system engineering and testing principles**

Mitigating the Risks of Deep Space Vehicle Control Software Development (1 of 2)

- **Design for the worst**
  - Vehicle "safe" mode is usually the most critical software
    - If we can get to "safe" mode, we are usually okay
  - Hardware monitors the software heartbeat
  - Advanced fault protection to
    - Detect and correct minor problems, or
    - Ensure achievement of "safe" mode
- **Rigorous development practices**
  - Peer reviews
  - Heritage reviews
  - Coding standards enforced by acceptance reviews
  - Repeatable and documented unit tests

Mitigating the Risks of Deep Space Vehicle Control Software Development (2 of 2)

- **Test, Test, Test!**
  - Test what you fly and fly what you test
  - Plan testing to verify the requirements
  - Plan testing to validate the operation
  - Test when you meet the hardware…
  - …and when the system is completed

Reducing the Cost of Deep Space Vehicle Control Software Development (1 of 2)

- **Autocode Generation**
  - State Charts, XML
  - Used largely in Fault Protection design, but also in communication interfaces (e.g., messages, commands, and telemetry)
  - Permits specification of design in one location and at a higher, more engineer-friendly level
  - Reduces errors in translation from design to implementation
  - Reduces cost of late changes and bug fixes
  - Expanded role?

Reducing the Cost of Deep Space Vehicle Control Software Development (2 of 2)

- **Reuse**
  - Emerging discipline, presently haphazard and based on similarity of mission and/or hardware
  - Formalized and intended at JPL by the Mission Data System (MDS) Project
  - Still several years from application
    - Mars Science Laboratory - 2009
- **Improved and Integrated Design and Verification Tools**
  - UML Tools
  - Formal Verification Tools
    - Model-checking of requirements, design, and implementation
    - Runtime-verification
    - Static analysis tools
    - Force thinking about the **properties** that the **system** must satisfy

Challenges of Robotic Space Exploration: A Summary (1 of 2)

- **JPL has a mission to continue exploration in Deep Space**
- **Deep Space missions are unique, difficult, and complex**
  - If they weren't, anyone could do it!
  - Software for Deep Space missions is also unique, difficult, and complex
- **Developing Deep Space software requires rigor, discipline, and many checks and balances**
  - Reviews
  - Testing
  - Documentation

Challenges of Robotic Space Exploration: A Summary (2 of 2)

- **Cost benefits are being realized through the application of innovative software engineering techniques and tools**
  - Autocode generation
    - State charts, XML, UML
  - Designing for reuse
    - Mission Data System
  - Advanced design and verification tools
    - UML tools
    - Model checkers
    - Runtime checkers
    - Static analysis

Questions?

Artist's conception:
A Mars sample-return mission blasting off from Mars