National Aeronautics and
Space Administration

NASA CR-188290

Lyndon B. Johnson Space Center
Houston, Texas 77058

SPACE GENERIC OPEN AVIONICS ARCHITECTURE (SGOAA) STANDARD SPECIFICATION

June 1994

Richard B. Wray
John R. Stovall

(This revision supersedes LESC-30354-B, issued December 1993)

Prepared by:

Lockheed Engineering & Sciences Company
Houston, Texas

Task Order 060-EK-AB1
Contract NAS 9-19100

for

FLIGHT DATA SYSTEMS DIVISION
JOHNSON SPACE CENTER

LESC-30354-C

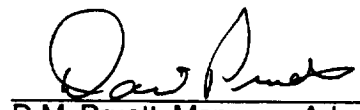
SPACE GENERIC OPEN AVIONICS ARCHITECTURE (SGOAA) STANDARD SPECIFICATION

June 1994

Richard B. Wray, Advanced Systems Engineering Specialist
John R. Stovall, Advanced Systems Engineering Specialist

APPROVED BY:

G.L. Clouette, Project Integration Specialist
Flight Data Systems Department



D.M. Pruett, Manager, Advanced Programs
Flight Data Systems Division

Prepared by:

Lockheed Engineering & Sciences Company
Houston, Texas

Subtask Order EK-AB1
Contract NAS 9-19100

for

FLIGHT DATA SYSTEMS DIVISION
JOHNSON SPACE CENTER

LESC-30354-C

PREFACE

This document has been produced by Mr. Richard B. Wray and Mr. John R. Stovall of Lockheed Engineering and Sciences Company (LESC), the codevelopers of the avionics architectures and standards represented in this document. The contributions of Mr. Ben Doeckel of LESC who participated in early development of the concepts for the avionics architectures and standards represented in this document is acknowledged. Special acknowledgment is also given to Mr. Dave Pruett of the National Aeronautics and Space Administration (NASA) Johnson Space Center (JSC) for his support of the Advanced Architecture Analysis, assistance in the development of the avionics architecture and constructive criticisms of the proposed standard. The Society of Automotive Engineers' (SAE's) Aeronautics Systems Division has been very helpful in fine tuning this architecture as they prepare their Generic Open Avionics Architecture version as an SAE standard.

DOCUMENT CHANGE RECORD

The following table summarizes the change activity associated with this document.

ISSUE AND DATE	CHANGE SUMMARY	SECTION

CONTENTS

Section **Page**

Section **Page**

Section **Page**

Section **Page**

TABLES

FIGURES

Figure **Page**

ACRONYMS

AP	Application Platform
API	Application Program Interface
BIT	Built-In-Test
BITE	Built-In-Test Equipment
C&T	Communications and Tracking
DBM	Data Base Manager
DPS	Data Processing Subsystem
DSS	Data System Services
DSM	Data System Manager
D&C	Display and Control
EE	External Environment
EEI	External Environment Interface
EIA	Electronics Industries Association
ELS	Environment and Life Support Subsystem
EP	Embedded Processor
EPS	Electrical Power Subsystem
FB+	Future Bus Plus
FDDI	Fiber Data Distribution Interface
GAP	General Avionics Processor
GNC	Guidance, Navigation and Control
GPPE	General Purpose Processing Element
I/O	Input/Output
ISA	Instruction Set Architecture
IOSM	Input/Output Data Services Manager
JSC	Johnson Space Center
LESC	Lockheed Engineering & Sciences Company

NASA	National Aeronautics and Space Administration
NSM	Network Services Manager
OS	Operating System
OSE	Open System Environment
OSI	Open Systems Interconnect
OSS	Operating System Services
POSIX	Portable Operating System Interface
RTE	Run Time Environment
SAE	Society of Automative Engineers
SAP	Special Avionics Processor
SDS	Space Data System
SDSS	Space Data System Services
SGOAA	Space Generic Open Avionics Architecture
SOCS	Space Operations Control Subsystem

1. INTRODUCTION

1.1 SCOPE

This standard establishes the Space Generic Open Avionics Architecture (SGOAA). The SGOAA includes a Generic Functional Model, Processing Structural Model and an Architecture Interface Model. This standard defines the requirements for applying these models to the development of spacecraft core avionics systems.

1.2 PURPOSE

The purpose of this standard is to provide an umbrella set of requirements for applying the generic architecture models to the design of a specific avionics hardware/software processing system. This standard defines a generic set of system interface points to facilitate identification of critical services and interfaces. It establishes the requirements for applying appropriate low level detailed implementation standards to those interfaces points. The generic core avionics functions and processing structural models provided herein are robustly tailorable to specific system applications and provide a platform upon which the interface model is to be applied.

The goal of the SGOAA is to facilitate the design and development of systems which provide:

- Applications portability and reuse
- System services portability and reuse
- Operating system portability and reuse
- Physical resource transparency and independence

1.3 APPLICATION GUIDANCE

This standard is intended to be used by both avionics system designers and avionics system implementors in the development of open systems architectures for avionics. The system under design shall [1] be expressed in the context of the System Architecture (see Section 5.1). The system under design shall [2] use generic system services as shown in the Generic Functional Models (see section 5.2). The system under design shall [3] be based upon the generic resource configuration shown in the Processing Structural Models (see Section 5.3) of this standard. The Architecture Interface Model (see section 5.4) shall [4] be directly

applied to identify the specific interfaces requiring application of lower level standards. The selection of specific lower level standards is dependent upon unique system requirements, but shall [5] be conducted in accordance with the guidelines provided in Section 4.2.

This architecture is scaleable and recursive, and can be applied to any hierarchical level of physical resource/software processing system, as discussed in Section 6.6.

1.4 BACKGROUND

Development of a SGOAA that satisfies the Portable Operating System Interface (POSIX) reference model [POSIX91], the Open System Interconnect (OSI) reference model [ISO7498] and the definition of an open system architecture was initiated to aid in providing the following benefits to future avionics programs:

- Provide the basis for establishing a set of specifications, standards and procedures that will become common to all systems used in simultaneously operational missions, e.g., to simplify interfaces between multiple vehicles (such as the shuttle and station) when performing a joint mission such as docking.
- Ensure that future avionics systems can be upgraded and maintained with minimal redesign impact to the existing avionics system by establishing the interfaces required to enable modular replacement of physical resources and software.
- Promote availability of multiple sources of needed avionics software and hardware (physical resources) by defining standard interfaces.
- Provide a pool of physical resources and software modules for multiple program re-use by defining standard interfaces and services, and promoting physical resource and software reuse and commonality.
- Insure access to the architecture and its design documentation for any vendor or agency desiring to propose new uses and applications, and to facilitate competition to contain cost growth.

A complete description of the SGOAA development model, technical considerations and application examples is contained in the technical guide [WRA93].

2. APPLICABLE DOCUMENTS

The following documents provide additional supplemental material applicable to this standard. They provide additional requirements or expand on requirements from this standard for generic open architectures.

2.1 STANDARDS

[ISO7498] "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", First Edition, International Standards Organization, October 1984.

[POSIX91] "Draft Guide to the POSIX Open Systems Environment", P1003.0/D14, IEEE Computer Society, November 1991.

[SYSB-1] "Systems Engineering", EIA Engineering Bulletin SYSB-1, Electronics Industries Association (EIA), December 1989.

2.2 SPECIFICATIONS

2.2.1 GOVERNMENT SPECIFICATIONS

[JSC 31000] Space Station Projects Description and Requirements Document, Vol. 3, Rev G, 4 April 1991.

[SSP 30235] Space Station Program Glossary, Acronyms and Abbreviations, CR BB007008A, No date.

[PAVE PILLAR] "Architecture Specification for PAVE PILLAR Avionics", SPA-90099001A, Aeronautical Systems Division, USAF, January 1987.

2.2.2 CONTRACTOR SPECIFICATIONS

2.3 OTHER PUBLICATIONS

[BOE91] Flanagan, Rich and Van Ausdal, Art, "SATWG Flight Data System Architecture Specification Outline" briefing, 25 October 1991.

[BOOCH87] Booch, Grady, "Software Engineering with Ada", 2nd Edition, Benjamin Cummings Publishing Comp., 1987.

[GD90A] General Dynamics "Space Avionics Requirements Study", 21 October 1990, Contract NAS8-37588, TD006 Presentation Package, as briefed to the SATWG.

[LAP90] Laprie, J. C., "Dependability: Basic Concepts and Terminology", J. C. Laprie - Editor, Published by International Federation for Information Processing (IFIP) Working Group 10.4 on Dependable Computing and Fault Tolerance, December 1990.

[WRA93] Wray, R.B. and Stovall, J.R., "Space Generic Open Avionics Architecture (SGOAA) Reference Model Technical Guide", Job Order 60-430, Contract NAS9-17900 for the JSC, NASA CR-188246, LESC-30347-A, April 1993. (Note: some of the figures in this reference have been superseded by the figures in revision C of the standard.)

3. CONVENTIONS AND DEFINITIONS

3.1 PURPOSE

Common document terminology's and general technical definitions are established herein for this standard. Definitions are taken or adapted from the [POSIX91] or [LAP90] where applicable or otherwise established as shown. Figures shown in this standard are an integral part of this standard.

3.2 DEFINITIONS

3.2.1 TERMINOLOGY

For the purposes of this standard, the following definitions apply:

3.2.1.1 Implementation Dependent

Implementation dependent means the implementation is to define and document the requirements for architectures, software constructs, correct data values or behavior. (Adapted from [POSIX91])

3.2.1.2 Informative

Informative is providing or disclosing information which is only instructive and not preceded by a "shall" (Adapted from [POSIX91]) Such material poses no requirements, and is primarily located in Section 6, Notes.

3.2.1.3 May

May indicates an optional feature, and can be interpreted in implementation as a feature that can be provided but is not required. (Adapted from [POSIX91])

3.2.1.4 Normative

Normative is prescribing or directing a norm or standard; used in standards to indicate text which poses requirements. (Adapted from [POSIX91]) Each section of this document is normative, except for section 6. Non-normative parts of each section are explicitly indicated.

3.2.1.5 Should

Should, in implementation, indicates a recommendation for implementors, but does not establish a requirement.

3.2.2 GENERAL TERMS

For the purposes of this standard, the following definitions apply:

3.2.2.1 Application

Application is the capability (service/function) provided by an information system specific to the satisfaction of a set of user requirements. [POSIX91]

3.2.2.2 Application Platform

Application Platform (AP) is the set of resources that supports the services on which an application or application software will run. Also known as a host platform. [POSIX91]

3.2.2.3 Application Program Interface

Application Program Interface (API) is the interface between the application software and the application platform, across which all services are provided. [POSIX91]

3.2.2.4 Application Software

Application Software is software that is specific to an application and is composed of programs, data and documentation. Application software has uniquely defined interfaces. [POSIX91]

3.2.2.5 Architecture

Architecture is defined for this standard as the structure of Application Software, API, AP, and External Environment Interfaces (EEIs) which describe the organization and interfaces of a system.

