Onboard Spacecraft Monitor and Control Standards

Peter Glück

Amalaye Oyake

Ashton Vaughs

Autonomy and Control Section

Jet Propulsion Laboratory

Background

- **There is a need to:**
  - Address non-standardization of the Spacecraft Onboard Interface (SOIF)
  - Leverage well defined standards that exist in the desktop and commercial computer industry (process control, embedded, etc.)
  - Recommend a well defined set of standards that would benefit software architecture for spacecraft onboard spacecraft
- **Objectives:**
  - Study possible Onboard Monitor and Control Standards being proposed in CCSDS TSG working group
  - Identify possible JPL contributions
  - Produce a recommended set of standards after a thorough search of the solution space (desktop, commercial)
  - Define a generic device attributes list
  - Represent the solution using a modeling language (UML, IDL)

Motivation (1 of 2)

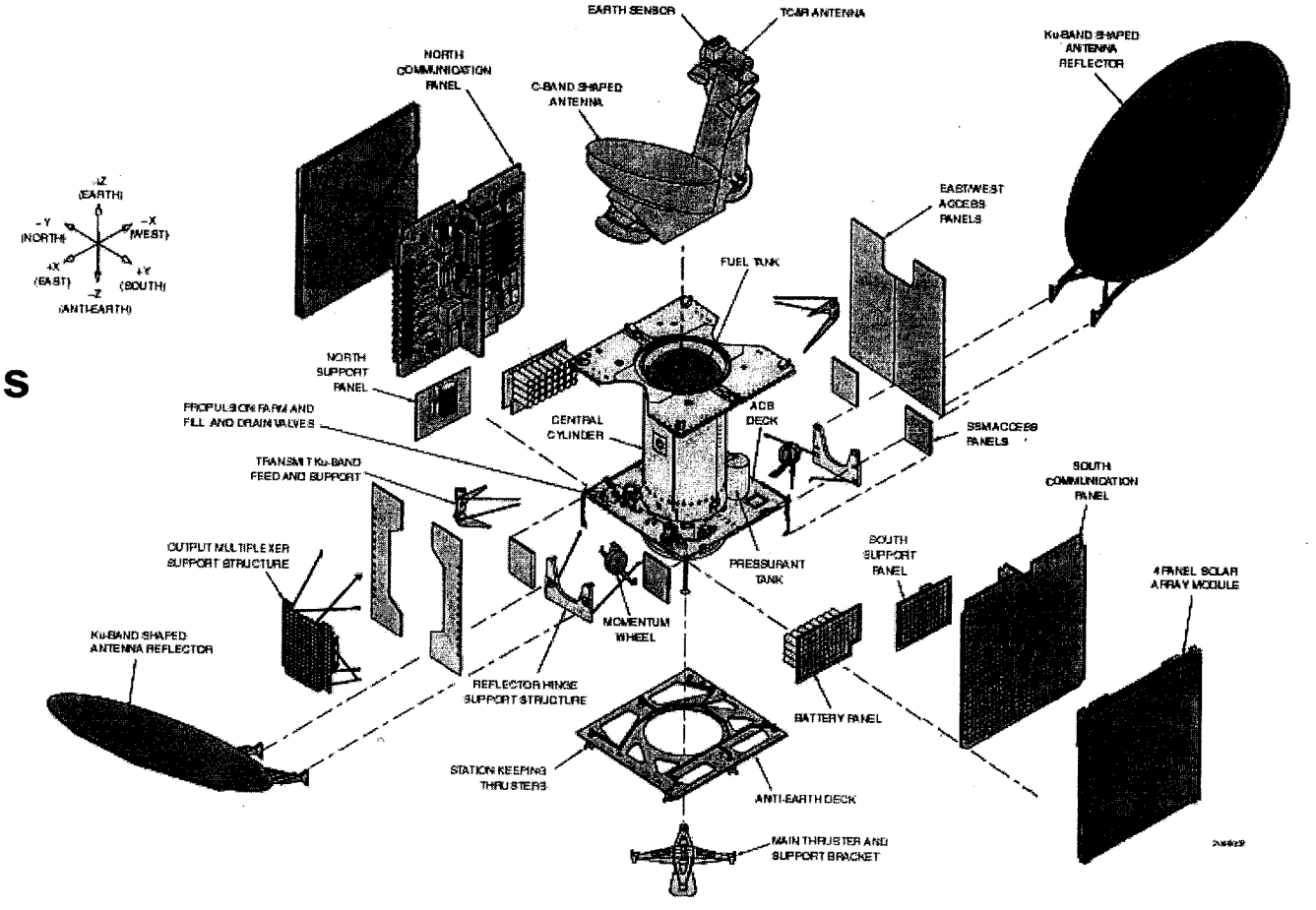
- **Spacecraft complexity is increasing**
  - Greater autonomy, multiple roles
  - Formations, constellations, and networks
- **Lack of current standards is an obstacle to mission development**
  - A standard will help in interfacing Monitor and Control layers
- **Software and hardware reusability**
  - Knowledge capture of previous work can be leveraged for future missions
  - Encourages use of well-defined design patterns
- **Take advantage of good bus technologies regardless of software/hardware interface**
  - RS232, 1553, CAN, Spacewire, I2C, USB, Infiniband, etc.

