-5-61

27236

p-16

1995107747

MISSION SAFETY CRITICAL SYSTEMS
RESEARCH & APPLICATIONS
ricis
SYMPOSIUM '92

A GENERIC ARCHITECTURE MODEL FOR SPACE DATA SYSTEMS

RICHARD B. WRAY

ABSTRACT

A Space Generic Open Avionics Architecture (SGOAA) was created for the NASA, to be the basis for an open, standard generic architecture for the entities in spacecraft core avionics. Its purpose is to be tailored by NASA to future space program avionics ranging from small vehicles such as Moon Ascent/ Descent Vehicles to large vehicles such as Mars Transfer Vehicles or Orbiting Stations. This architecture standard consists of several parts: (1) a system architecture, (2) a generic processing hardware architecture, (3) a six class architecture interface model, (4) a system services functional subsystem architecture model, and (5) an operations control functional subsystem architecture model.

This paper describes the SGOAA model. It includes the definition of the key architecture requirements; the use of standards in designing the architecture; examples of other architecture standards; identification of the SGOAA model; the relationships between the SGOAA, POSIX and OSI models; and the generic system architecture. Then the six classes of the architecture interface model are summarized. Plans for the architecture are reviewed.

BIOGRAPHY

Richard B. Wray has a dual MS/MBA in Systems Management, Acquisition, and Contracting; a MBA with Distinction Honors in General Management; a MS in Systems Engineering and a BS in Mathematics. He is the Vice President-Technical of the National Council on Systems Engineering (NCoSE) Texas Gulf Coast Chapter, and Chairman of the NCoSE Systems Engineering Process Working Group to define a nationally recognized systems engineering process.

Mr. Wray has been with Lockheed Corporation for over eight years as an Advanced Systems Engineer responsible for developing and implementing plans, performing systems engineering and analysis, and implementing systems. He is leading the definition of a Space Generic Avionics architecture and simulation using advanced computer aided systems engineering (CASE) Tools. Prior to this, he performed and led analyses of system operational concepts, functional and performance requirements, hardware/software trade-offs and system timelines for advanced avionics systems and architectures. Mr. Wray was the lead software systems engineer for the definition and development of a software system architecture, software analyses and the DoD-STD-2167A specifications for an avionics subsystem and its interfaces to other avionics on the Advanced Tactical Fighter.

Overview SPACE GENERIC OPEN AVIONICS ARCHITECTURE (SGOAA)

for the Mission and Safety Critical Systems Symposium

28 October 1992

Dick Wray
John Stovall

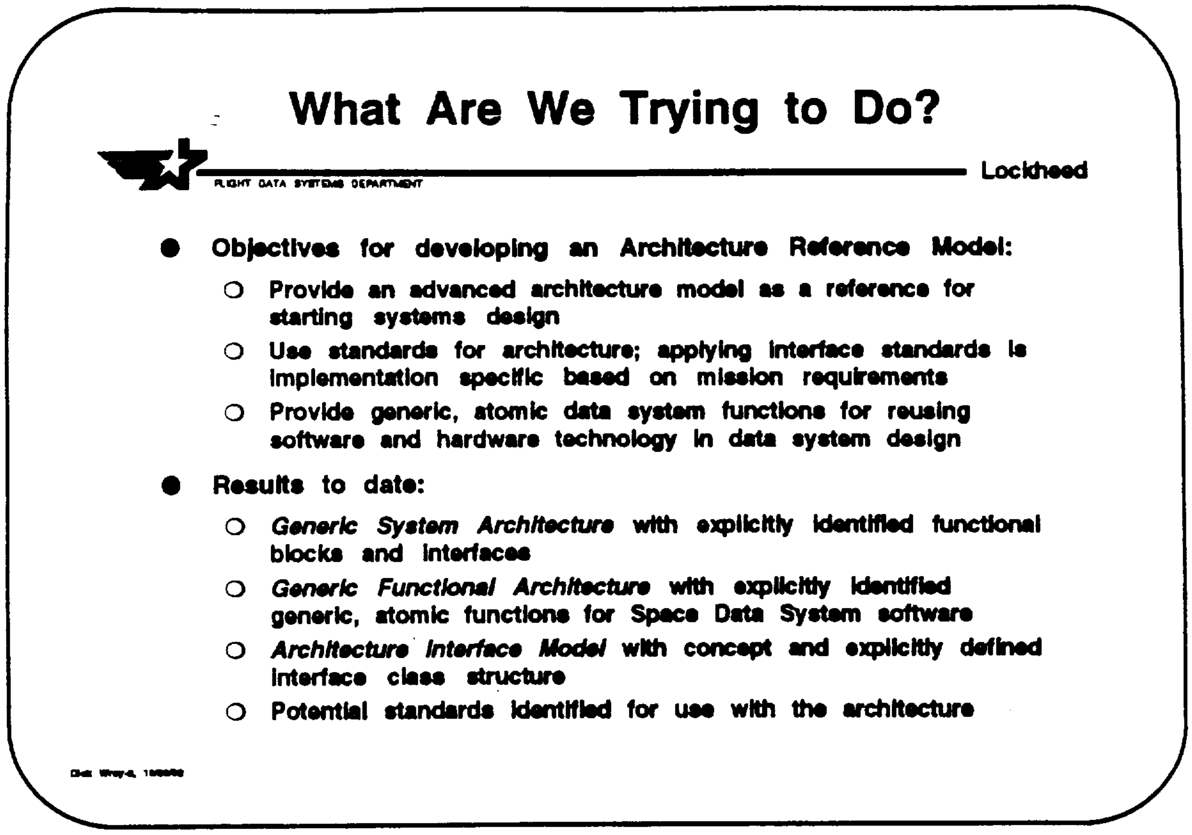
Notes:

- What we are trying to do
- What is an architecture reference model
- What is the Space Generic Open Avionics Architecture (SGOAA) Model
- Summary

Acknowledgements

The architecture presented here was developed by David Pruett of NASA's Johnson Space Center, Dick Wray and John Stovall of the Lockheed Engineering and Sciences Company (LESC). Additional support was provided by Ben Doeckel of LESC.

GOAL: TO CREATE AN ARCHITECTURE *BLUEPRINT* FOR TAILORING TO NEW PROGRAMS



What Are We Trying to Do?

FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

- Objectives for developing an Architecture Reference Model:
  - Provide an advanced architecture model as a reference for starting systems design
  - Use standards for architecture; applying interface standards is implementation specific based on mission requirements
  - Provide generic, atomic data system functions for reusing software and hardware technology in data system design
- Results to date:
  - *Generic System Architecture* with explicitly identified functional blocks and interfaces
  - *Generic Functional Architecture* with explicitly identified generic, atomic functions for Space Data System software
  - *Architecture Interface Model* with concept and explicitly defined interface class structure
  - Potential standards identified for use with the architecture

[illegible]

Notes:

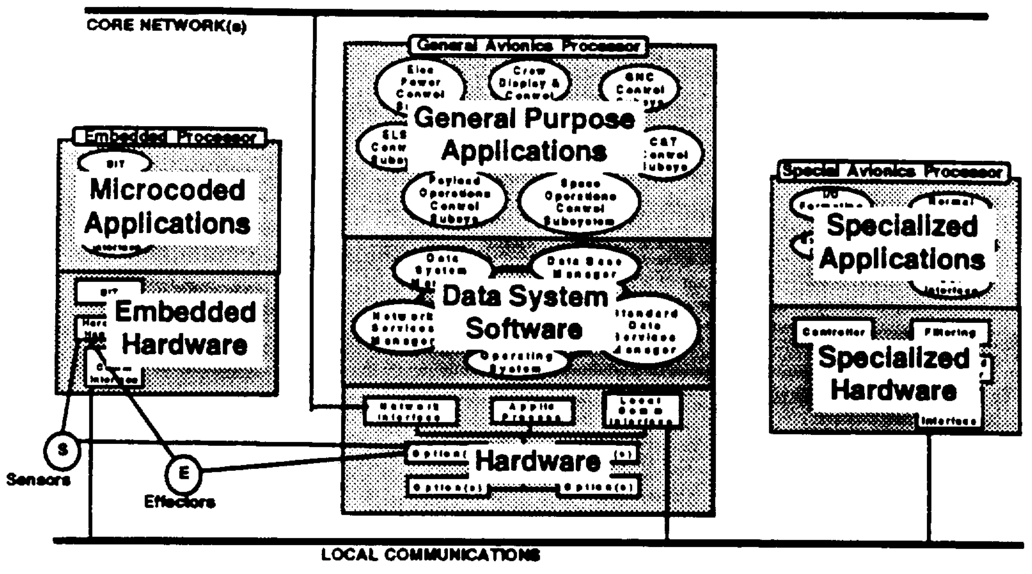
What is an Architecture Reference Model?