3.2.2.6 Availability

Availability is a measure of the probability that a designated system will delivery the correct service when called upon at any random point in time. [Adapted from LAP90]

3.2.2.7 Avionics System

Avionics System is defined for the purpose of this standard as the set of all electronic and processing based subsystems on a space vehicle, including all physical resources, software and other electronics needed to control and operate the space vehicle. It is the collection of system elements and allocated capabilities that provides the coordinated functionality for end-to-end processing in handling the information needed to interface the space vehicle's major components, to control its interaction with its environment, and to respond to human commands. (Adapted from [JSC 31000])

3.2.2.8 Communication Interface

Communication Interface is the boundary between applications and the external environment, such as applications on other host platforms, external data transport facilities and devices. The communications interface may be internal to one space vehicle or across multiple space vehicles. [POSIX91]

The services provided are those whose protocol state, syntax and format all must be standardized for interoperability.

3.2.2.9 Component

Component is one of the parts resulting when an entity is decomposed into constituent parts.

3.2.2.10 Continuity

Continuity means that requirements changes are proportional to design changes, i.e., that changes in the requirements will propagate into changes of the same order of magnitude in the design.

3.2.2.11 Control System

Control Subsystem is an application which selects and implements alternative actions based on a priori criteria or real time guidance.

3.2.2.12 Core Avionics

Core Avionics are the control subsystems and the supporting avionics (physical resources and software) needed to enable these control subsystems to function. Core avionics include the controls for each of the traditional space avionics hardware subsystems (such as Guidance Navigation and Control (GN&C) and Communications and Tracking (C&T). The avionics hardware sensors and effectors are outside the core avionics boundary.

3.2.2.13 Data

Data are the sensor outputs to the system, input to applications from the system, output from applications to the system, input to crew or operations control elements from the system, outputs from crew or operations control elements to the system. Data may include commands.

3.2.2.14 Data Base Manager

Data Base Manager (DBM) is the control subsystem which manages structured data files, file transfers and file redundancy management.

3.2.2.15 Data Processing Subsystem

Data Processing Subsystem (DPS) is an application subsystem providing data processing services. Data processing subsystems do not perform control subsystem functions.

3.2.2.16 Data System

Data System (for example the Space Data System (SDS)) is a network of system services, onboard computational resources, data storage, and human-machine interface devices which provide onboard command and control, data transmission, computation/processing, and operating capabilities to support a space vehicle's users (crew and controllers), interfacing systems, applications and subsystem

3.2.2.17 Data System Manager

Data System Manager (DSM) is the control subsystem which manages the housekeeping and status control services for the SDSS.

3.2.2.18 Data System Services

Data System Services (DSS) is a service subsystem with a generic functional architecture designed to provide a comprehensive set of services to all vehicles and subsystems (for example the Space Data System Services (SDSS)).

3.2.2.19 Decomposability

Decomposability means requirements can be broken into smaller pieces with potentially simpler solutions or at least better understanding and a capability for further decomposition as needed.

3.2.2.20 Degraded Mode

Degraded mode is a system condition wherein some system elements (such as physical resources, software, human, or procedural) are sufficiently unhealthy that the system cannot operate normally.

3.2.2.21 Dependability

Dependability is the trustworthiness of an avionics system and whether reliance can justifiably be placed on the service it delivers. Depending on the application(s) intended for the system, different emphasis may be put on different facets of dependability, i.e. dependability may be viewed according to different, but complementary, properties, which enable the attributes of dependability to be defined: (Derived from [LAP90]).

- with respect to the readiness for usage, dependable means available;
- with respect to the continuity of service, dependable means reliable;
- with respect to the avoidance of catastrophic consequences on the environment, dependable means safe;
- with respect to the prevention of unauthorized access and/or handling of information, dependable means secure.

3.2.2.22 Direct Interface

Direct Interface is the connection between an entity sending or receiving data with another entity receiving or sending data for transmission of the same data along the routing path associated with moving data from the source of the data to the end user of the data. Data is used by an entity in a direct manner if it passes the data on without changing the data; thus, for example, network operating systems are direct interfaces between applications when they package or unpack data and send it to another network node.

3.2.2.23 Distributed System

Distributed System is a collection of computers, memories, buses and networks that are concurrently operating in a cooperative manner and communicating with each other.

3.2.2.24 End-user

End-user of data is the last entity which makes a significant transformation, conversion or operation on the data.

3.2.2.25 Entity

Entity is an abstract element that represents an object in the real world, its data attributes and essential services with their respective performance and quality characteristics.

3.2.2.26 Error

Error is defined to be that part of the system state which is liable to lead to subsequent failure. [LAP90]

3.2.2.27 Error Processing

Error Processing is the action taken in order to eliminate errors from a system. Error processing involves error detection followed by either error recovery or error compensation. Error recovery replaces an error-free state for the erroneous one. Error compensation uses the redundancy of the state to enable the delivery of an error free service from the erroneous (internal) state. [LAP90]

3.2.2.28 External Environment

External Environment (EE) is a set of external entities with which the application platform exchanges information. These entities are classified into the general categories of human users, information interchange entities and communication entities. [POSIX91]

3.2.2.29 External Environment Interface

External Environment Interface (EEI) is the interface between the application platform and the EE across which information is exchanged. The EEI is defined primarily in support of system and application interoperability. This interface consists of human/computer interaction services, information services, and communications services. [POSIX91]

3.2.2.30 Extensibility

Extensibility is the ability of an architecture to be extended or adapted to new conditions, changes in specifications or new technologies.

3.2.2.31 Failure

Failure is a deviation of the delivered service from the specified service, where the service specification is an agreed description of the expected function and/or service. [LAP90]

3.2.2.32 Fault

Fault is defined as the adjudged or hypothesized cause of an error. [LAP90]

3.2.2.33 Fault Tolerance

Fault Tolerance is the capability for providing a service to ensure compliance with the specification, in spite of faults. Fault tolerance is carried out by error processing and fault treatment. Error processing is aimed at removing errors from the system state, if possible before failure occurrence; fault treatment is aimed at preventing faults from being activated again. [LAP90]

3.2.2.34 Fault Treatment

Fault Treatment is the action taken in order to prevent a fault from being re-activated. The first step in fault treatment is fault diagnosis, which consists of determining the cause(s) of error(s), in terms of both location and nature. This is followed by fault passivation, which prevents the fault from being activated again. If the system is no longer capable of delivering the same service as before, then a reconfiguration may take place. [LAP90]

3.2.2.35 Flight Critical Function/Interface

Flight Critical Function is a function or interface which, if it fails, could cause loss of vehicle control resulting in loss of the vehicle and, if present, crew. The function or interface is characterized by the presence of hard deadlines (usually in the range of milliseconds), where missing a deadline is a failure.

3.2.2.36 Function

Function is an action/task that the system must perform to satisfy customer and end user needs. Control of mission critical functions may require hard deadlines, where missing a deadline is a failure.

3.2.2.37 Generic Architecture

Generic Architecture is an architecture where the elements of the architecture do not depend on any one mission or program for their definition. The elements of a generic architecture can be tailored to apply to many different missions and programs.

3.2.2.38 Hardware

Hardware is physical equipment used in data processing as opposed to programs, procedures, rules and associated documentation. [POSIX91]

3.2.2.39 Human/Computer Interface

Human/Computer Interface is the boundary across which direct interaction between a human being and the application platform take place.

3.2.2.40 Input/Output Data Services Manager

Input/Output Data Services Manager (IOSM) is the interface handling subsystem that manages the services that process requests for interaction between sensors, effectors, applications and other services.

3.2.2.41 Interface

Interface is the shared boundary between two functional units, defined by functional and other physical characteristics, as appropriate.

3.2.2.42 Interoperability

Interoperability is the ability of two or more systems to exchange information and to mutually use the information that has been exchanged. [POSIX91]

3.2.2.43 Logical Interface

Logical Interface is the requirement associated with establishing a data interchange between a source of data and the end user of the data. The end user of the data must be identified to include the requirements for the data and the source supplying the data must also be identified. Data routing is transparent to logical interface entities. Routing of the data should not be a concern to the source and end user because the routing (i.e., direct requirements) is transparent to these entities.

3.2.2.44 Mission Critical Function/Interface

Mission Critical Function or Interface is any function or interface which, if it fails, results in an incomplete mission, a mission abort or a loss of payload.

3.2.2.45 Mission Ready Mode

Mission Ready Mode is a system condition wherein all system elements, including physical resources, software, human and procedural, are available to enable the system to perform its intended function and the current mission for which it is intended.

3.2.2.46 Mode

Mode is a predefined set of physical resource and software configurations, and associated procedures used to organize and manage the conditions of operation for an avionics system's behavior, as planned, pre-planned or directed by a human.

3.2.2.47 Modular Architecture

Modular Architecture is an architecture composed of discrete components such that the design of one component depends only on the interface to other components, not on their internal design. A modular architecture is decomposable, understandable, protected, has continuity and is organized in a robust structure. It is desirable that a change in one component has minimal impact on other components. (Adapted from [SSP 30235]).

3.2.2.48 Network Services Manager

Network Services Manager (NSM) is a control subsystem which manages peer-to-peer communication between applications running on distributed processing elements communicating over a network.

3.2.2.49 Object

Object is something perceptible to the sense of vision or touch or to the mind.

3.2.2.50 Onboard Health Management

Onboard Health Management is the physical resource and software used to monitor and control on board avionics system resources to prevent or respond to system failure. This includes the ability to efficiently monitor, checkout, and test the avionics system, core avionics, and related non-avionics subsystems before, during, and after operation, as applicable. Onboard health management supports, as required, reconfiguration of avionics system resources to prevent catastrophic failure.

3.2.2.51 Open Forum

Open Forum is the review of a subject in a public consensus process.

3.2.2.52 Open Specification

Open Specifications are public specifications that are maintained by an open, public consensus process to accommodate new technologies over time and that are consistent with international standards. The public consensus process for open specifications must be maintained and accepted by an open forum. [POSIX91]

3.2.2.53 Open System

Open System is a system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications: [POSIX91]

- to be ported with minimal changes across a wide range of systems
- to interoperate with other applications on local and remote systems
- to interact with users in a style that facilitates user portability

3.2.2.54 Open System Interface Standards

Open System Interface Standards are standards that provide for open specifications of open systems.

3.2.2.55 Open System Application Program Interface

Open System Application Program Interface is a combination of standards-based interfaces specifying a complete interface between application software and the underlying application platform. This is divided into the following parts: [POSIX91]

- Human/Computer Interaction Services API
- Information Services API
- Communication Services API
- System Services API

3.2.2.56 Open Systems Architecture

Open Systems Architecture is an architecture for an open system using open specifications. It consists of a structure of interconnected functional subsystems (i.e., black boxes) using non-proprietary communications, based on open specifications for interfaces, and providing high level system services. The interface between the application software and the

underlying application platform must be based on an Open System Application Program Interface. To be open, the architecture must be extensible through the addition of subsystems, services and resources following open specification rules.