Motivation (2 of 2)

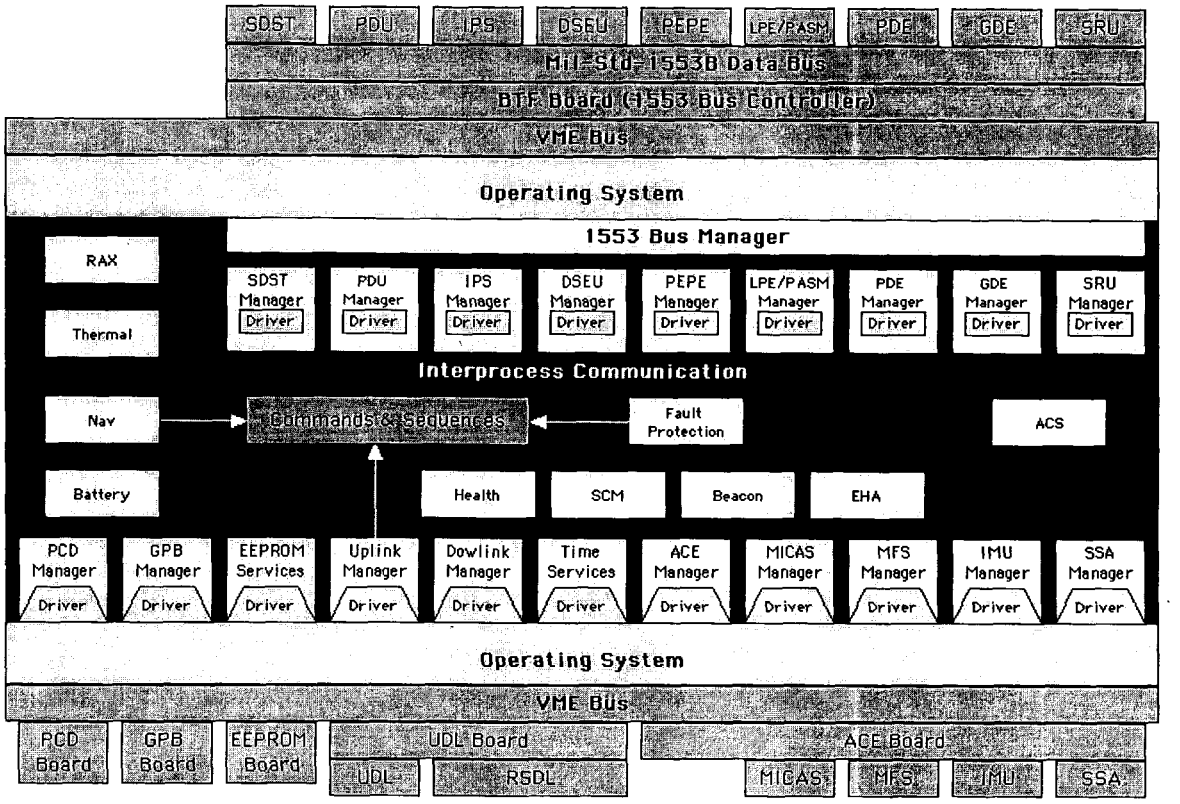
- **Decouple software applications from data bus implementation**
  - Enables reuse with different busses
    - Therefore different spacecraft
  - Changing the bus should affect only the interface

Hardware Complexity

- Many instruments ~ many buses ~ many challenges
- 1553
- RS232
- RS485
- VME
- Compact PCI
- Custom designs