- Architecture
  - A set of black boxes, interfaces between the black boxes and interfaces from them to the external environment
  - All functions and performance defined at the interfaces between the black boxes
  - Software "black boxes" as well as hardware black boxes
- Physical Interfaces
  - Interfaces identified with physical connectivity between black boxes (hardware and software)
  - Interfaces handle signal and data flow between the black boxes
- Logical Interfaces
  - Interfaces identified with the meaning of data passed between two black boxes where one is the originator and one is the user of the data
  - Information exchange can cross many physical interfaces for one logical interface, and therefore cannot be identified with any one physical interface

What is the Generic System Architecture Model ?

FLIGHT DATA SYSTEMS DEPARTMENT

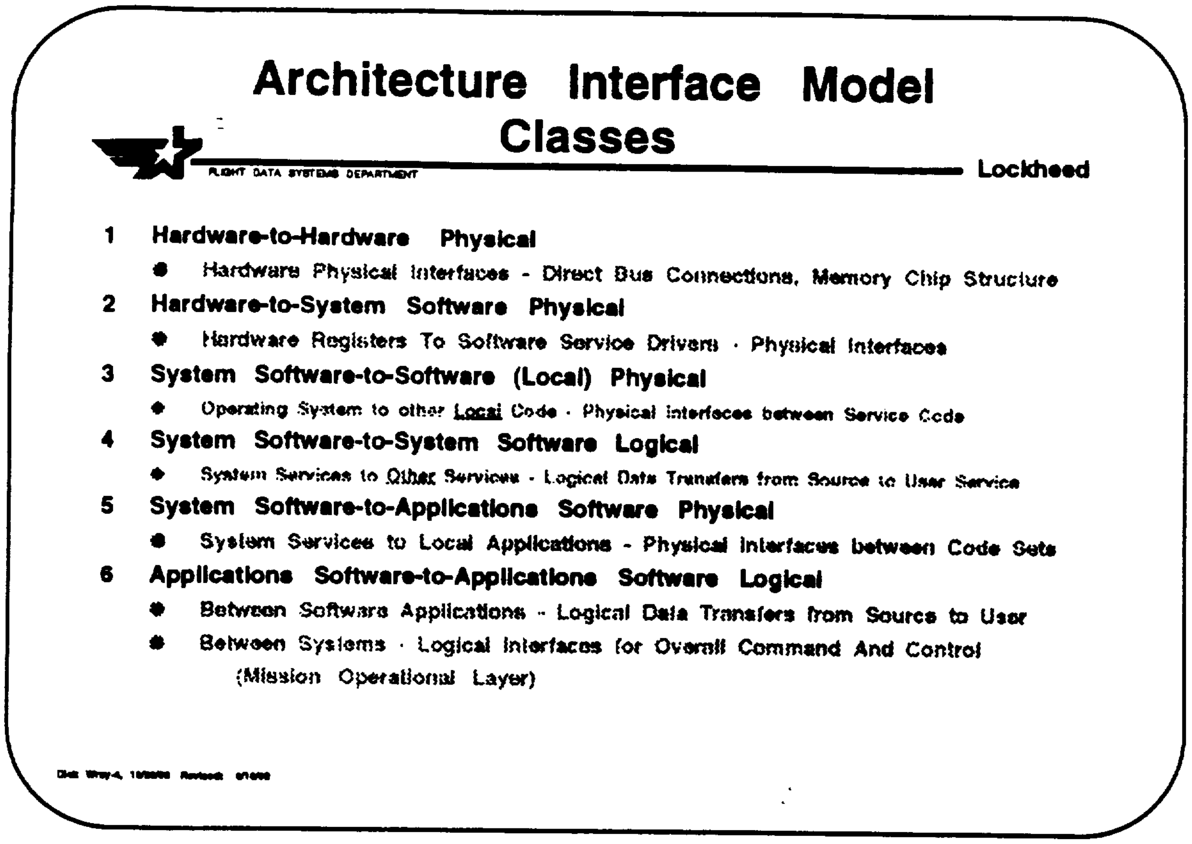
Lockheed



Dick Wray-6, 10/30/89

Notes:

- The generic and open system architecture proposed consists of processors which are standard, processors which can be tailored to users applications and missions needs, multiple communications mechanisms, and specialized hardware operating over standardized interfaces to the processors which manipulate the data they receive or provide.
- There are three types of processors interconnected by two types of communications. The processors provided in the architecture are the General Avionics Processor (GAP) for general purpose processing, the Special Avionics Processor (SAP) for specialized processing support, the Embedded Processor (EP) for the execution of high speed processing witin the sensor and effector devices.
- The communications provided are two types: core networks for interconnecting sets of general processors or nodes, and local communications for interconnecting EPs and SAPs with their supported GAPs and general purpose processing applications.
- There are sensors and effectors which can either interact directly with the main processors (the GAPs) or indirectly through the EPs built into the sensors and effectors (if applicable).



Architecture Interface Model Classes

FLIGHT DATA SYSTEMS DEPARTMENT Lockheed

1 **Hardware-to-Hardware Physical**
- Hardware Physical Interfaces - Direct Bus Connections, Memory Chip Structure

2 **Hardware-to-System Software Physical**
- Hardware Registers To Software Service Drivers - Physical Interfaces

3 **System Software-to-Software (Local) Physical**
- Operating System to other Local Code - Physical Interfaces between Service Code

4 **System Software-to-System Software Logical**
- System Services to Other Services - Logical Data Transfers from Source to User Service

5 **System Software-to-Applications Software Physical**
- System Services to Local Applications - Physical Interfaces between Code Sets

6 **Applications Software-to-Applications Software Logical**
- Between Software Applications - Logical Data Transfers from Source to User
- Between Systems - Logical Interfaces for Overall Command And Control (Mission Operational Layer)

[illegible]

Notes:

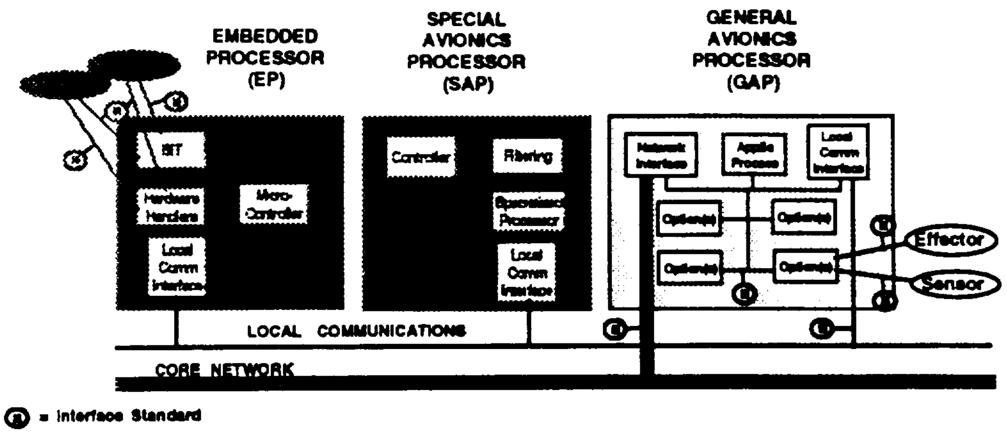
- **The architecture interface model focuses on logical and physical interfaces to isolate and enable specifications of effective interfaces. Logical interfaces are those connecting elements which provide information with those needing it; these are "ought to" type interfaces not real interfaces. So hardware logical does not exist because either it connects or it does not, no "ought to" is involved. This progression from hardware to system logical interfaces moves from the things people can touch and feel to the conceptual/logical. Software physical interfaces are those interfaces between two sets of software code,e.g., one code package calling another.**
- **Class 1, Hardware-to-Hardware Interfaces (Physical), addresses hardware component modularity and portability, and maintainability and technology upgradability of platform hardware over extended space avionics life cycles. These hardware interface standards must be definitive as to the software driver interface requirements needed to communicate with that hardware.**
- **Class 2, Hardware-to-System Software Interface (Physical), is the Operating System (OS) hardware driver software to invoke platform services internal to the Application Platform. Each of these standards must specify the software interface binding requirements for "plugging" a driver into the OS.**
- **Class 3, System Software-to-Software (Local Physical), is provides access form the operating system to all other platform services and applications in support of application portability.**
- **Class 4, System Software-to-System Software Interfaces (Logical), is the internal interface for transfer of data between Application Platform Data System Services. For example, this is the logical interface between the Data Base Manager in an Application Platform communicating with the Data Base Manager in another platform.**
- **Class 5, System Software-to-Application Software Interfaces (Physical), is defined primarily to support Application Software portability.**
- **Class 6, Application Software-to-Application Software (Logical), consists of application-to-application software interfaces for both local/node and other systems. Applications software interfaces are the internal interfaces for transfer of data between Application Software within an Application Platform. These are also the external logical interfaces between Application Software on the Application Platform with Application Software on other Application Platforms.**