3.2.2.57 Open System Environment

Open System Environment (OSE) is the comprehensive set of interfaces, services and supporting formats, plus user aspects for interoperability or for portability of applications, data, or people, as specified by information technology standards and profiles. [POSIX91]

3.2.2.58 Operating System Services

Operating System Services (OSS) is the layer that isolates the DSS and applications from the application platform physical resources. The OSS provides services for at least management, allocation, and deallocation of the processor, memory, timing and input/output (I/O) processing resources for applications and DSS.

3.2.2.59 Operationally Ready Mode

Operationally Ready mode is a system condition wherein most system physical resource, software, human and procedural elements are functioning correctly, but not all subsystems are configured as needed for a mission to be performed.

3.2.2.60 Physical Resources

Physical resources are those functions based in hardware in the application platform.

3.2.2.61 Platform

See Application Platform definition.

3.2.2.62 Portability

Portability is the ease with which software can be transferred from one platform, application or information system to another. [POSIX91]

3.2.2.63 Profiling

Profiling is the process of selecting a set of one or more base standards, and where applicable, the identification of chosen classes, subsets, options, and parameters of those base standards, necessary for accomplishing a particular function. (The profile selection process is discussed in section 6 of [POSIX91]).

3.2.2.64 Protection

Protection means that the architecture will limit the effect of abnormal conditions in design elements at run-time to just the affected modules or as a minimum will limit the propagation of abnormal conditions.

3.2.2.65 Protocol

A Protocol is a set of semantic and syntactic rules that must be followed to perform communications functions within a communications system. (Adapted from [POSIX 91].

3.2.2.66 Red-tagged Mode

Red-tagged mode is a system condition wherein sufficient system physical resource, software, human or procedural elements are failed that the system cannot operate at all.

3.2.2.67 Reliability

Reliability is a measure of the probability that an item will deliver the correct service under specified conditions without failure, for a specified period of time.

3.2.2.68 Resource Access Services

Resource Access Services are those low level services which enable the DSS to interact with physical resources.

3.2.2.69 Requirements Architecture

Requirements Architecture is an architecture that can be tailored for design implementation based on actual system requirements.

3.2.2.70 Robustness

Robustness is the measure of a system's ability to support continued functioning under abnormal operating conditions.

3.2.2.71 Safety Critical Function

Safety Critical Function is any function which has an associated condition, event, operation, process, equipment or system (including software) with the potential for catastrophic injury or damage to onboard systems, life, or environment. (adapted from [SSP 30235] and [LAP90].

3.2.2.72 Service

Service is the work performed for a user by subsystem or application software.

3.2.2.73 Service Subsystem

Service Subsystem is software on an application platform which provides transparent services to the using control or data processing subsystem.

3.2.2.74 Software

Software is the programs, procedures, rules, and any associated documentation pertaining to the operation of a data processing system. [POSIX91]

3.2.2.75 Source

Source is the originator of data passed across a logical interface.

3.2.2.76 Space Data System

Space Data System (SDS) - See Data System definition.

3.2.2.77 Space Data System Services

Space Data System Services (SDSS) - See Data System Services definition.

3.2.2.78 Space Generic Open Avionics Architecture

SGOAA is the target open architecture standard which provides an umbrella set of requirements for applying a generic architecture interface model to the design of specific avionics physical resource/software systems. This standard defines a generic set of system interface points and establishes the requirements for applying appropriate low level detailed implementation standards to those interfaces points. The generic core avionics system and processing physical resource architecture models provided by the standard are robustly tailorable to specific system applications and provide a platform upon which the generic interface model is to be applied.

3.2.2.79 Space Operations Control Subsystem

The Space Operations Control Subsystem (SOCS) is the high level integrating command and control functional entity for a space vehicle and mission. SOCS functions may be allocated to both ground mission control facilities and onboard space vehicle facilities.

3.2.2.80 Standard

Standard is a document established by consensus and approved by a recognized body, that provides, for common and repeated use, rules, guidelines, or characteristics for activities or their results, aimed at the achievement of the maximum degree of order in a given context.

3.2.2.81 Standardized Profile

Standardized Profile is a balloted formal, harmonized document that specifies a profile. [POSIX91]

3.2.2.82 System

System is the composite of equipment, material, computer software, personnel, facilities and information/procedural data that satisfies a user need. [SYSB-1]

3.2.2.83 System Physical Resources Architecture

System Physical Resource Architecture is an architecture consisting of the set of physical resources in a configuration of distributed computers, memories, buses and network elements.

3.2.2.84 System Software Architecture

System Software Architecture is an architecture consisting of the elements and interfaces between software components in a system.

3.2.2.85 System Services Software

System Services Software is common software, independent of application software, which is needed to run application software and enable it to interface to data within a system or across the EEI. This is similar to the POSIX entity, system software, which is defined as the application independent software that supports the running of application software.

3.2.2.86 Task

Task is a software entity that is executed in parallel with other parts of a software program to perform an action. [BOOCH87]

3.2.2.87 Understandability

Understandability means all requirements related to a subject can be found and viewed together, and individually and jointly understood by the analysts and designers.

3.2.2.88 User

User is another system (human or physical) which interacts with the target system.

4. GENERAL REQUIREMENTS

The SGOAA shall be used to determine the [1] initial functional services, [2] interface points and [3] processing structural requirements for the control of, and information exchange between, onboard subsystems, support to the crew, and effective interaction with offboard systems. In accordance with system requirements, a SGOAA compliant architecture shall [4] meet open standards criteria. A SGOAA compliant system architecture shall [5] provide data acquisition, data storage, data processing and data communication functions that interconnect architectural elements as shown in the functional interface diagram, Figure 4-1. Architectures developed in accordance with this standard shall [6] meet the following general requirements for developing new architectural elements and for using existing applications and mission elements.

4.1 ARCHITECTURE SYSTEMS REQUIREMENTS

An architecture developed in accordance with this standard shall [1] satisfy the open systems architecture definition incorporated in this standard. The open architecture so developed shall [2] be capable of being readily expanded in functionality and performance without redesign or significant modification to the existing system. An architecture satisfying this standard shall [3] provide information hiding, abstraction, inheritance, modularity, robustness and extensibility.

Control subsystems may be decomposed into lower level subsystems. A control subsystem usually implements a unique avionics capability. These control subsystems may have flight, mission, or safety critical functions.

An architecture prepared in accordance with this standard shall [4] be a requirements architecture, i.e., one that can be tailored for design implementation based on actual system requirements.

4.2 LOWER LEVEL STANDARDS SELECTION

Lower level standards developed by accredited standards development organizations (which use an open forum) shall [1] be preferred in selection over those standards developed by bodies using a closed forum. Lower level standards shall [2] be selected by the process of developing a standardized profile. Architecture specifications for which there is

no draft or approved standard shall [3] not be selected. One of the driving requirements for selection shall [4] be selection of a standard that provides the full range of services required to satisfy the system applications. Other factors to consider in standards selection shall [5] be degree of openness in development, stage of completion, stability, compliance with national and international standards, degree of satisfying a SGOAA service need, consistency with the SGOAA and availability for implementation without restrictions.

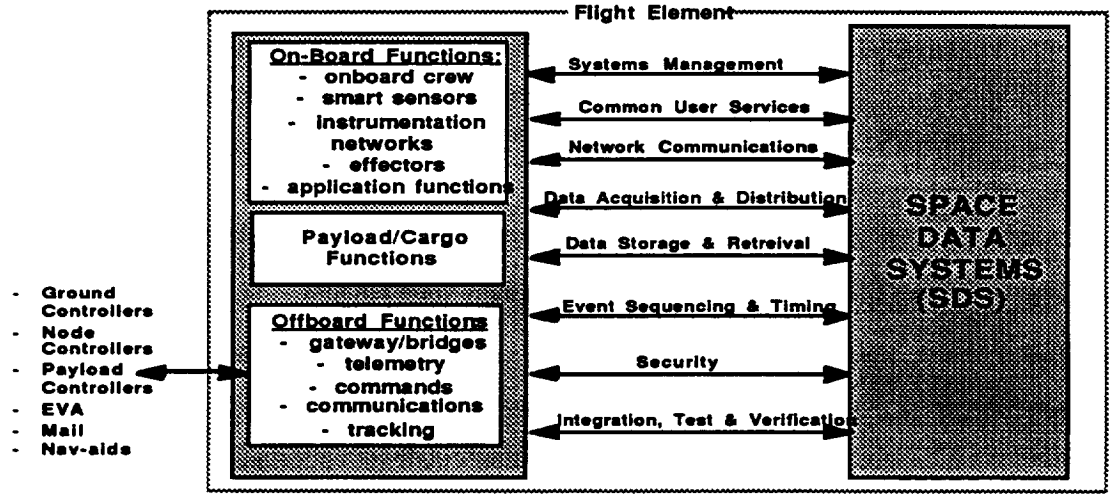


Figure 4-1. SGOAA Functional Interfaces

Preference shall [6] be given to existing mature standards, followed by emerging standards, and only if necessary, followed by new standards. The order of selection within these preferences is as follows:

- Approved standards developed by (a) accredited international bodies, (b) accredited regional bodies and (c) accredited national bodies.
- Draft standards developed by (a) accredited international bodies, (b) accredited regional bodies and (c) accredited national bodies.
- Recognized de facto standards and specifications developed by nonaccredited bodies using an open forum.
- Approved standards and specifications developed by nonaccredited international standards bodies using a closed forum.
- Approved standards and specifications developed by nonaccredited national standards bodies using a closed forum.

4.3 ARCHITECTURE FEATURES

An Architecture prepared in accordance with this standard shall [1] provide the following features.

4.3.1 CRITICAL INTERFACES

An architecture prepared in accordance with this standard shall support [1] flight, [2] mission and [3] safety critical functions and interfaces, as required.

4.3.2 NON-CRITICAL INTERFACES

An architecture prepared in accordance with this standard shall [1] support non-critical support functions and interfaces, as required.

4.3.3 INTERFACE STANDARDIZATION

An architecture prepared in accordance with this standard shall [1] provide standard interfaces and shall [2] also allow user definable interfaces where no standards exist or a standard is not applicable. Interfaces between physical resources and other physical resource entities shall [3] be based on standards. Interfaces between physical resources and software shall [4] be based on standards. Interfaces between system services and applications shall [5] be based on standards. The following interfaces shall [6] be prohibited in an architecture compliant with this standard: (1) direct, non-service task to task communications, and (2) applications to applications direct information exchanges, which bypass use of system services.

4.3.4 CREW OVERRIDE

For crewed vehicles, an architecture prepared in accordance with this standard shall [1] enable crew intervention, through multiple techniques, to safely override or inhibit automatic flight, mission or safety critical functions. For uncrewed vehicles, the architecture shall [2] enable ground control station intervention to safely override or inhibit flight, mission or safety critical functions.