Software Complexity

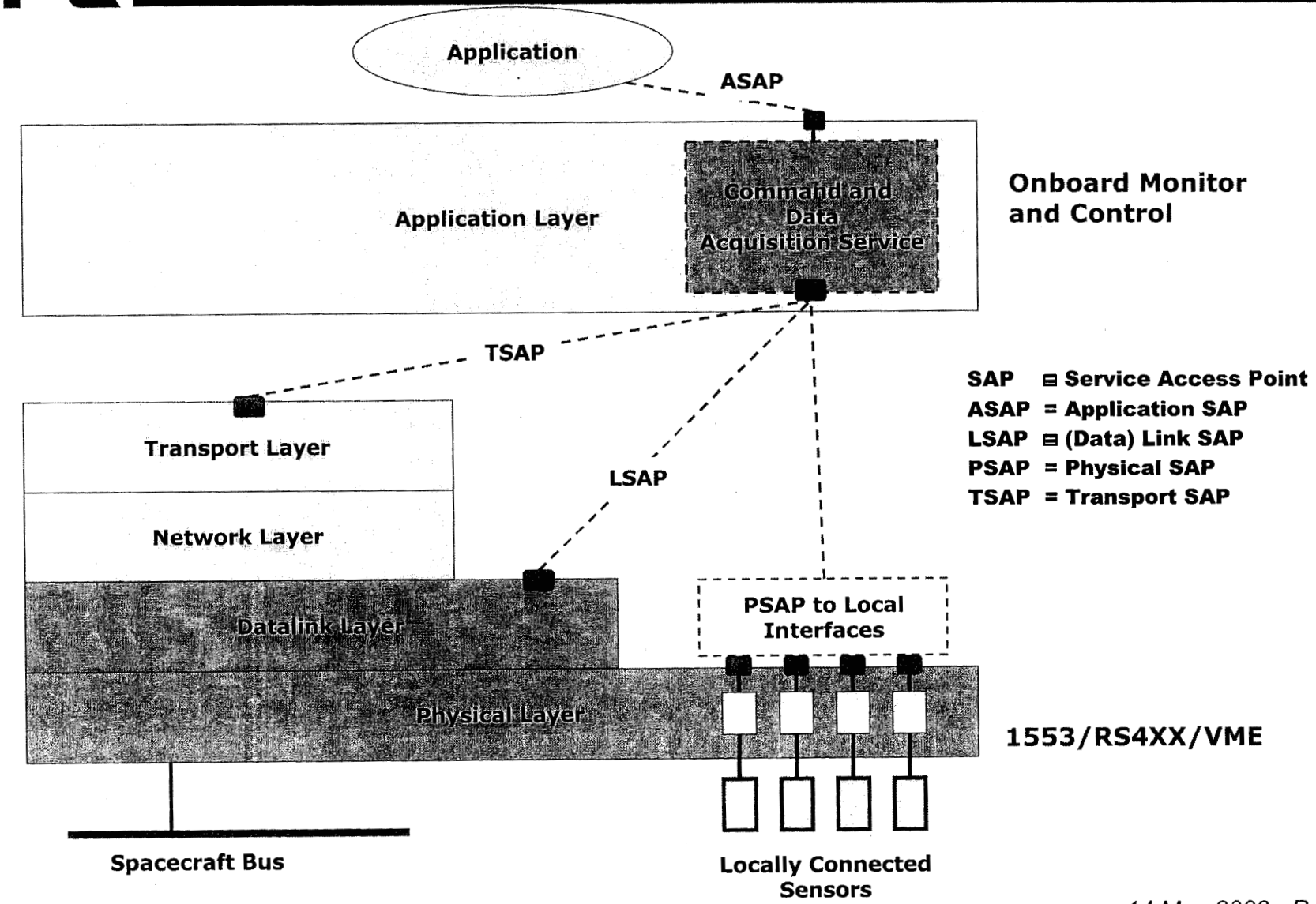


- Pink = Hardware
- Green = O/S
- Yellow = FSW module,
- Orange = Commanding
- Light blue = Device driver
- Dark blue = Interprocess communication
- Vertical positions show relationships between hardware and software
- Arrows indicate the available sources of commands and sequences

Approach

- **Study commercial standards (MS WDM, Fieldbus)**
- **Review current JPL practices and develop SOIF use cases**
- **Develop JPL mission-specific requirements**
- **Generate device attributes list**
  - Draft attributes for some devices (star trackers, cameras, etc.)
- **Develop recommendations to CCSDS WG**
- **Develop an abstract framework using UML & XML schema**
  - Distributed points of control
  - Allowed relationships
  - Roles and responsibilities
  - Definition of all the actors (UML)
- **Develop a library of design patterns**