Hardware-to-Hardware Physical Interfaces

FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

Class 1. Hardware Application Platform

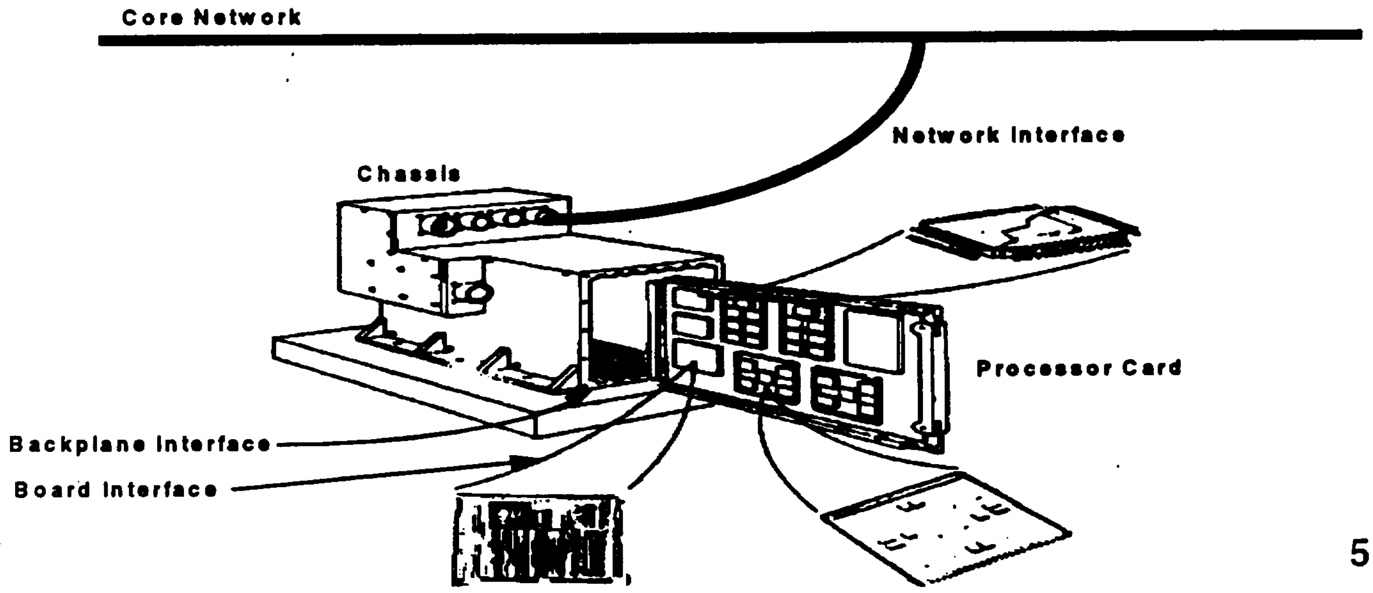


- Plugs Identify Interface Standard Constraints
- GAP based on Internal Interface Standard Constraints

Dat. Way 4, 10/93

Notes:

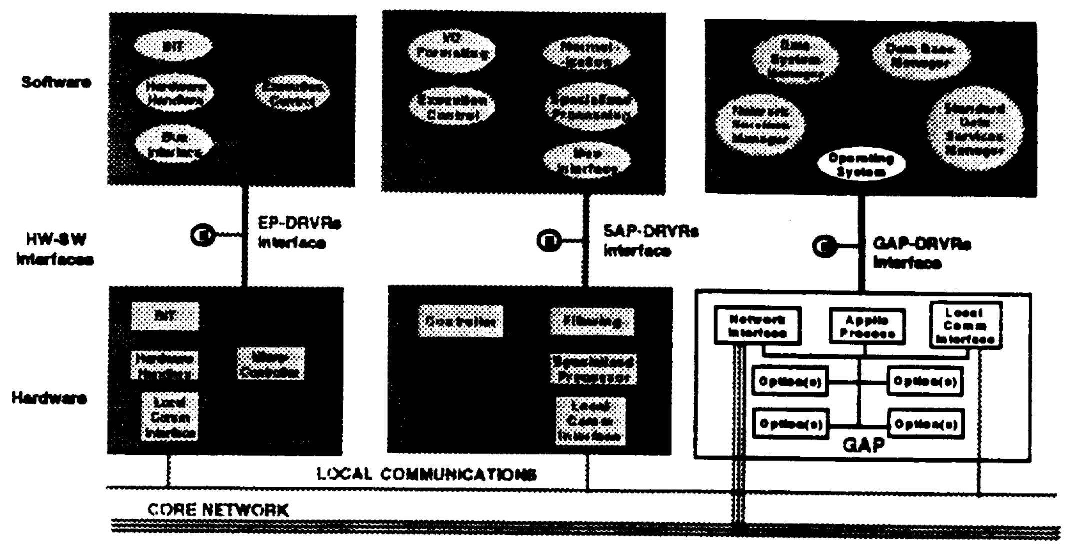
- The hardware to hardware physical interfaces are shown in this slide. These interfaces consists of the nuts, and bolts, chips and wires of the hardware architecture described previously. With regard to the model, it consists of all the hardware to hardware interfaces within each processing element, as well as the hardware interfaces to the external environment by way of the core network, local communications or direct interfaces. The focus in this standard is on GAPs which provide the greatest flexibility in configuring the system to accomplish different purposes in avionics. The GAP includes hardware components to interface to a core network, to interface to local buses, to process applications, and optional components for other purposes (such as serial input and output to direct analog and discrete links). As implied by the darker shading on the GAP, the GAP is the focus of efforts to standardize the hardware processor support due to its general purpose nature. An example of such hardware interfaces is shown below.



Hardware-to-System Software Physical Interfaces

FLIGHT DATA SYSTEMS DEPARTMENT — Lockheed

Class 2. Application Platform Hardware to Service Drivers

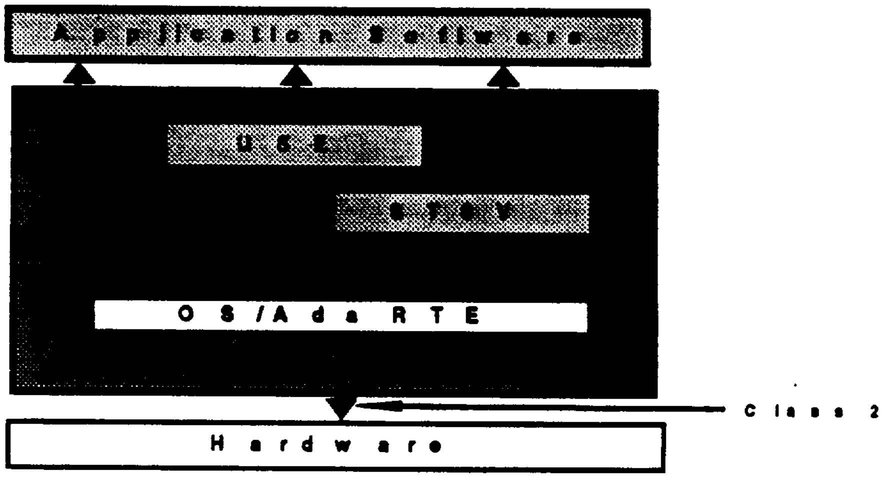


Dick Wray-4, [illegible] Revised: [illegible]

Notes:

- The hardware to system software interfaces are shown in this slide. These consist of the interfaces from the system software drivers (i.e. in the OS, data system manager, etc.) to the hardware instruction set architecture (ISA) and register usage. With regard to the model it is internal to each processing element. The grayed out elements are a repeat of the previous figure; the white elements represent the new capabilities and interfaces added by this class. This class provides low level software drivers to interact with the hardware for each of the processor types (EPs, SAPs, and GAPs). The drivers are (obviously) hardware dependent, but this enables the architecture to begin to partition out the hardware dependencies, which is a key in providing for technology upgradability in the future. All the drivers for all processor types are contained in the Space Data System Services (SDSS) sub-architecture.