4.3.5 DATA SYSTEM SERVICES

An architecture prepared in accordance with this standard shall [1] include requirements for Data System Services (DSS). This shall [2] consist of consideration of at least requirements for input/output data services management, network services management, data base management, data system management, and an operating system.

4.3.6 RESOURCE CONTROL

An architecture prepared in accordance with this standard shall [1] provide for control of the system resources that are used for control and information processing in onboard systems by use of system services as requested by applications through a standard interface.

4.3.7 ONBOARD HEALTH MANAGEMENT

An architecture compliant with this standard shall [1] provide at least health management, status monitoring and warning capability to monitor critical functions in onboard systems, subsystems, components and crew and shall [2] provide avionics system level error recovery and fault treatment for non-critical hard deadline functions. Fault tolerance shall [3] be carried out by error processing and fault treatment. System service built-in-test (BIT) to include error detection, processing and recovery and fault treatment shall [4] be incorporated into software control modules. Hardware built-in-test equipment (BITE) to include error detection, processing and recovery and fault treatment shall [5] be incorporated into physical resource modules. Error processing and recovery and fault treatment for flight critical functions shall [6] be performed at the system services and/or physical resource module level. The interface between hardware BITE and health and status applications shall [7] be through system services. Error detection, processing and recovery and fault treatment shall [8] be timely enough to prevent loss of critical functions.

At least two levels of health management, status and warning capability may be provided in compliant architectures: first are applications prepared for the user's platform with knowledge of the mission, system and user goals; second are services which utilize standard health management capabilities (i.e., in DSS).

On board health management that controls allocation of avionics system resources shall [9] be implemented in applications where knowledge of the mission or specific system is unique and cannot be entered into the table-driven health management DSS. The interface

between the reconfiguration physical resource and the controlling applications shall [10] be through standard DSS. Applications performing on board health management shall [11] be capable of overriding reconfiguration decisions made by fault tolerance functions provided in Data System Management under DSS.

An architecture compliant with this standard shall [12] provide operating modes for at least: (1) mission ready, (2) operationally ready, (3) degraded, and (4) red-tagged.

4.4 ARCHITECTURE QUALITIES

An architecture prepared in accordance with this standard shall [1] provide the following qualities.

4.4.1 COMMONALITY

An architecture shall [1] be comprised of common physical resource and software components to the maximum possible extent.

4.4.2 GROWTH AND SPARE CAPACITY

An architecture prepared in accordance with this standard shall [1] accommodate growth and spare capacity in data storage, processing throughput, external communications throughput, input/output and additional sensors/effectors as required by system documentation.

4.4.3 MODULARITY

An architecture prepared in accordance with this standard shall [1] be modular.

4.4.4 SERVICE TRANSPARENCY

An architecture prepared in accordance with this standard shall [1] be implemented with sufficient transparency that the user will have visibility into the operation of services, but not necessarily the implementation of services.

4.4.5 TECHNOLOGY TRANSPARENCY

An architecture prepared in accordance with this standard shall [1] be implemented with sufficient transparency that technologies applied to design can be upgraded without revising the architecture and without negative impact on the user.

4.4.6 INTEROPERABILITY

An architecture prepared in accordance with this standard shall [1] support interoperability by providing standard interfaces between multiple programs.

4.4.7 DEPENDABILITY

An architecture prepared in accordance with this standard shall [1] meet dependability requirements in a manner that supports standard interfaces, commonality, modularity and interoperability. Such an architecture shall [2] further satisfy the following subparagraphs.

4.4.7.1 Availability

An architecture compliant with this standard shall [1] be designed to satisfy the specified availability requirements of the designated system.

4.4.7.2 Reliability

An architecture compliant with this standard shall [1] be designed to satisfy the specified reliability requirements of the designated item.

4.4.7.3 Safety

An architecture compliant with this standard shall [1] provide an interface not dependent upon avionics system specific safety features for applications that are required to be portable.

4.4.7.4 Security

An architecture compliant with this standard shall [1] provide an interface not dependent upon avionics system specific security features for applications that are required to be portable.

5. ARCHITECTURE DETAILED REQUIREMENTS

The SGOAA models are based on partitioning between logical and direct requirements as illustrated in Figure 5-1. The SGOAA models are established to include architectural functions, interfaces and processing structures for all avionics systems. This SGOAA requirements description addresses system services for the Space Data System. This description also identifies applications examples in the Operations Control Subsystem. Interfaces in this model are valid for both one platform, multiple platforms and architectures on one or more vehicles.

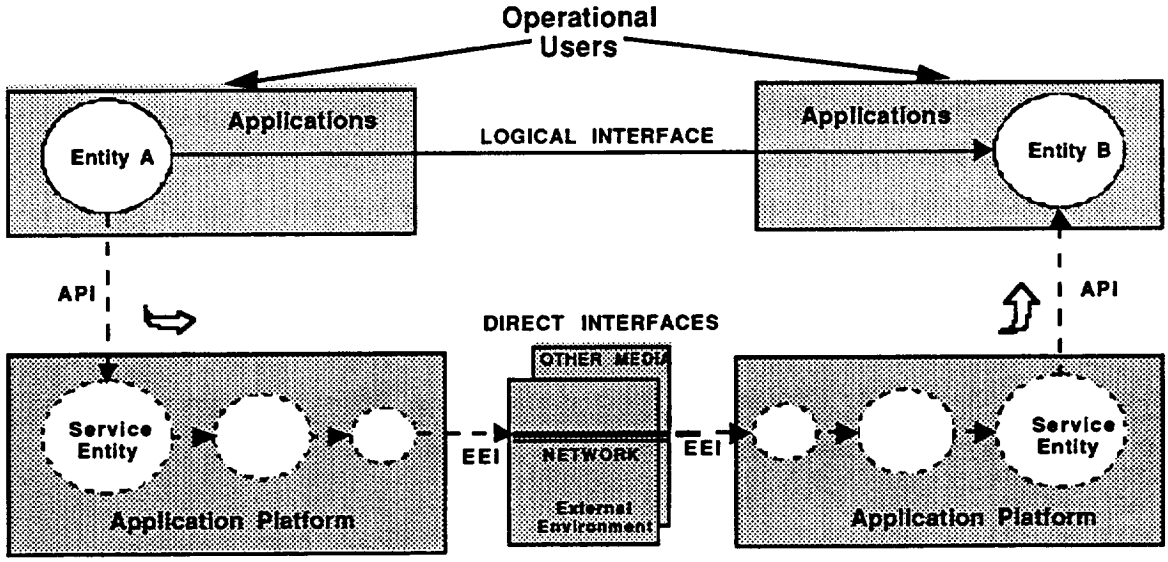


Figure 5-1. Logical System Requirements Flowdown to Direct Design Requirements

The SGOAA Models are to be used to define how system requirements are to be applied at the appropriate system level to determine the logical and direct interface points. High level logical interface requirements are captured in applications meeting operational user needs. Direct interface (for derived) requirements are established for services which implement the logical requirements. Low level logical interface requirements can subsequently be derived between services. System logical data flow requirements should be created for each client/server entity addressing the data attributes needed by that entity or needed to be provided for some other entity. The logical data flow requirements should identify the source of the data and the end-user needing the data, as well as the characteristic attributes required of the data. Logical data flow requirements should not be concerned with the mechanism for implementing the data interchange. Implementation related requirements for the interfaces are a direct interface issue relating to the mechanisms provided for

flowing the data from the source to the end-user. Sources of the design requirements for the interfaces, application platform physical resource and application platform services should be derived from the applications requirements and their logical data attribute requirements based on the operational user's needs.

5.1 SYSTEM ARCHITECTURE REQUIREMENTS

The SGOAA System Architecture, as shown in Figure 5-2, shall [1] form the basis for creating a model of the system under development.

System architecture models shall [2] consist of a functional definition of the types of processors and communications paths required. The model shown in Figure 5-2 has three types of processors interconnected by two types of communications. This model only shows one of each type of processor; the number of instances of each type of processor is variable depending upon system unique requirements and may range for 0 to n. For example, a centralized system architecture may look just like Figure 5-2, while a distributed system architecture may have multiple General Avionics Processor (GAPs), Special Avionics Processor (SAPs) and Embedded Processor (EPs). Either type of architecture may have many system interconnects and/or local interconnect mechanisms. More than one sensor and effector will usually be the rule in most non-trivial systems.

The processors shown in the system architecture in Figure 5-2 are a GAP for general purpose processing, a SAP for specialized processing support (vector/massively parallel/other), and an EP for the function of processing data within the sensor and effector devices. The sensors and effectors shown in the example may also interact directly with the main processors (the GAPs) or indirectly through EPs built into the sensors and effectors (if applicable).

Communications paths illustrated are of three types: system interconnects such as core networks for interconnecting sets of general processors or nodes, local interconnects such as local buses for interconnecting EPs and SAPs with their supported GAPs and general purpose processing applications, and internal interconnects such as backplane buses. System models shall [2] follow the general format of Figure 5-2, but shall [3] be tailored to match individual system requirements, in particular the program's application of sizing to the "system".