Proposed CCSDS SOIF Command & Data Acquisition Service Architecture



SOIF Hardware Support Wish List

- **Data / Control Bus**
  - MIL-1553B Data Bus (from NASA-GSFC; NASA-JPL, Honeywell, CNES)
  - 4-255 Data Bus (from ESA)
  - MIL-1773 Data Bus (from NASA-JPL)
  - (10/100/Gigabit) Ethernet (from NASA-JPL, Fisher-Rosemount, SRI International)
  - VME/Compact-PCI (from NASA)
- **Serial Link**
  - Spacewire (from ESA)
  - Liaisons Numeriques Rapides (LNR) (from CNES)
  - IEEE-1394 (from NASA-JPL, NASAGSFC, Honeywell)
  - Packet Wire (from Saab Ericsson Space)
  - Inter-IC (I2C) Bus (from NASA-JPL, NASA-GSFC, Honeywell))
  - Foundation H1 Fieldbus (Boeing, Fisher-Rosemount, SRI International)
  - RS232, RS4XX (Legacy applications)
- **Discrete Interfaces**
  - 16-bit Serial Digital Output Interface
  - 16-bit Serial Digital Input Interface
  - 16-bit Bi-directional Serial Digital I/F
  - Bi-level Discrete Input Interface
  - Analogue Signal Interface
  - On/Off & High Power Pulse CMD I/F
  - Switch Status Interface
  - Timing and Clock Output Interface (All Discrete I/Fs from ESA)

ESA Experience
Integral Mission (1 of 2)

- **International Gamma-Ray Astrophysics Laboratory (INTEGRAL) launched 17 October 2002**
- **ESA Position:**
  - Mastering software development for large space systems is a major priority for both ESA and the space industry
  - Several initiatives to improve the development and quality of onboard software are being investigated
  - Employ reusable and configurable software infrastructure to build mission-specific control systems that are adaptable to support all different types of missions ranging from small missions to complex ones

ESA Experience
Integral Mission (2 of 2)

- **Integral Mission OPS provided with standardized Monitor & Control Framework**
  - SCOS-2000 provides all essential functions to monitor and control a satellite
  - On-orbit and during ground test
  - Platform/vendor independent
  - Scalable
  - Follows a client/server concept (heavily COTS ~ CORBA, SQL, etc)
- **NOTE: THIS IS A GROUND SYSTEM IMPLEMENTATION**

ESA Experience Lessons Learned

- **Well defined and adopted standards are helpful:**
  - Spacecraft software complexity is alleviated
  - Standards assist mission development
  - Enables software and hardware reusability
    - Knowledge capture of previous work can be leveraged for future missions
    - Encourages standardization using well-defined design patterns
  - Permits hardware engineers to be more flexible with technologies
    - RS232, 1553, CAN, Spacewire, I2C, USB, Infiniband, etc.
    - Reduces sensitivity to legacy software implementations

DS1 Experience

- **DS1 inherited Mars Pathfinder software for core services**
- **Strengths**
  - Flight-proven software base for the same CPU
- **Weaknesses**
  - Much not applicable to DS1 mission (ACS, EDL, Rover, Lander, FP).
  - Lots of experienced support promised, but little actually available
    - Mars Pathfinder mission operations
    - Vacations, attrition, new and more interesting assignments
  - Much time lost understanding intricacies and hardware differences
  - Six months to get the inherited software working well with DS1's system
  - Different ground system
- **Inheritance came mid-stream in response to technology readiness concerns**
- **Lessons**
  - Software inheritance is a very good thing when properly planned
  - Personnel familiar with the inherited software should also be inherited

JPL Standardization Efforts

- **SuperMoca – Demo used Fieldbus FMS**
  - NASA-sponsored task
    - Develop an architecture, technologies and open system specifications for the monitor and control of remote space vehicles and supporting remote ground networks
  - Status: No longer ongoing ~ Michael K. Jones.
- **SOIF – CCSDS International Standardization Effort.**
  - Link: http://www.ccsds.org
  - Status: Ongoing effort ~ Joseph F. Smith.
- **SIM Testbed**
  - Uses CORBA as the Object Request Broker
    - JPL Centric solution
    - Robust Implementation with devices supported as C++ Objects
  - Link: http://rtc-lib.jpl.nasa.gov/View/Collection-196
  - Status: Fully implemented.