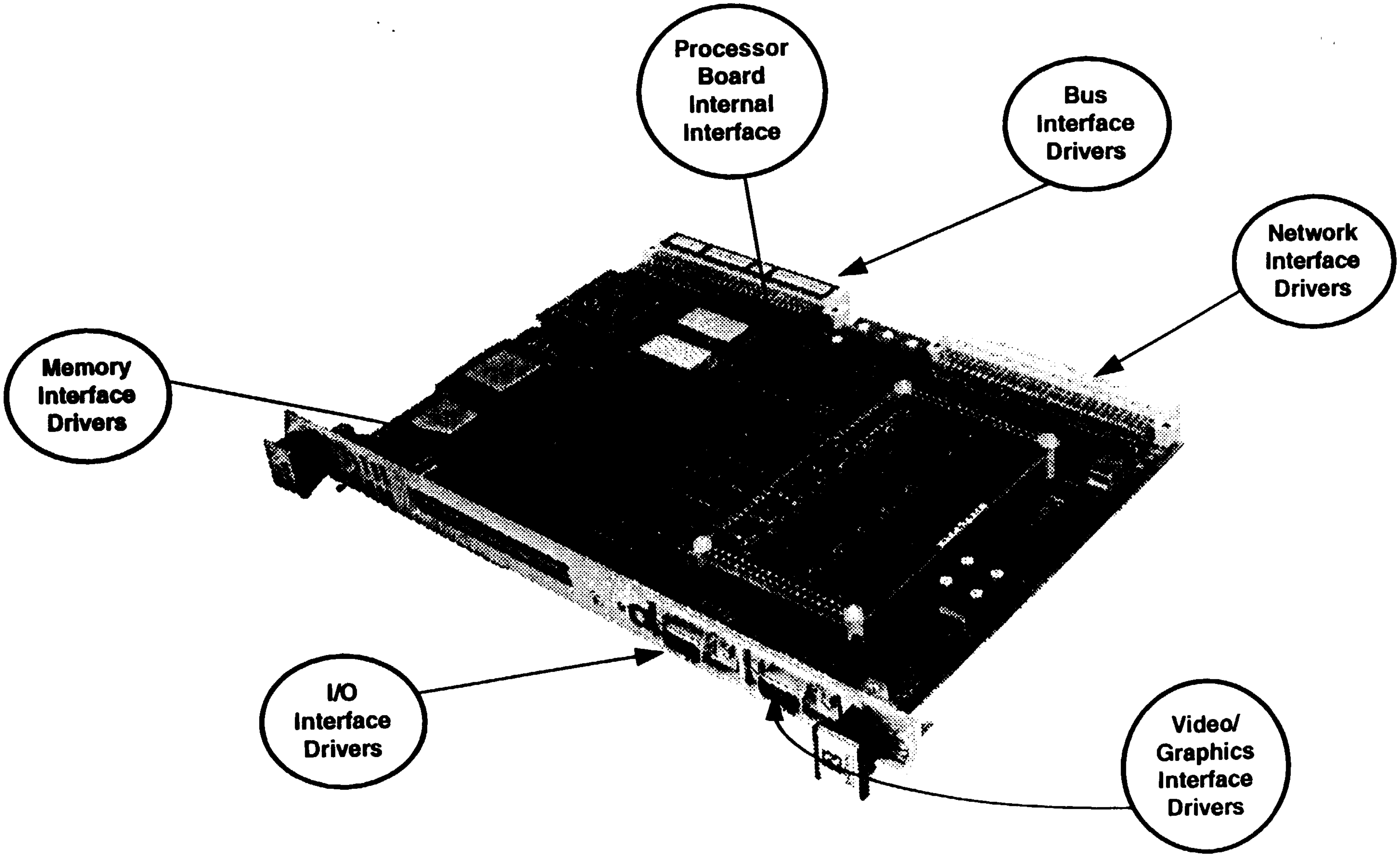
- The system services software for the GAP are organized into five categories. These categories are the Data System Manager, Data Base Manager, Standard Data Services Manager, Operating System, and Network Services Manager. Note the naming convention used for the GAP hardware to the OS drivers, i.e., GAP-DRVR. This single link will be broken open in the next figure to show its component elements. An example (drawn from the Space Station Freedom) of software driver interfaces is shown below.



Example of Hardware to Software Physical Interface

Lockheed

FLIGHT DATA SYSTEMS DEPARTMENT

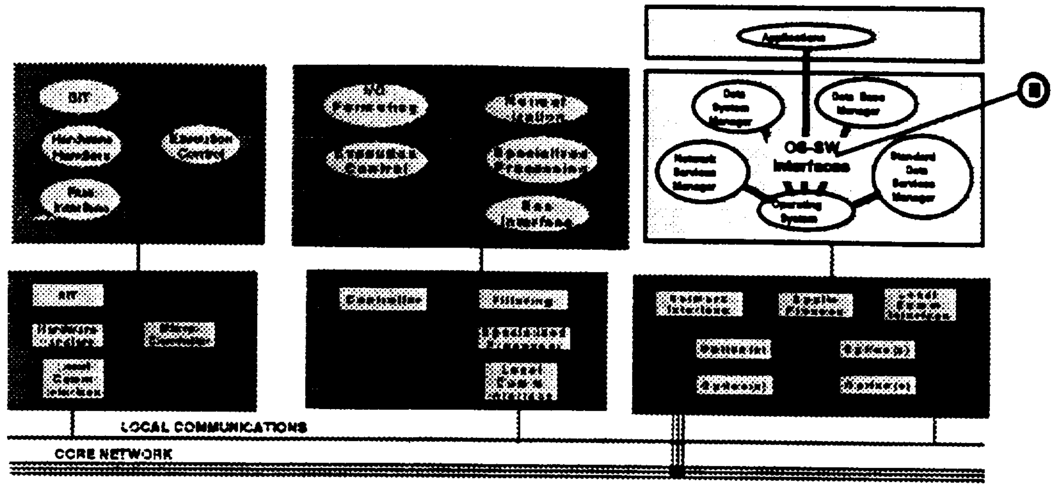

Processor Board Internal Interface
Bus Interface Drivers
Network Interface Drivers
Memory Interface Drivers
I/O Interface Drivers
Video/ Graphics Interface Drivers

System Software-to-Software (Local) Physical Interfaces

FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

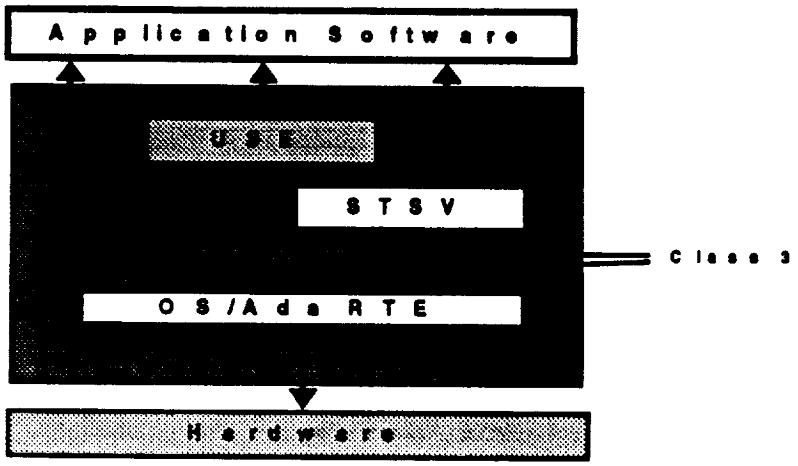
Class 3. Operating System to Other Code (i.e., Services or Applications)



Disk Wrap-7, 10/20/92 Revised: 3/1/93

Notes:

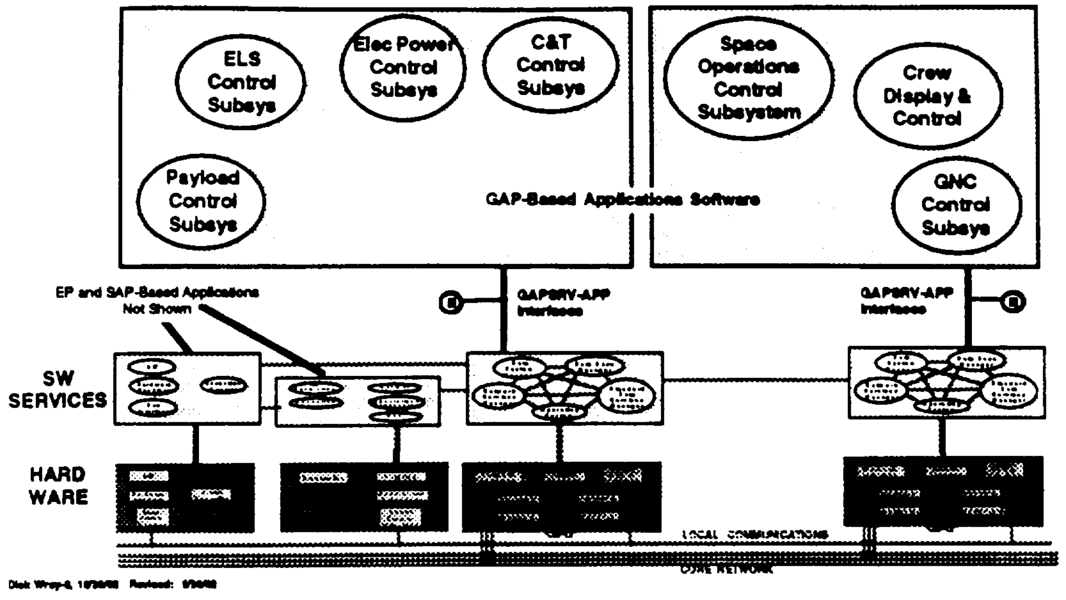
- The system software drivers to local system software service interfaces are shown in next. These consist of the Input/Output handler calling conventions and context switch conversions between the system software drivers on one processing element interfacing with one or more system software services to provide for local information exchange. The grayed out parts of the figure represent the material covered in Classes 1 and 2, the white parts of the figure are the new material added in Class 3. Since Class 2 provided the software drivers to isolate the hardware, Class 3 provides the remainder of the local software services needed to operate the computer system. They all fall into the Space Data System Services (SDSS) sub-architecture, consisting of the Data System Manager, Data Base Manager, Standard Data Services Manager, Operating System, and Network Services Manager. Class 3 provides all remaining services and the interfaces between the local services for effective local interprocess communications and support. These interfaces are physical interfaces because they enable software service code to interact with software service code in other local entities. Class 3 interfaces meet derived requirements based on the need of an application to support users.
- The naming convention (e.g., OS-SW) is shown in this figure to indicate all the OS links both down to OS drivers and up to other high level processes will be identified explicitly by their names. An example from the Space Station Freedom project is shown below of these interfaces.



System Software-to-Applications Software Physical Interfaces

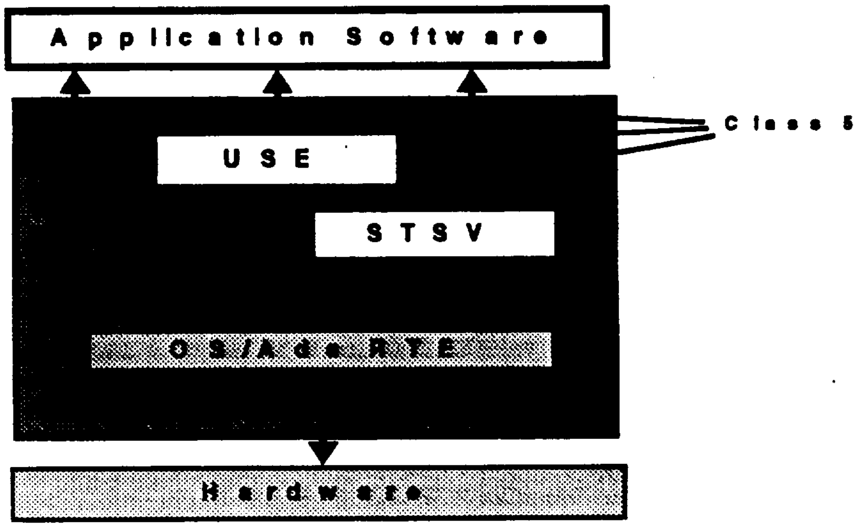
FLIGHT DATA SYSTEMS DEPARTMENT Lockheed

Class 5. Services to Applications



Notes:

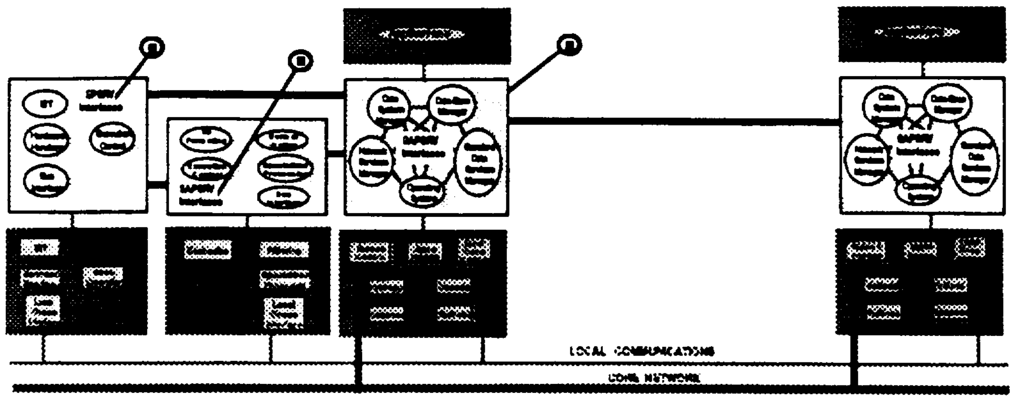
- The system software services to applications software interfaces are shown here. This is the physical interface within a processing element between the application software and the system software (language bindings/specification) to allow provision of needed services. The grayed out parts of the figure represent the material covered in Classes 1 to 4, the white parts of the figure are the new material added in Class 5. Since Classes 1 to 4 isolated the hardware and software services in all the processors, Class 5 adds the interface capability for services in any processor to interact with an application executing in the processor; this provides the basic multi-processor capability to meet actual user requirements in processing. Applications can operate in any GAP, with potential partitioning of an application across multiple GAPs. Similarly, applications can operate in any SAP or any EP. These interfaces are physical interfaces because the applications software code is interacting with the service software code. Class 5 interfaces meet derived requirements based on the need of an application to support users in a multi-processing environment.
- The naming conventions identify the higher level interfaces which will be specified in more detail in lower level diagrams. An example of these interfaces from Space Station Freedom is shown below.