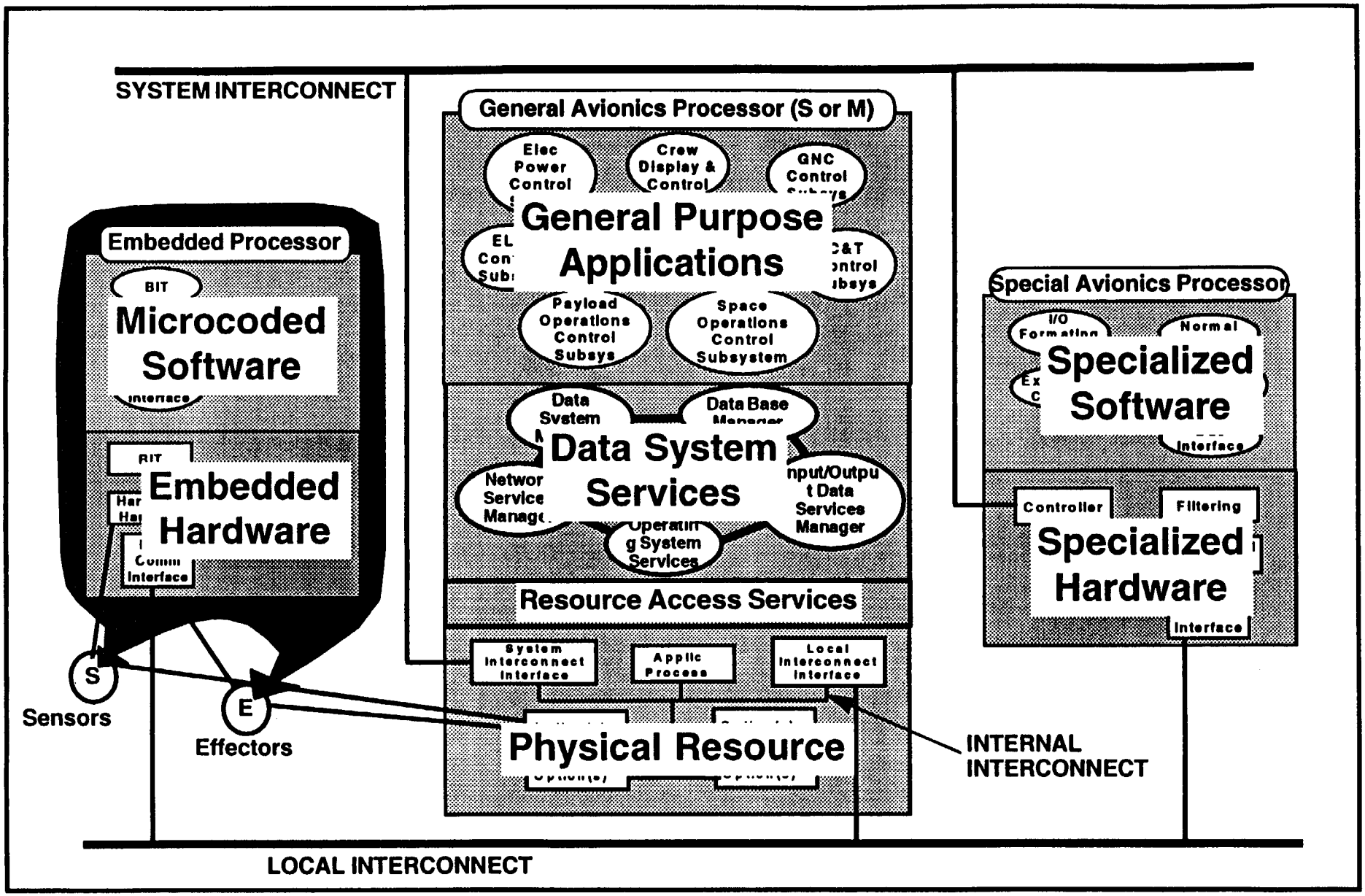


Figure 5-2. System Architecture

5.2 GENERIC FUNCTIONAL MODEL REQUIREMENTS

An architecture compliant with the SGOAA Generic Functional Model requirements shall [1] include consideration of capabilities drawn from at least five categories of services: the Data System Manager (DSM), Data Base Manager (DBM), Input/Output Data Services Manager (IOSM), Operating System Services (OSS), and Network Services Manager (NSM), as shown in Figure 5-3.

5.2.1 DATA SYSTEM SERVICES ARCHITECTURAL REQUIREMENTS

Interfaces from external entities to the DSS shall [1] be as shown in Figure 5-3. Control of the data system resources shall [2] flow at least through the DSM to ensure coordination of the system configuration at all points for reliability. Input and output sensor data shall [3] flow at least through the IOSM to ensure data handling consistency and reliability. Crew display and control (D&C) shall [4] be capable of operating through at least the IOSM and the DSM to insure at least one normal and one alternative path for direct low level system command and control by the crew. Similarly, operations control shall [5] be capable of operating through at least the IOSM and the DSM to insure at least one normal and one alternative path for direct low level system control by the ground or mission control. Access by applications will be as required by the system requirements documents.

The DSS architecture for the GAP shall [6] include consideration for at least the services as organized and shown in Figure 5-4. Implementation requirements for such services will depend on the system requirements documents.

The DSS architecture for the SAP shall [7] include consideration of capabilities drawn from at least five categories of services: BIT, I/O handling, normalization, specialized processing, and local communications. The system services for the EP shall [8] include consideration of capabilities drawn from at least the four categories of services: BIT, hardware handling, local communications and microprocessor execution control.

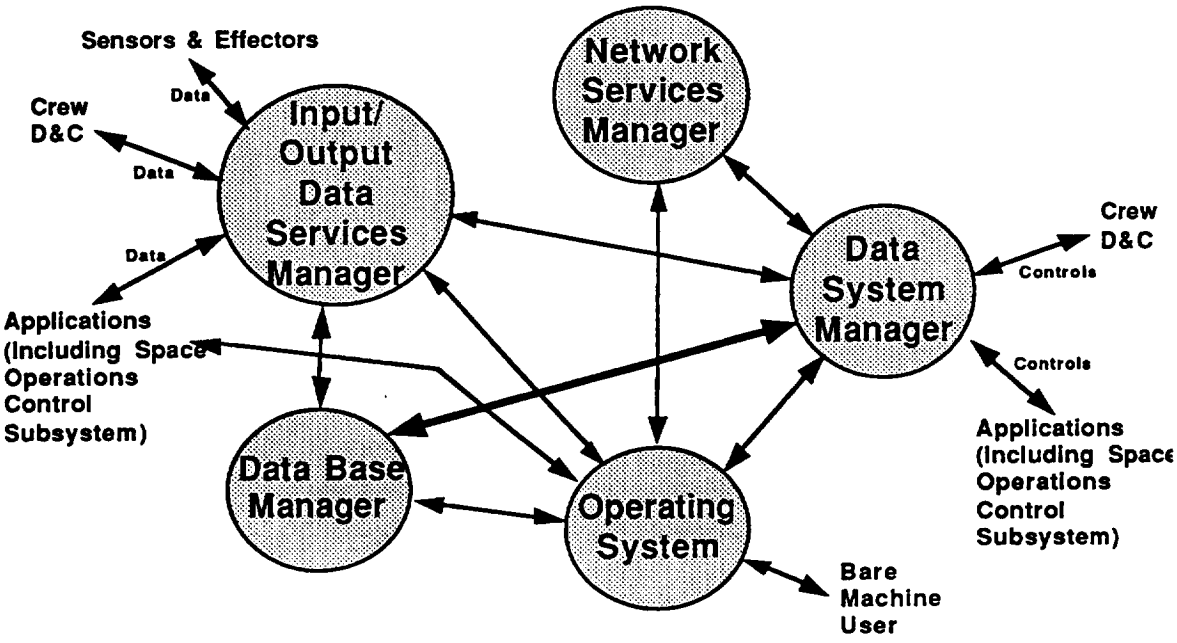


Figure 5-3. Data System Services Preferred Interface Elements

5.2.1.1 Input/Output Data Services Management

The IOSM shall [1] provide interface to the system users for data processing and data communication services. Services to be provided to the users shall [2] be derived directly from user requirements. The input/output data services management shall [3] include at least requirements for input/output services data acquisition, input/output services data distribution and reports generation.

5.2.1.2 Data System Management

The DSM shall [1] provide the housekeeping and control services for the SDSS. Command and control service requirements shall [2] be derived directly from user needs. Data system management shall [3] include at least requirements for configuration management, timing service control, initialization startup and reconfiguration, error processing, error recovery, fault treatment and reporting of health status to onboard health management. The DSM shall [4] execute under the operating system. There is a command and control interface to the crew and to the Space Operations Control Subsystem (SOCS). Command and control service requirements are derived directly from user needs.

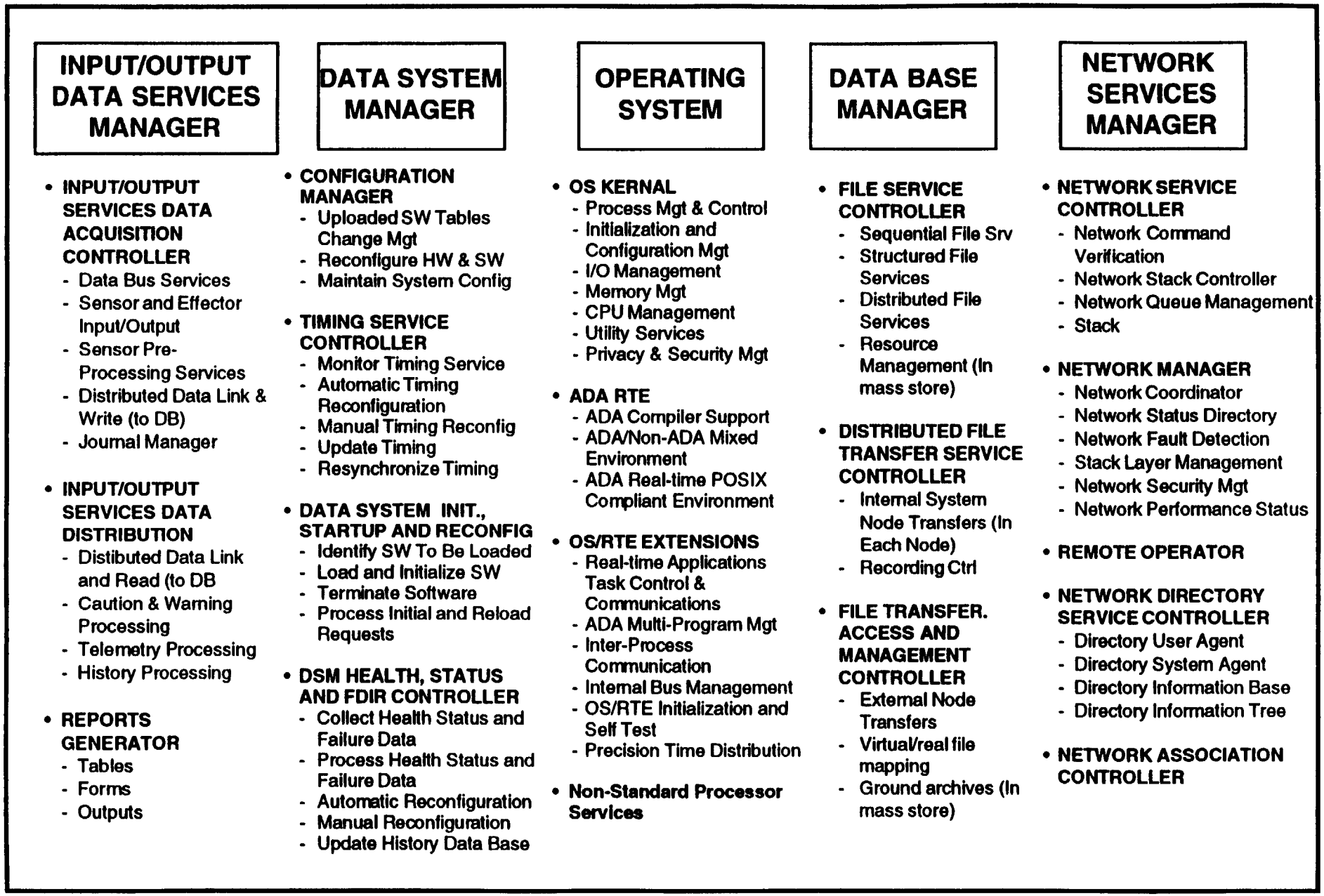


Figure 5-4. Interface Service Elements

5.2.1.3 Network Services Management

The NSM shall [1] provide for peer-to-peer communication between applications on distributed processing elements communicating over the SDSS system interconnect which require use of network communications between applications in distributed processing environments. The network services management shall [2] include at least requirements for network services, network management, remote operation, network directory service, and network association control.