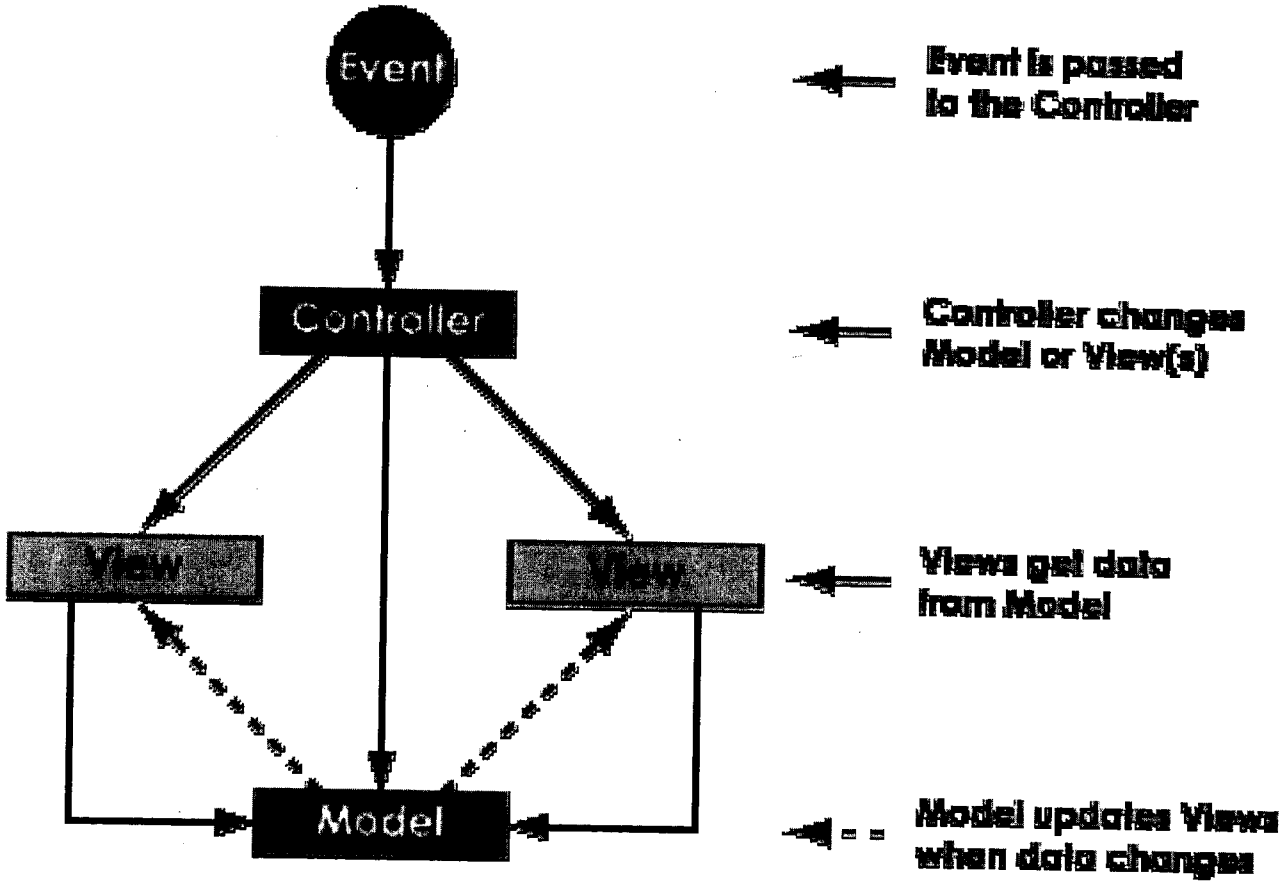
Approach - Hardware Abstraction Layers

- **A component of an operating system/FSW**
  - Functions as an API for hardware abstraction
  - Resides at the device level, a layer below the standard API level
  - Devices are:
    - sensors (e.g., star trackers)
    - actuators (e.g., switches)
    - payloads (e.g., spectrometers)
- **Enable programmers to write device-independent applications**
  - Device details are abstracted away
- **Exposes well-defined design patterns to the application developer**

Approach - Design Patterns

- **Device driver frameworks typically use these design patterns**
- **Model-View-Controller (MVC)**
  - Classic design pattern used when there is a need to maintain multiple views of the same data
  - Hinges on a clean separation of objects into one of three categories
    - models for maintaining data
    - views for displaying all or a portion of the data
    - controllers for handling events that affect the model or view(s)
- **Configurator Pattern**
  - Decouples configuration data from the components that abstract the devices
- **Mediator Pattern**
  - How to connect two components together without directly coupling them
- **Wrapper Façade**
  - Provides an object-oriented interface to legacy or traditional code
- **Device/Sensor Proxy**
  - Provides a uniform interface to a device