System Software-to-System Software Logical Interfaces

FLIGHT DATA SYSTEMS DEPARTMENT Lockheed

Class 4. Services to Other Services Data Transfers

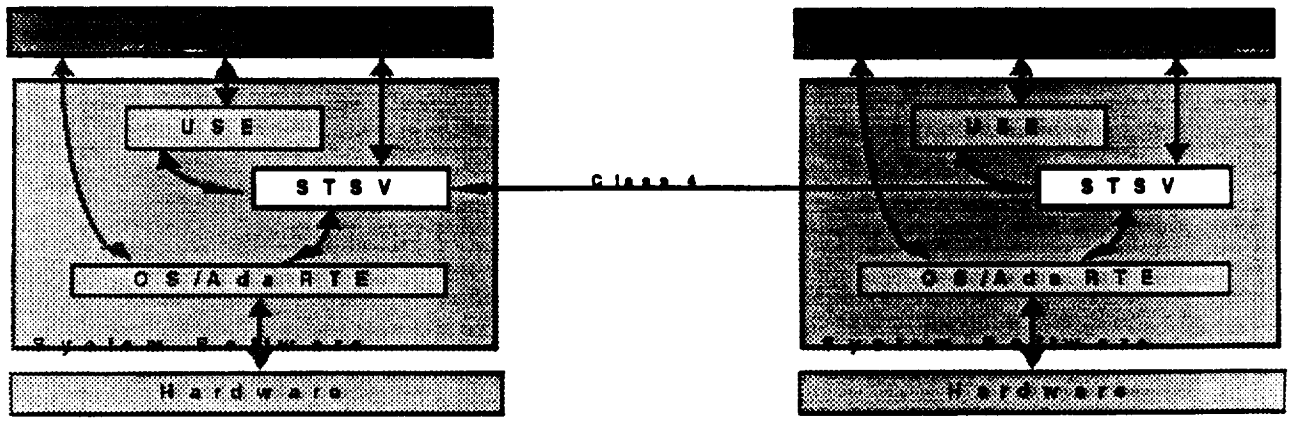


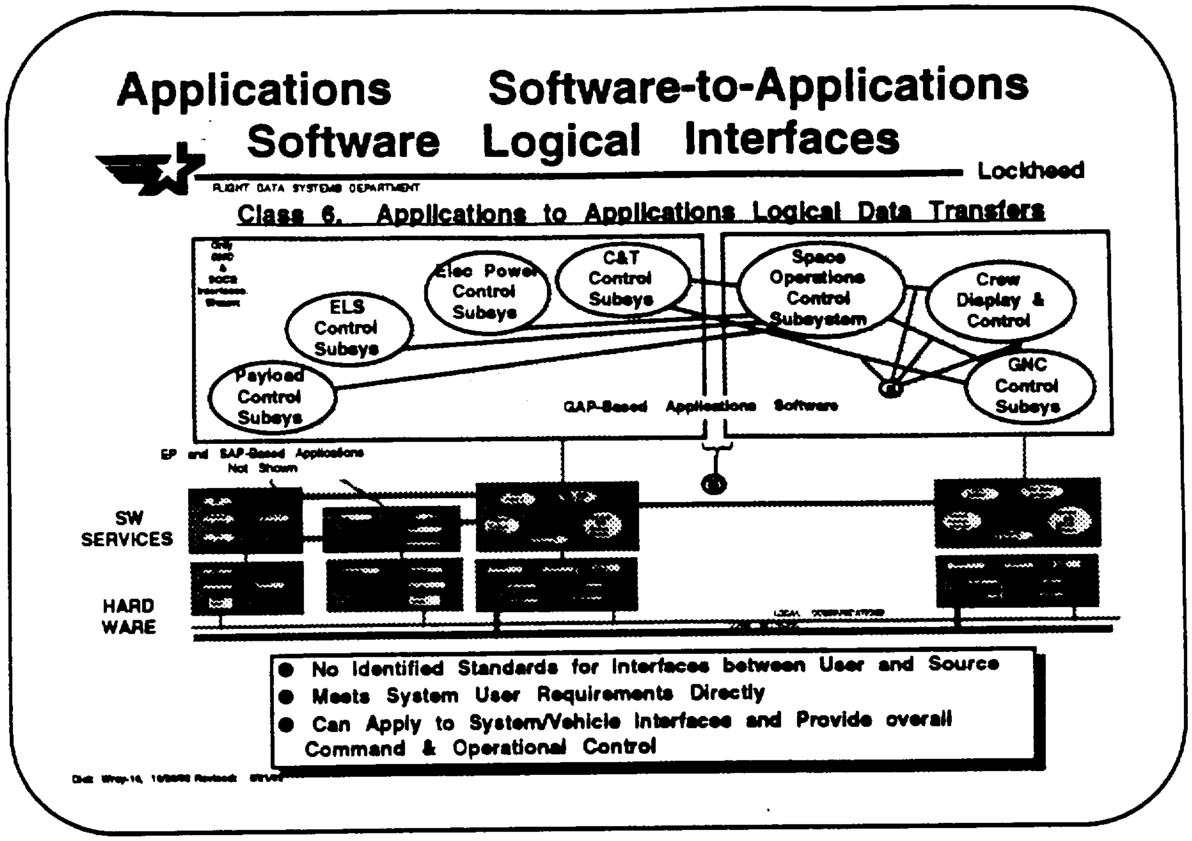
- **Lateral Interfaces between Service Using Data and Service Generating Data**
- **No Logical Architectural Difference between Local and Remote Services Access**
- **Meets Requirements for Applications Support - A Derived Requirement**

Disk Wrap-4, 10/30/92 Revised: 8/18/92

Notes:

- The system software services to remote system software interfaces are shown in this slide. This is the peer to peer interface of system software in one processing element (GAP,SAP or EP) interfacing with the system software in an external processing element to coordinate operations in a distributed environment. The grayed out parts of the figure represent the material covered in Classes 1 to 3, the white parts of the figure are the new material added in Class 4. Since Classes 1 to 3 isolated the hardware and software services in each processor, Class 4 adds the interface capability for services in one processor to interact with services in another processor; this is the heart of multi-processor capability needed in modern space avionics systems. GAP services can interact with EP and SAP services and other GAP services. These interfaces are logical interfaces because the service originating data is interacting with the service that will use the data (i.e., that will transform the data into another form for a purpose). Class 4 interfaces meet derived requirements based on the need of an application to support users in a multi-processing environment.
- The GAP to services interfaces are defined by the naming convention as GAPSRV- to indicate that GAP services would be specified by the standard interface specification, and could be broken out by subsequent lower level charts. An example from the station is shown below.





Applications Software-to-Applications Software Logical Interfaces

FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

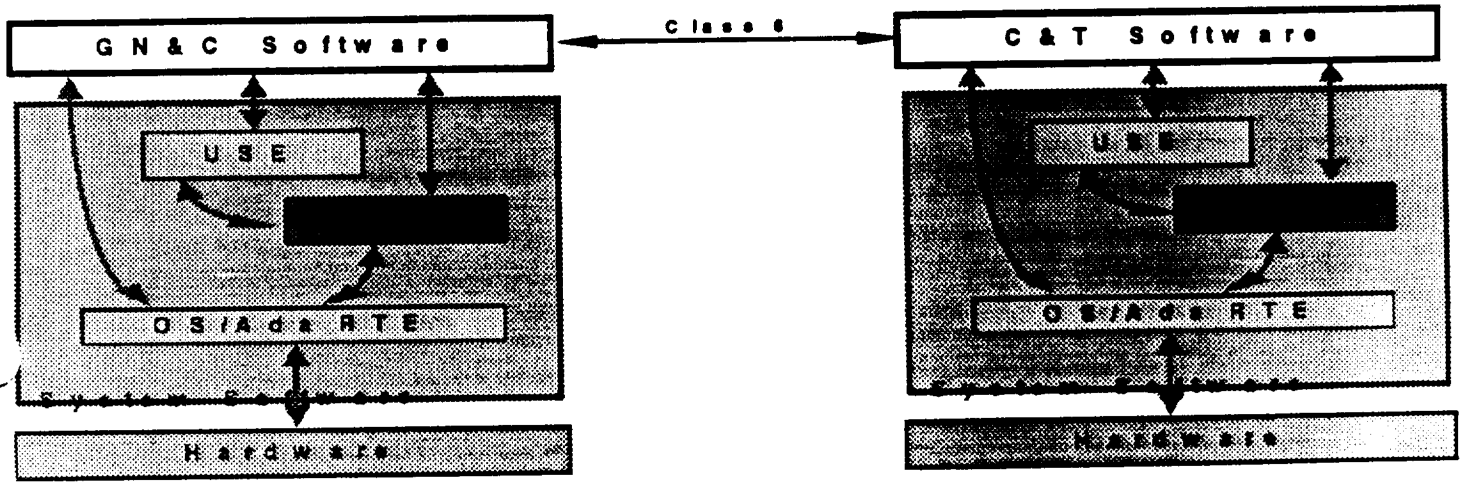
Class 6. Applications to Applications Logical Data Transfers