5.2.1.4 Data Base Management

The DBM shall [1] provide services to the SDSS subsystems and application users for the management of structured data files, file transfers and file redundancy management. The data base management shall [2] include at least requirements for file services, distributed file transfer services, file transfer access and management, and node directory. All communication with and requests for services from the DBM are preferred to be through the IOSM.

5.2.1.5 Operating System Services

The OSS shall [1] provide the layer of SDSS that isolates other services as well as applications from the data processing physical resource element. OSS shall [2] provide management, allocation, and deallocation of the processor, memory, timing and I/O processing resources for applications and system services and physical resources that are independent of the mission. The OSS shall [3] offer at least open standard Operating System (OS) services such as an OS kernel and/or a run time environment (RTE) and OS/RTE extensions. Resource allocation and control that is mission dependent shall [4] be treated as an application.

A bare machine user may interface directly with the RTE.

5.3 PROCESSING STRUCTURAL MODEL REQUIREMENTS

An architecture compliant with the SGOAA Processing Structural Model requirements shall [1] consist of the capabilities for physical resources as shown below. The processing structure shall [2] be based on GAPs, SAPs, EPs and appropriate interconnects as required below.

5.3.1 PROCESSING RESOURCE STRUCTURE

The GAP architecture shall [1] be configured to provide physical resource components to interface to a system interconnect, to interface to local interconnects, to process applications, perform BIT and optional components for other purposes as required by the system. The SAP architecture shall [2] be configured to provide physical resource components for control, filtering, bus interface, BIT and other specialized purposes as required by the system. The EP architecture shall [3] be configured to provide physical resources for microcontrol, BIT, hardware handling and setup, and bus interface as required by the system.

a. The processing resource structure shall [4] provide communications from at least one of three levels of communications: (1) System Interconnects (e.g., Fiber Data Distribution Interfaces - FDDI), (2) Local Interconnects (e.g., MIL-STD-1553 bus and RS-449 links), and (3), Internal Interconnects (e.g., VME backplane). This is illustrated in Figure 5-5.

b. System interconnects may be implemented by high capacity peer to peer communication links providing communications between host platforms or sets of multiple nodes using techniques such as FDDI or by direct links between high data rate elements.

c. Local interconnects may be implemented by a combinations of buses and direct links for analog, discrete or serial communications within nodes, subsystem elements or components and within one host platform.

d. Internal interconnects may be implemented by a combination of low level communications, such as backplane buses to connect devices (e.g., circuit boards connected by VME) and internal component links.

The communications from sensors or effectors to EPs are only possible through direct links because the intention of the architecture is that embedded processors are those processors embedded in the sensor or effector hardware devices to minimize the communications latencies.

5.3.1.1 Generic Processing External Architecture

The Generic Processing External Architectures shall [1] be defined as shown in the example in Figure 5-5. The architecture system interconnect represents the inter-subsystem connectivity, and can be implemented by a combination of one or more communications paths using point-to-point, ring, bus or other architecture designs. Typically, system

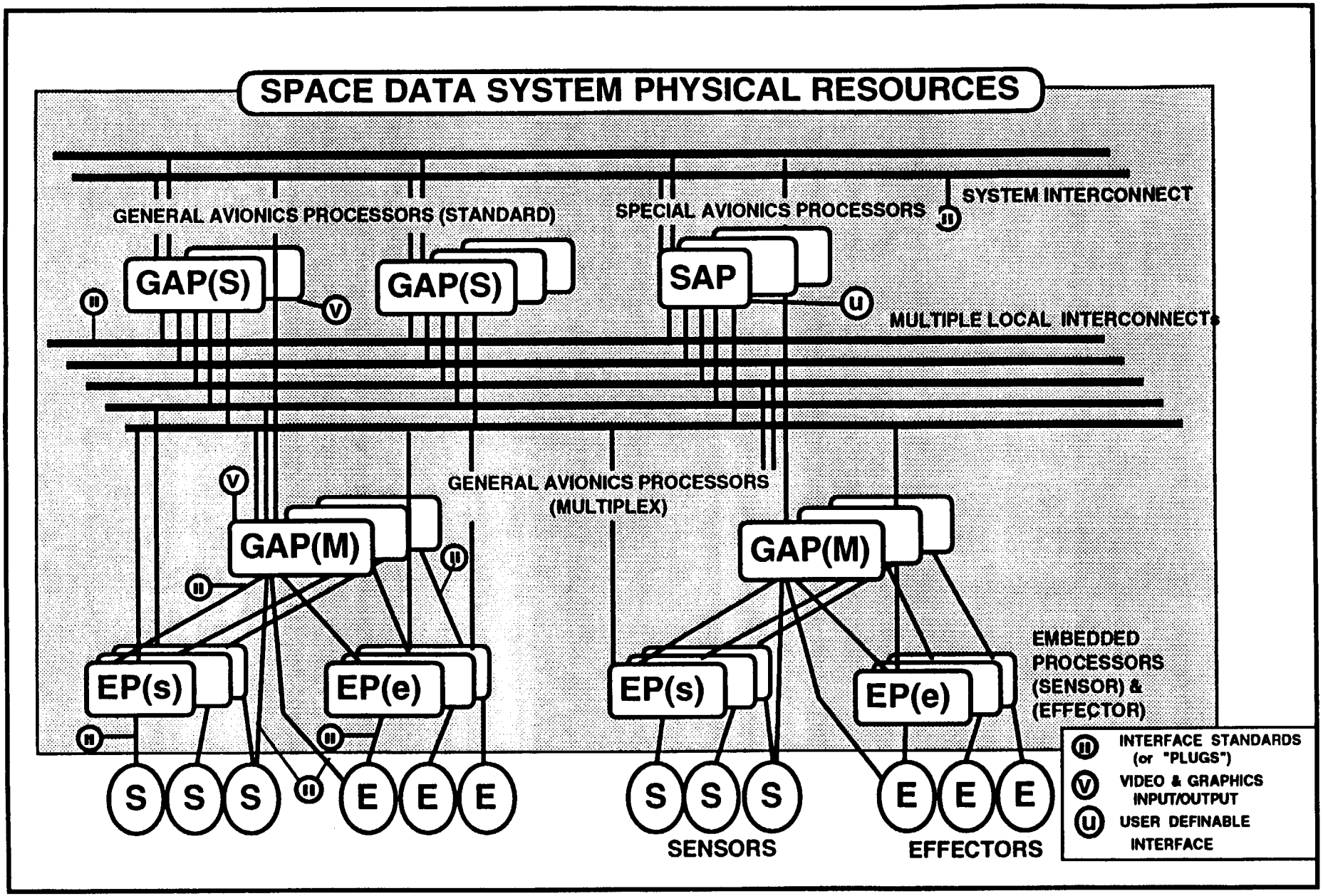


Figure 5-5. Generic Processing External Physical Resource Architecture and Interfaces

interconnects such as core networks are implemented by lower level standards such as FDDI or Ethernet. Local interconnects provide the intra-subsystem connectivity for high speed data communications between processors within one subsystem. Typically, the local interconnects are implemented by lower level standards such as MIL-STD 1553B for local command and data buses, RS-488 for timing controls, and direct links for analog and discrete signals. The interface plugs shown represent the unique physical resources interfaces which shall [2] be defined by standards.

5.3.1.2 GAP Architecture

GAPs represent general purpose data processors. A GAP, if required, shall [1] be one of two forms: one for standard general purpose use [GAP(S)] and one for multiplexing and demultiplexing signals [GAP(M)]. Typically, GAP devices are used where slow response times (such as on the order of seconds to tens of seconds) are required. An example of a compliant implementation of GAP(S) processors is the General Purpose Processing Element (GPPE) in the F-22 program. An example of a compliant implementation of the GAP(M) is the Multiplexer-DeMultiplexer processor in the Space Station program.

The requirements for general purpose processing elements in a vehicle shall [2] be defined as shown in the GAP architecture presented in Figure 5-6. The generic physical resource elements shown in the figure comprise the basic, generic physical resource modular elements in the SGOAA. The processor may be configured as a GAP(S) or GAP(M) depending on the set of functions required by a specific application.

5.3.1.2.1 GAP Function Set

The GAP function set in Figure 5-6 is a shopping list of modular functions which can be used to build the needed configuration. Each module shown provides a specific independently procurable service. Additional unique service functions may be added by defining additional modules. The actual implementation in physical resources is interface standard, technology and detailed design dependent. System performance requirements for physical resource modular elements shall [1] be a primary consideration in module selection to perform a specific function. System error processing and fault treatment requirements for physical resource modular element Built-In Test Equipment (BITE) shall [2] also be considered in physical resource modular element selection. Specific physical resource interfaces that shall [3] be defined by lower level standards are shown in Figure 5-6.