Approach - Model-View-Controller Design Pattern



Progress - Industry Practices
MS Windows Driver Model (1 of 2)

- **Problem: Device drivers for DOS/Windows 3.x/95 were horrid**
  - To encourage drivers compatible with all Microsoft Windows operating systems, the MS WDM was introduced
  - Kernel-mode drivers that follow WDM rules are called WDM drivers
- **All WDM drivers must:**
  - Include Wdm.h
  - Be designed as a bus driver, a function driver, or a filter driver
  - Create device objects as WDM Device Objects and Device Stacks
  - Support Plug and Play
  - Support Power Management
  - Support Windows Management Instrumentation (WMI)
  - A WDM driver is loaded and unloaded dynamically based on entries in the registry

Progress - Industry Practices
MS Windows Driver Model (2 of 2)

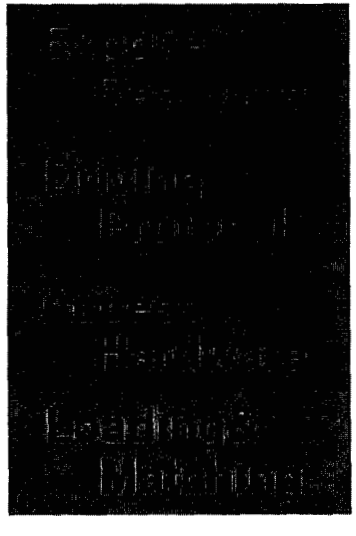
- **Incremental changes to WDM have resulted in later-released operating systems supporting additional WDM features**
  - Later versions of WDM generally support all the features available in earlier versions of WDM
- **WDM Model developed over many iterations**
  - Windows 98 - Windows 2000 - Windows XP
- **Many design patterns defined**
  - No one size fits all model
- **Very robust design**
  - Good drivers should rarely "blue screen"
- **Highly modular design**
  - Drivers can be layered

Progress - Industry Practices
Apple I/O Kit (1 of 4)

- **Apple's object-oriented device driver programming framework**
  - True plug-and-play
  - Dynamic device management ("hot plugging")
  - Power management
- **Two types of device interfaces**
  - An intermediate I/O Kit plug-in interface:
    - Used to create device interfaces for specific devices
  - A specific device interface.
- **Similar to Microsoft COM**
- **Highly adaptable**
- **Complete redesign**
  - Neither Mac OS 8/9 driver model nor the FreeBSD Model offered sufficient features

Progress - Industry Practices
Apple I/O Kit (2 of 4)

General Thinking…



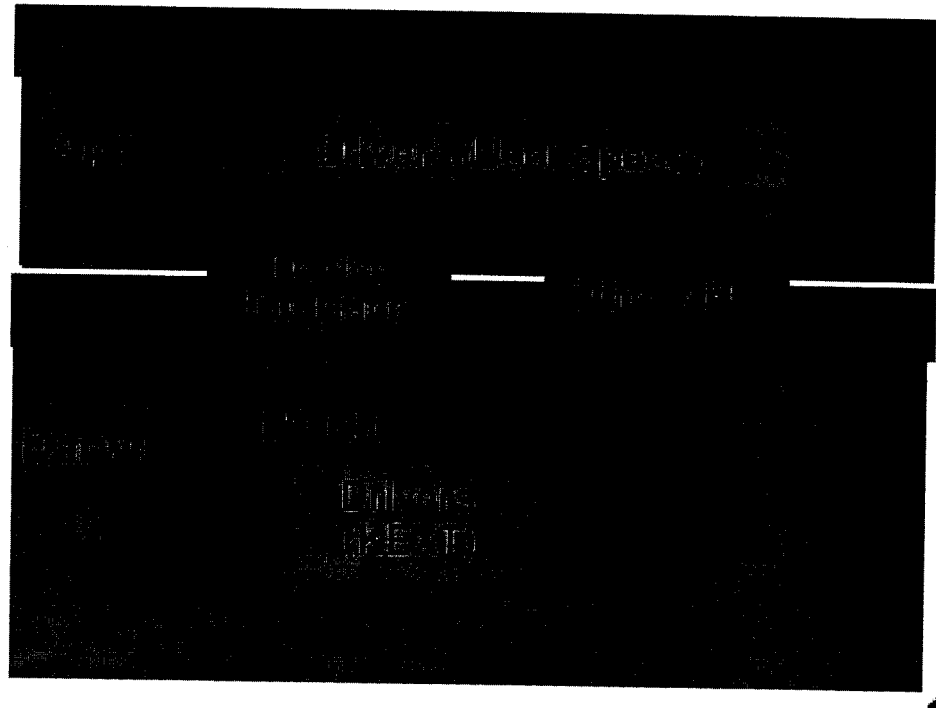
- Loading & Matching
  - A Driver should be Loaded and Matched to the Appropriate Device
- Access Hardware
  - A Driver Need a Channel to Read from and Write to the Device
- Driving Protocol
  - The Command Set to Control and Manage the Device
- Expose Resource
  - A Driver should Expose Itself so that Other Applications can Interact with the Device