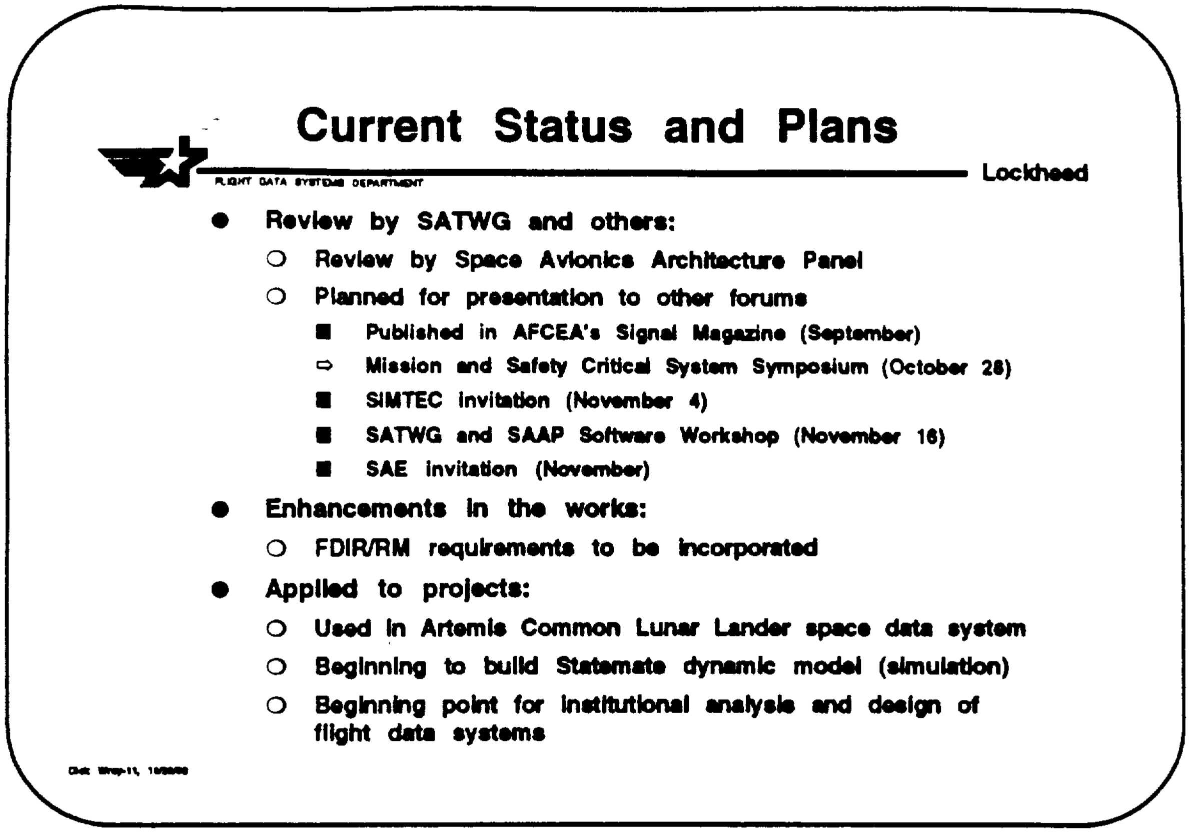
Elec Power Control Subsys
C&T Control Subsys
Space Operations Control Subsystem
Crew Display & Control
ELS Control Subsys
GNC Control Subsys
Payload Control Subsys
GAP-Based Applications Software
EP and SAP-Based Applications Not Shown
SW SERVICES
HARD WARE

- No Identified Standards for Interfaces between User and Source
- Meets System User Requirements Directly
- Can Apply to System/Vehicle Interfaces and Provide overall Command & Operational Control

Notes:

- The applications software to applications software interfaces are shown. This peer to peer information exchange and coordination interface between application software modules. Applications do not communicate directly; hence this is a logical interface. All communication is through a Class 5 (P) standard interface to System Services which provides the physical communications path between applications. This interface may be between applications within a processing element or between applications in separate processing elements. The grayed out parts of the figure represent the material covered in Classes 1 to 5, the white parts of the figure are the new material added in Class 6. Since Classes 1 to 5 isolated the hardware, software services and applications in any processor, Class 6 adds the interface capability for an application in any processor to interact with another application executing in any processor; this provides the basic multi-processor capability to meet multiple actual user requirements in processing. Applications can operate in any processor (i.e., GAP, SAP or EP), with cooperating interactions to support the needs of the users. The interfaces are logical interfaces because the application originating data is interacting with applications that will use the data (i.e., that will transform the data into a form useful to the user or to another application for a user's ultimate purpose). Class 6 interfaces meet user and derived requirements based on the need of multiple applications to support users in a multi-processing environment. The example below is provided from the station.





Current Status and Plans

FLIGHT DATA SYSTEMS DEPARTMENT

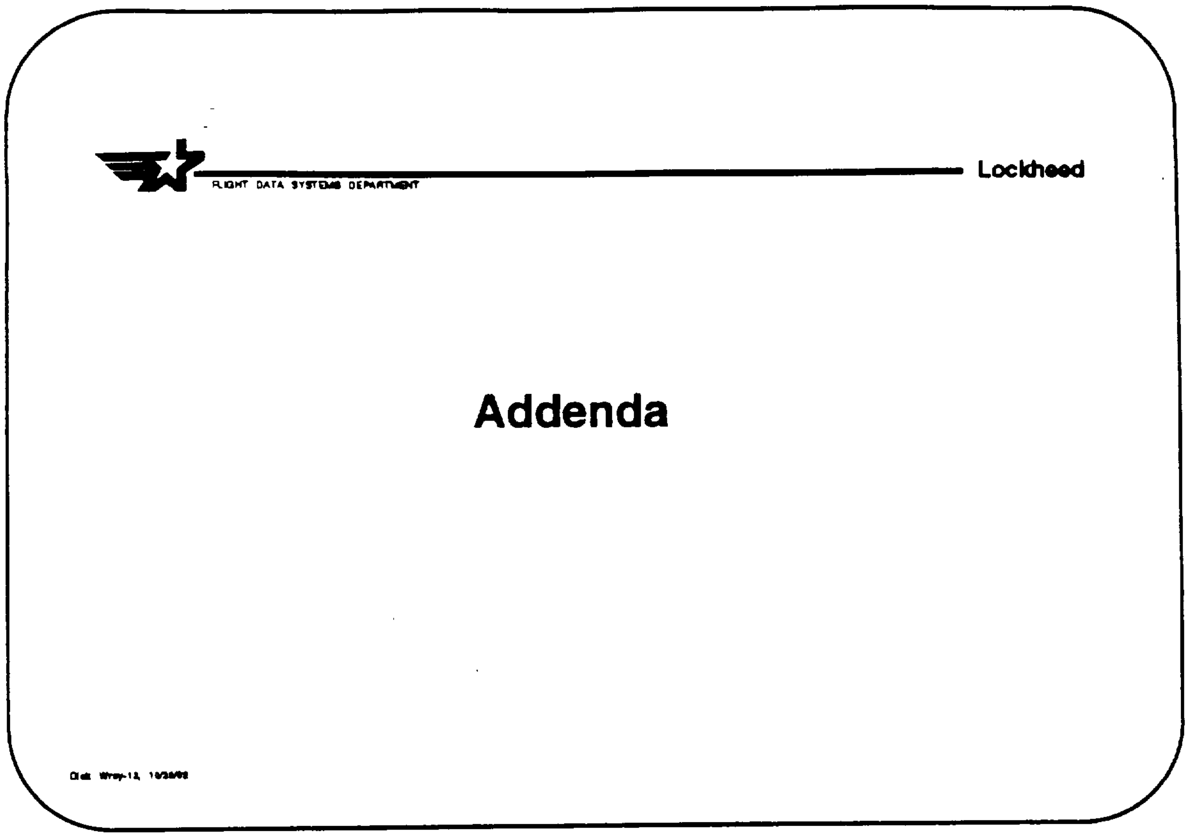
Lockheed

- Review by SATWG and others:
  - Review by Space Avionics Architecture Panel
  - Planned for presentation to other forums
    - Published in AFCEA's Signal Magazine (September)
    - ⇨ Mission and Safety Critical System Symposium (October 28)
    - SIMTEC invitation (November 4)
    - SATWG and SAAP Software Workshop (November 16)
    - SAE invitation (November)
- Enhancements in the works:
  - FDIR/RM requirements to be incorporated
- Applied to projects:
  - Used in Artemis Common Lunar Lander space data system
  - Beginning to build Statemate dynamic model (simulation)
  - Beginning point for institutional analysis and design of flight data systems

[illegible]

Notes:

- The Reference Architecture Model Must Be Based on Standards
- It Must Span Platforms for All Missions and Operational Requirements
- A Space Generic Open Avionics Architecture Must be Adaptable
- Avionics Control Structure Must be Integrated in an Architecture
- Support Tailoring to Multiple Missions
- The advanced avionics architectures must fit and extend the POSIX Open Systems Environment model
- The space generic avionics functional architecture was successfully applied to the Common Lunar Lander, with a preliminary design for the data system in 2 days
- The architecture interface model makes an explicit and rational model of how hardware and software interfaces should be defined
- A common advanced architecture for all future space platforms is feasible and achievable