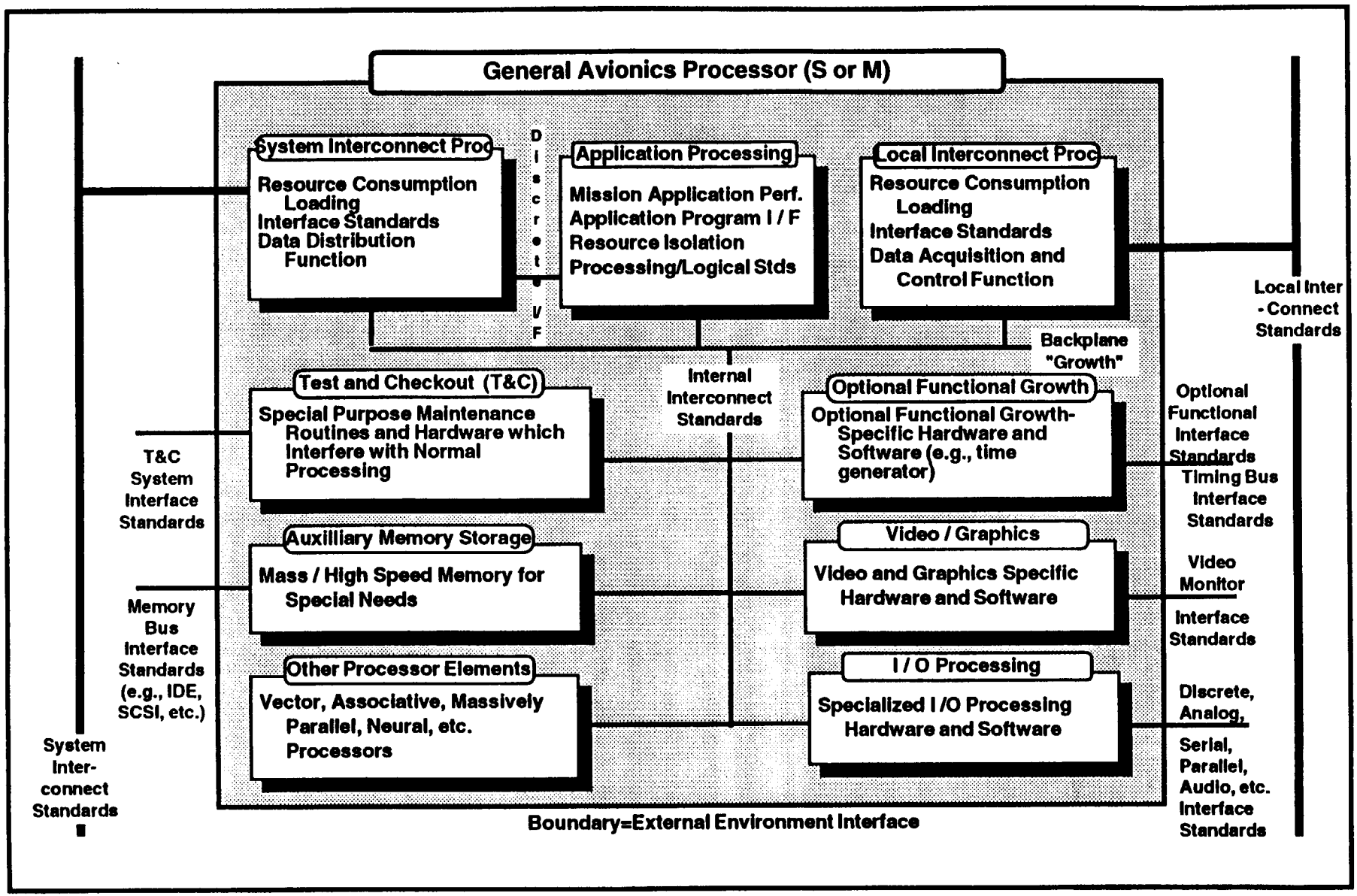


Figure 5-6. Generic Processing Internal Physical Resource Architecture

5.3.1.2.2 Internal Interconnect Interface Standards

Internal interconnect interface standards shall [1] be imposed to provide modularity with the capability for technology upgrades and multiple vendor sources of processing functions modules. Although only one internal interconnect bus is shown for the backplane in Figure 5-6, the actual bus implementation may consist of multiple buses depending upon the specific application. Possible buses include data, time, test, and local memory. Multiple standards exist for all of these bus types.

5.3.1.2.3 Lower Level Interface Standards

Lower level interface standards as illustrated in Figure 5-6 shall [1] be selected for system interconnect, test and checkout system, mass memory, timing bus, discrete data , analog data, serial data , parallel data, local interconnect, video/graphics, audio and optional functional growth interfaces. For example, to implement the functions of the basic GAP(S) would require implementation of the system interconnect processing, application processing and local communications (e.g., local interconnect and I/O) processing functions of the GAP Physical Resource Architecture shown in Figure 5-6. A internal interconnect bus standard such as Future Bus Plus (FB+), VME or Pi Bus would be imposed as the backplane data bus standard. The backplane bus standard used in a specific architecture implementation might consist of one or more specific buses; separate buses are permitted for uses such as test and maintenance.

5.3.1.3 SAP Architecture

SAPs if required, shall [1] provide the special purpose processing which is usually needed in high power embedded computers and may be implemented by devices such as vector or associative processors, massively parallel data processors, or arithmetic coprocessors. The SAP thus provides functions not included in the generic processor function set such as vector or parallel processing. Typically, SAP devices are used where response times (such as on the order of hundreds of milliseconds to a second) significantly faster than in a GAP are required. Examples include the associative and vector processors used in the F-22 program.

5.3.1.4 EP Architecture

Within each sensor or effector, this architecture allows, but does not require, the placement of processors embedded in the sensor or effector unit. EPs if required, shall [1] be one of two forms: one for effector processing [EP(e)] and one for sensor processing [EP(s)]. A processor may be configured as a EP(s) or EP(e) depending on the function required by a specific

application. EPs shall [2] provide the very high speed processing necessary to manipulate and convert analog data to digital data while performing some preprocessing on the data to reduce the data rate to a more acceptable level for linkage back to the GAP(M). Typically, EP devices are used where very fast response times (such as on the order of milliseconds or less) are required. Where the data rate with the sensor or effector is acceptable to the GAP(M) and no other pre-processing is required, direct interface to the GAP(M) may be used. Sensors and effectors interface to the EP devices either through local communication interfaces or through direct links.

5.3.1.5 Lower Level Interface Architecture

Lower level interface standards shall [1] be selected for implementing system interconnects, local interconnects, GAP to EP direct links, GAP to S direct links, GAP to E direct links, EP to S direct links, and EP to E direct links. User definable interfaces shall [2] be provided for the SAPs. Lower level video and graphics interface standards shall [3] be selected to define implementations for connecting the GAP devices to humans for development, operation and maintenance of the systems.

5.3.2 RESERVED

5.4 ARCHITECTURE INTERFACE MODEL REQUIREMENTS

An architecture compliant with the SGOAA Interface Model requirements shall [1] consist of nine classes of interfaces as shown in Figure 5-7 and defined in Table 5-1. These classes are the levels of interfaces from physical resource up to high level systems which are to be completely defined in an architecture developed in accordance with this standard. Definition of each interface class shall [2] be in accordance with the requirements contained in the following paragraphs.

For flight and safety critical functions, the exchange of information for error processing and control shall [3] be restricted to classes 1D, 1L, 2D, 2L, 3D, and 3X. For mission critical functions, the exchange of information for error processing and control shall [4] be restricted to classes 1D, 1L, 2D, 2L, 3D, 3X, and 3L. For any critical functions or service function with hard deadlines, an architecture prepared in compliance with this standard shall [5] not allow the exchange of information for error processing across Interface Classes 4D and 4L. Errors introduced by data transmission are removed at the lower interface class. The data itself, however, may still have errors if the source of the data was in error. Error processing on

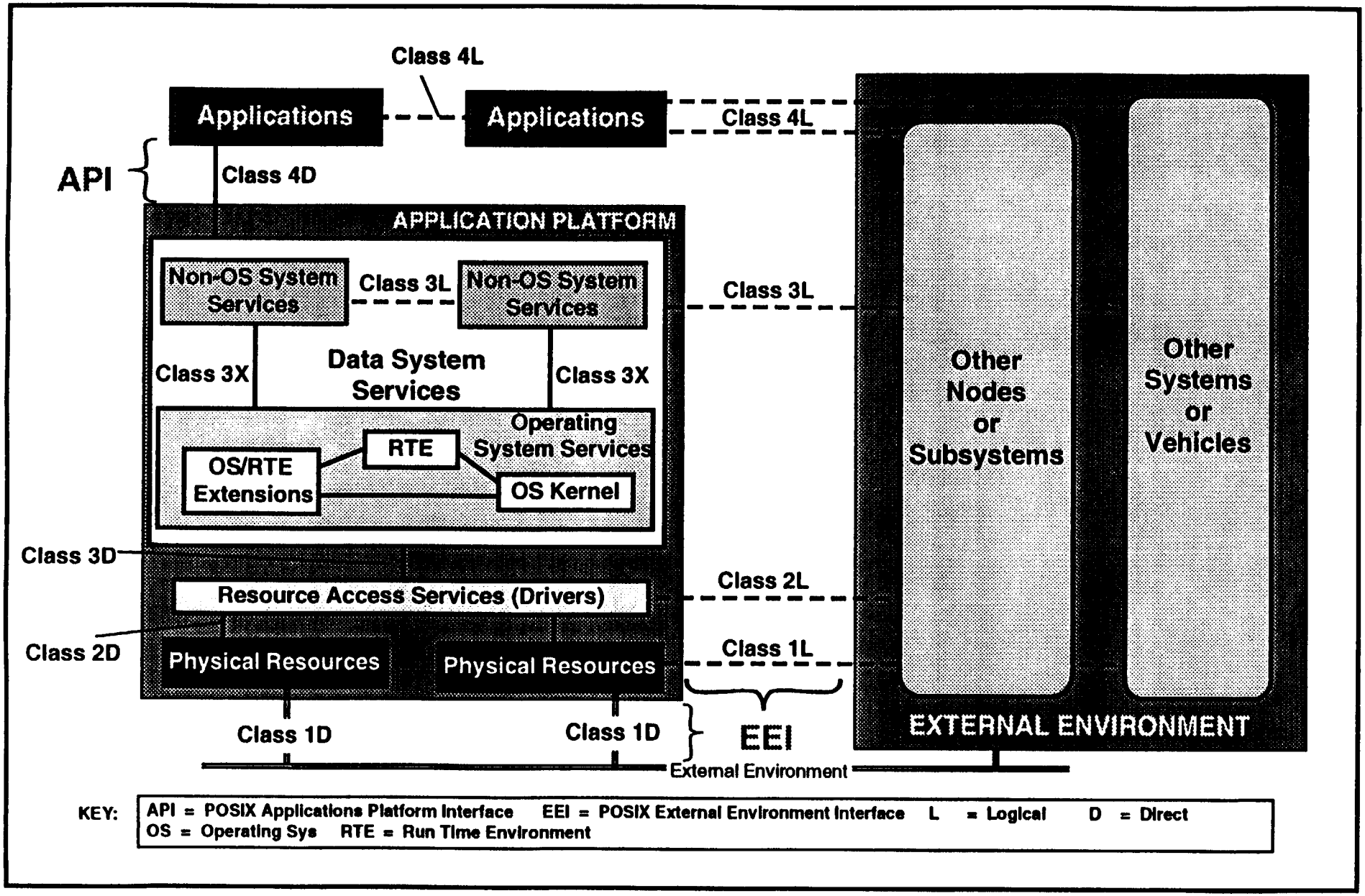


Figure 5-7. General Interface Model Standard Reference

Table 5-1. ARCHITECTURAL INTERFACE CLASSES

Class	Description
1D	**Physical Resources-to-Physical Resources Direct:** Class 1D physical resources to physical resources direct interfaces are the direct connections between different types of physical resources needed to enable buses and communications links to address processors or needed to enable processors to address memory registers.
1L	**Physical Resources-to-Physical Resources Logical Peer:** Class 1L physical resources to physical resources logical peer interfaces are the requirements for establishing a data interchange interface between physical resources that enable bus or communications link boards to address their peers in another node or system .
2D	**Resource Access Services-to-Physical Resources Direct:** Class 2D resource access services to physical resources direct interfaces are the direct connections between hardware registers and resource access services or other services performing that function, such as drivers needed to enable address registers to move data packets from hardware to services, and other drivers which can respond to the data packets.
2L	**Resource Access Services-to-Resource Access Services Logical Peer:** Class 2L resource access services to resource access services interfaces are the requirements for establishing a data interchange interface between resource access services in one node with those of the same or another node which enable low level peer data exchange such as between drivers for different hardware.
3D	**Data System Services-to-Resource Access Services Direct:** Class 3D data system services to resource access services direct interfaces are the direct connections between data system service code and resource access service code sets, which enable data system services to receive and interpret data packets, and pass them on to other service code which will process them locally.
3X	**Operating System Services-to-Non-OSS Data System Services Direct:** Class 3X operating system services to other non-operating system data system services direct interfaces are the direct connections between operating system service code and other non-OSS service code sets, which enable operating system services to conduct privileged interactions with local non-OSS service code.
3L	**Data System Services-to-Data System Services Logical Peer:** Class 3L data system services to other data system services logical interfaces are the requirements for establishing data interchange interface local services to determine the identity of the intended service in other local or remote locations which need the register data being stored and to pass the data appropriately, without requiring knowledge of service physical location. Enables the handling of logical data transfers from source to user service.
4D	**Data System Services-to-Applications Direct:** Class 4D data system services to applications direct interfaces are the direct connections which enable data system service code to access and process data with applications code.
4L	**Applications-to-Applications Logical:** Class 4L applications to applications logical interfaces are the requirements for establishing a data interchange interface enabling an application originating data to pass it to an application which needs to use the data, or enable an application needing data to determine the source from which the data must be obtained. These are logical data transfers from source to user. This interface provides the requirements for establishing a data interchange interface that allow applications in different systems or in the same system to communicate, thus enabling applications to interact across or within system boundaries to accomplish a mutual purpose. These interfaces may be applicable to applications executing in the same processor, in different processors in the same node or in different systems.

these data errors may take place in the application, but it must be understood that such error processing reduces the portability of this application.