Progress - Industry Practices
Apple I/O Kit (3 of 4)

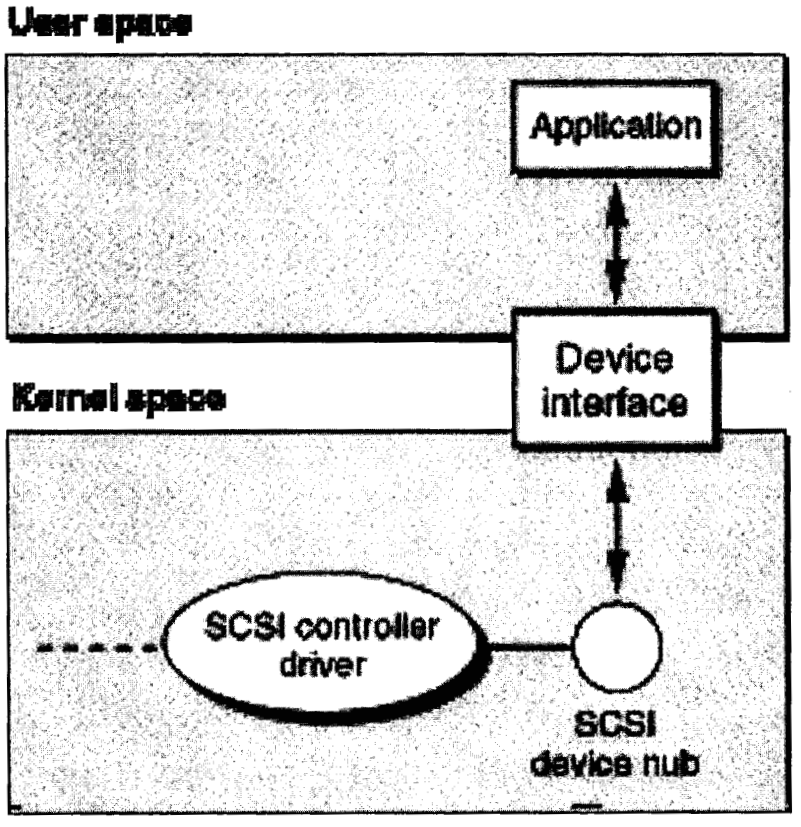
Device Driver in Mac OS X

User Space

Kernel Space



Progress - Industry Practices
Apple I/O Kit (4 of 4)