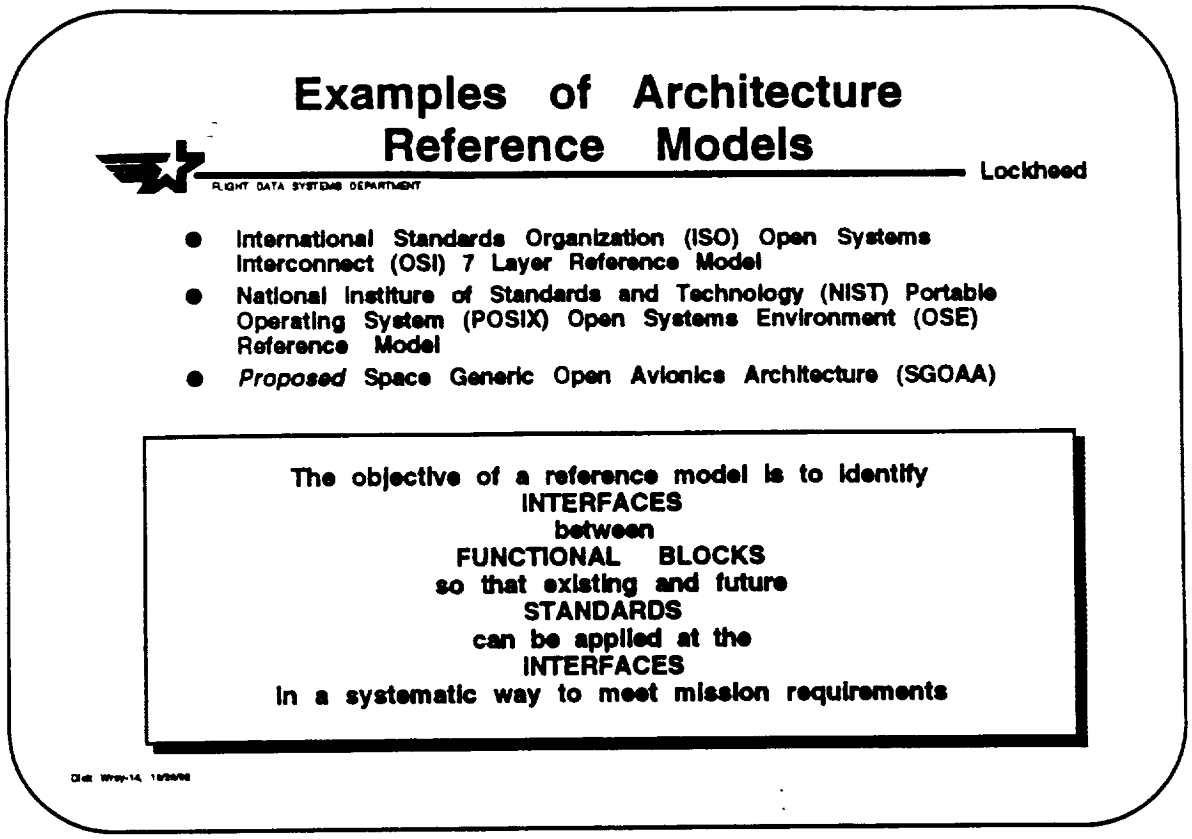
FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed

Addenda

Disk: Wray-12, 10/28/92

Notes:



Examples of Architecture Reference Models

Lockheed

FLIGHT DATA SYSTEMS DEPARTMENT

- International Standards Organization (ISO) Open Systems Interconnect (OSI) 7 Layer Reference Model
- National Institure of Standards and Technology (NIST) Portable Operating System (POSIX) Open Systems Environment (OSE) Reference Model
- *Proposed* Space Generic Open Avionics Architecture (SGOAA)

The objective of a reference model is to identify
INTERFACES
between
FUNCTIONAL BLOCKS
so that existing and future
STANDARDS
can be applied at the
INTERFACES
in a systematic way to meet mission requirements

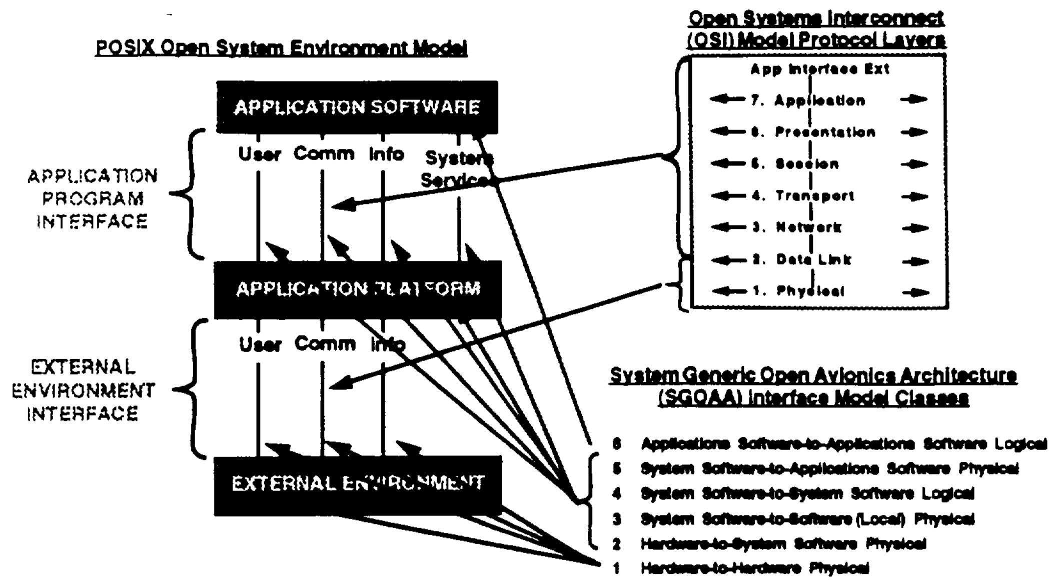
Dist. Wray-14, 10/30/92

Notes:

Relationships between OSI and SGOAA Models

FLIGHT DATA SYSTEMS DEPARTMENT

Lockheed


POSIX Open System Environment Model
APPLICATION SOFTWARE
APPLICATION PROGRAM INTERFACE
User
Comm
Info
System Services
APPLICATION PLATFORM
EXTERNAL ENVIRONMENT INTERFACE
User
Comm
Info
EXTERNAL ENVIRONMENT
Open Systems Interconnect (OSI) Model Protocol Layers
App Interface Ext
7. Application
6. Presentation
5. Session
4. Transport
3. Network
2. Data Link
1. Physical
System Generic Open Avionics Architecture (SGOAA) Interface Model Classes
6 Applications Software-to-Applications Software Logical
5 System Software-to-Applications Software Physical
4 System Software-to-System Software Logical
3 System Software-to-Software (Local) Physical
2 Hardware-to-System Software Physical
1 Hardware-to-Hardware Physical

Disk: Wray-15, 10/20/92 Revised: 6/18/92

Notes:

- The POSIX Open System Environment (OSE) Reference Model is the basis for the generic and open avionics architecture models, and for application software portability and interoperability. It can be related to the Open System Interconnect (OSI) model and the SGOAA interface model as shown in this slide. The OSE model communications link protocols are defined in detail by the OSI model for peer-to-peer communications. The OSE model interface classes are defined in detail by the SGOAA.
- The OSE Reference Model enables application software portability at the source code level and application software and system service interoperability between heterogeneous systems. Definition of entities and interfaces based on the OSE model can facilitate requirements definition for designs which have the open and generic characteristics needed.
- There are three types of entities used in the OSE Model: Application Software, Application Platform and External Environment. There are two types of interfaces: the Applications Program Interface and the External Environment Interface.
- The OSI Reference Model is a Network Services Model. Network Service is only one resource of many competing resource processes provided by both the POSIX and SGOAA Models. Applications gain access to POSIX Network Services via the POSIX API Communications Services Interface and to SGOAA Network Services via the SGOAA Class 5 Interface (Applications Software-to-System Services Software). In the OSI model, applications gain access to Network Services via an applications-to-services interface. Interfaces provided by Network Services must be open network interfaces, protocol independent and provide for network protocol interoperability. The POSIX OSE reference model focuses on the requirements of application portability and system interoperability at the source code level by addressing these objectives at the Applications Program Interface (API) and at the External Environment Interface (EEI). Internal Application Platform Interfaces are not addressed.
- The OSI model may be mapped into just the communications links of the POSIX OSE model API and the EEI to define the communications protocols. The SGOAA model may be mapped into the user, communications, information, and systems services links of the POSIX OSE model to define the content of all the interfaces. Thus, the three models are complementary.