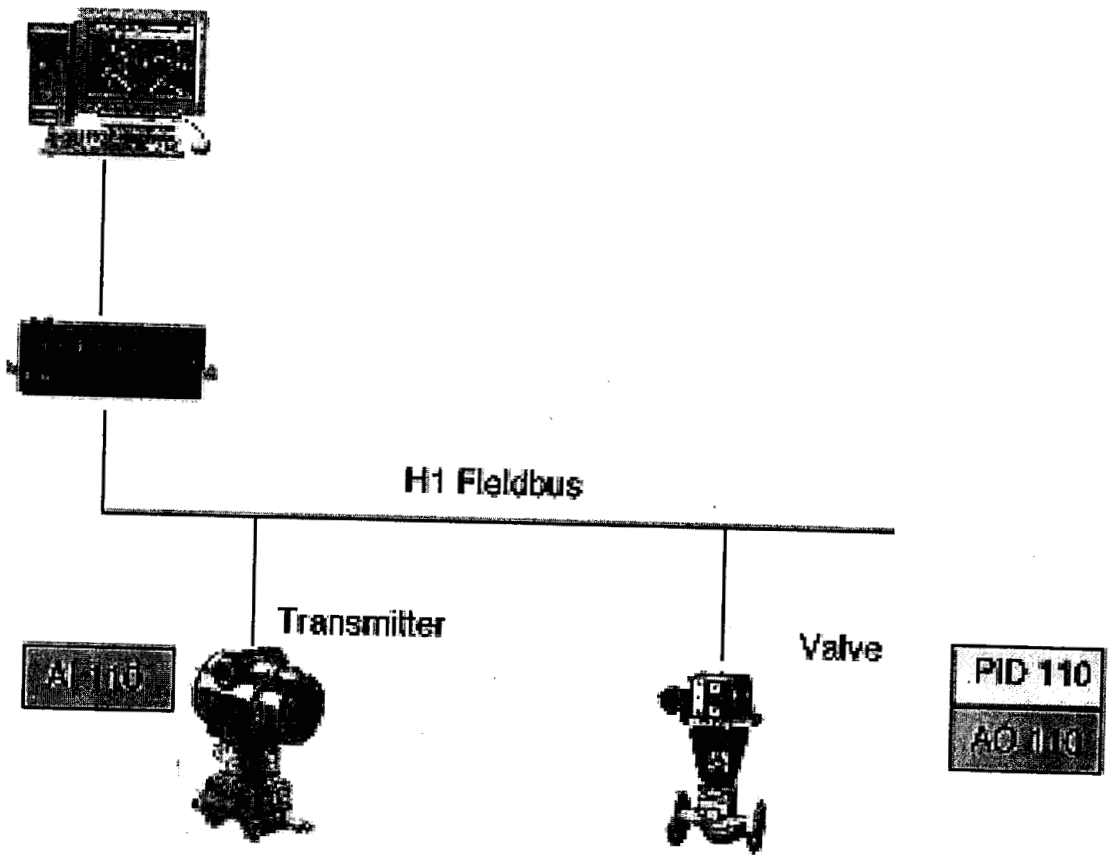
Progress - Industry Practices Foundation Fieldbus (1 of 4)

- **Foundation Fieldbus**
  - Bi-directional digital communications network
  - Enables connection of multiple field instruments and process operator stations
  - Based on ISO/OSI seven-layer communications model
  - True plug and play device technology
  - Functionality is wrapped in the device
    - Each device is like a CORBA Object
    - Functions are defined via the Fieldbus specific ICs that allow the device to communicate over the network
  - Fieldbus Messaging Specification (FMS)
    - Contains definitions of Application Layer services
    - Specifies services and message formats for accessing Function Block (FB) parameters
  - Function Block
    - A generalized concept of the functionality in field instruments and control systems e.g. analog input, analog output, etc.
  - FMS and Function Block provide a basis for a Device Attribute List.
- **The SuperMoca work at JPL demonstrated the synergy of this technology and current JPL practices ~ Michael K. Jones**

Progress - Industry Practices Foundation Fieldbus (2 of 4)

- **Function Blocks ensure device interoperability**
  - User Layer defines a Function Block Application Process (FBAP) using Resource Blocks, Function Blocks, Transducer Blocks, System Management, Network Management and Device Description technology
  - Blocks are incorporated into fieldbus devices to achieve the desired device functionality
  - Defines standard features and behaviors so devices can interoperate
  - Each block's parameters are represented by object descriptions that define how the parameters are communicated on the fieldbus network. Devices from different manufacturers to interoperate.
- **Fieldbus Control includes**
  - Synchronization of function block execution
  - Communication of function block parameters on the fieldbus
  - Publication of time-of-day to all devices
    - Automatic switch over to a redundant time publisher
  - Automatic assignment of device addresses
  - Searches for parameter names or "tags" on the fieldbus
    - Handled by System Management and Network Management.

Progress - Industry Practices Foundation FieldBus (3 of 4)



Progress - Industry Practices Foundation Fieldbus (4 of 4)

- **Fieldbus provides**
  - A Device Attribute List that we can reference
  - A Device Messaging Specification
  - A Pattern for device commanding
  - A successful framework implementation

Progress - Industry Practices Summary

- **We have examined:**
  - CCSDS requirements and case studies
  - Microsoft WDM
  - Apple I/O Kit
  - Fieldbus
- **Question ~ How can this fit into JPL/Industry?**
  - JPL, and space industry are now very VxWorks, VME and Compact-PCI centric
  - Wind River and Green Hills have a large install base
  - Mathworks supports all of the above, and has a large install base
  - This commonality provides a good starting point
  - Allows us to map a reference implementation to real hardware

Issues

- **The level of buy-in by JPL and related agencies needs to be discovered and solution space within JPL needs to be defined**
- **These concepts need to be marketed, within JPL**
  - CCSDS must address the propagation of SOIF through NASA/ESA
- **Once interest is generated there should be working groups**
- **Besides NASA/ESA adoption, these ideas need to introduced to the commercial vendor domain**
- **There should be a proposal and a commitment for actual prototype work**