5.4.1 CLASS 1D - PHYSICAL RESOURCES-TO-PHYSICAL RESOURCES DIRECT INTERFACE REQUIREMENTS

The Class 1 Physical Resources-to-Physical Resources Direct Interface shall [1] be defined in accordance with the processing structural models discussed in paragraph 5.3.

Physical Resources to Physical Resources direct interfaces shall [2] be defined as shown in Figure 5-8. These interfaces consist of the nuts and bolts, chips and wires of the system architecture model described in paragraph 5.1. With regard to the model, this interface shall [3] consist of all the Physical Resources to Physical Resources interfaces within each processing element, as well as the Physical Resources interfaces to the external environment by way of the system interconnect, local interconnects, internal interconnects or direct interfaces. This architecture shall [4] provide for three classes of processors: the EPs, SAPs and GAPs for which standardized interfaces shall [5] be required to be selected from a set of acceptable lower level interface standards.

All types of error processing on data transmissions through the class 1D interface shall [6] be allowed.

5.4.2 CLASS 1L - PHYSICAL RESOURCES-TO-PHYSICAL RESOURCES LOGICAL PEER INTERFACE REQUIREMENTS

Physical resources to physical resources logical peer interfaces are shown in Figure 5-9. These interfaces shall [1] consist of the requirements for establishing a data interchange interface/protocol between physical resources enabling communication link physical resources to address their peers in another node or system..

All types of error processing on data transmissions through the Class 1L interface shall [2] be allowed.

5.4.3 CLASS 2D - RESOURCES ACCESS SERVICES-TO-PHYSICAL RESOURCES DIRECT INTERFACE REQUIREMENTS

Resource Access Services to Physical Resources Direct interfaces are shown in Figure 5-10. These interfaces shall [1] consist of the interfaces from the resource access services or other

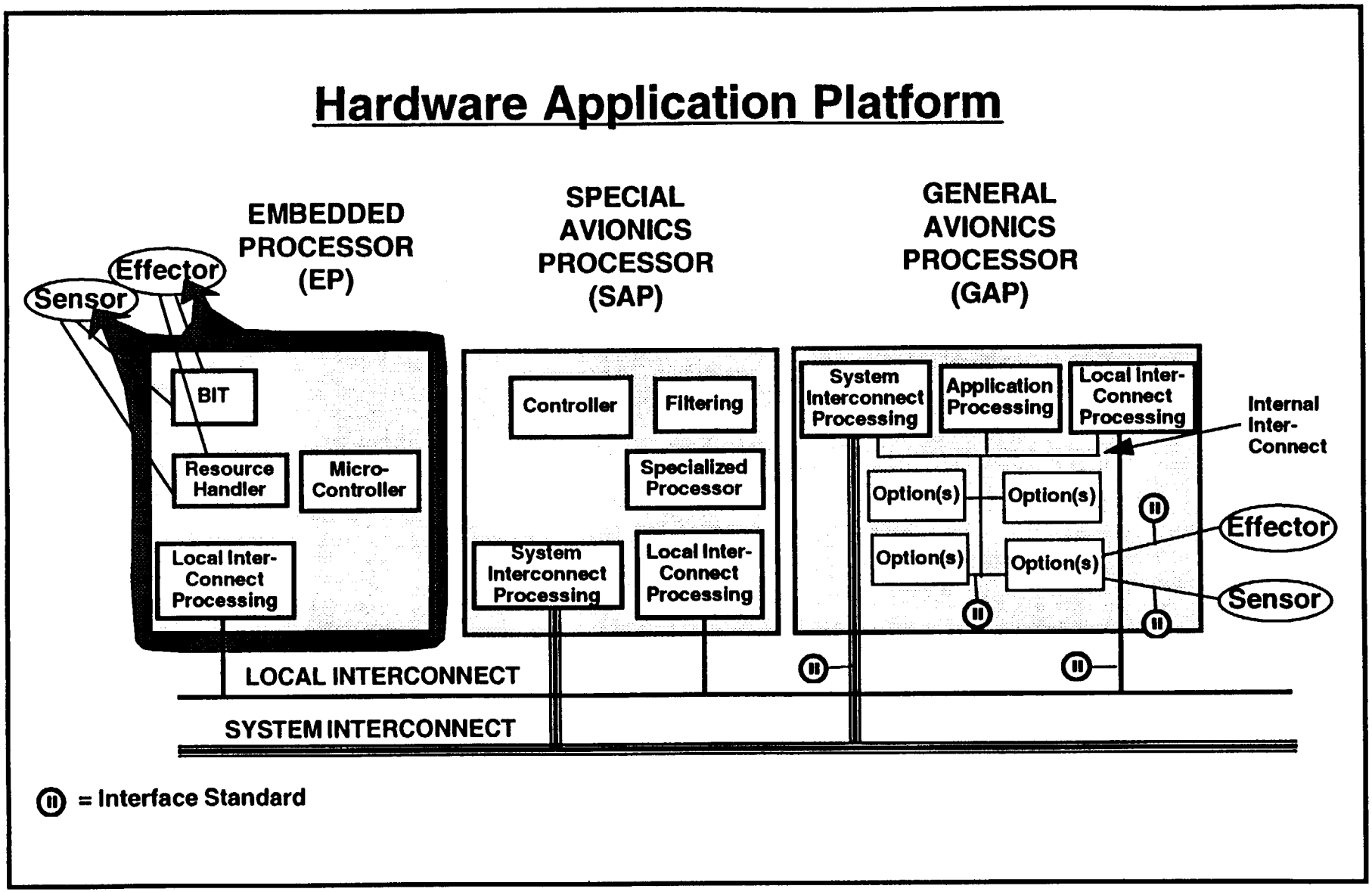


Figure 5-8. Class 1D Physical Resources-to-Physical Resources Direct Interfaces

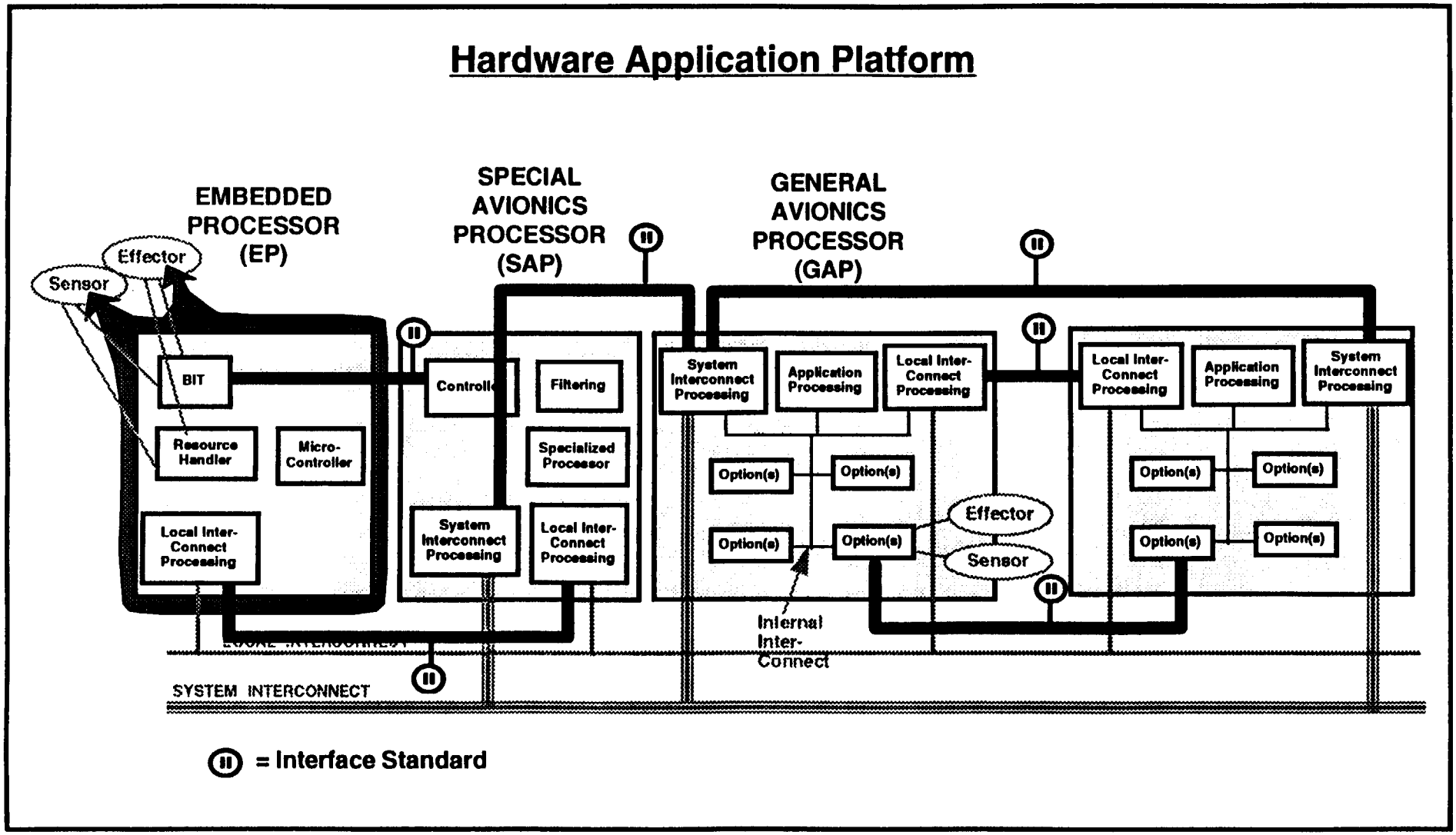


Figure 5-9. Class 1L Physical Resources-to-Physical Resources Logical Peer Interfaces

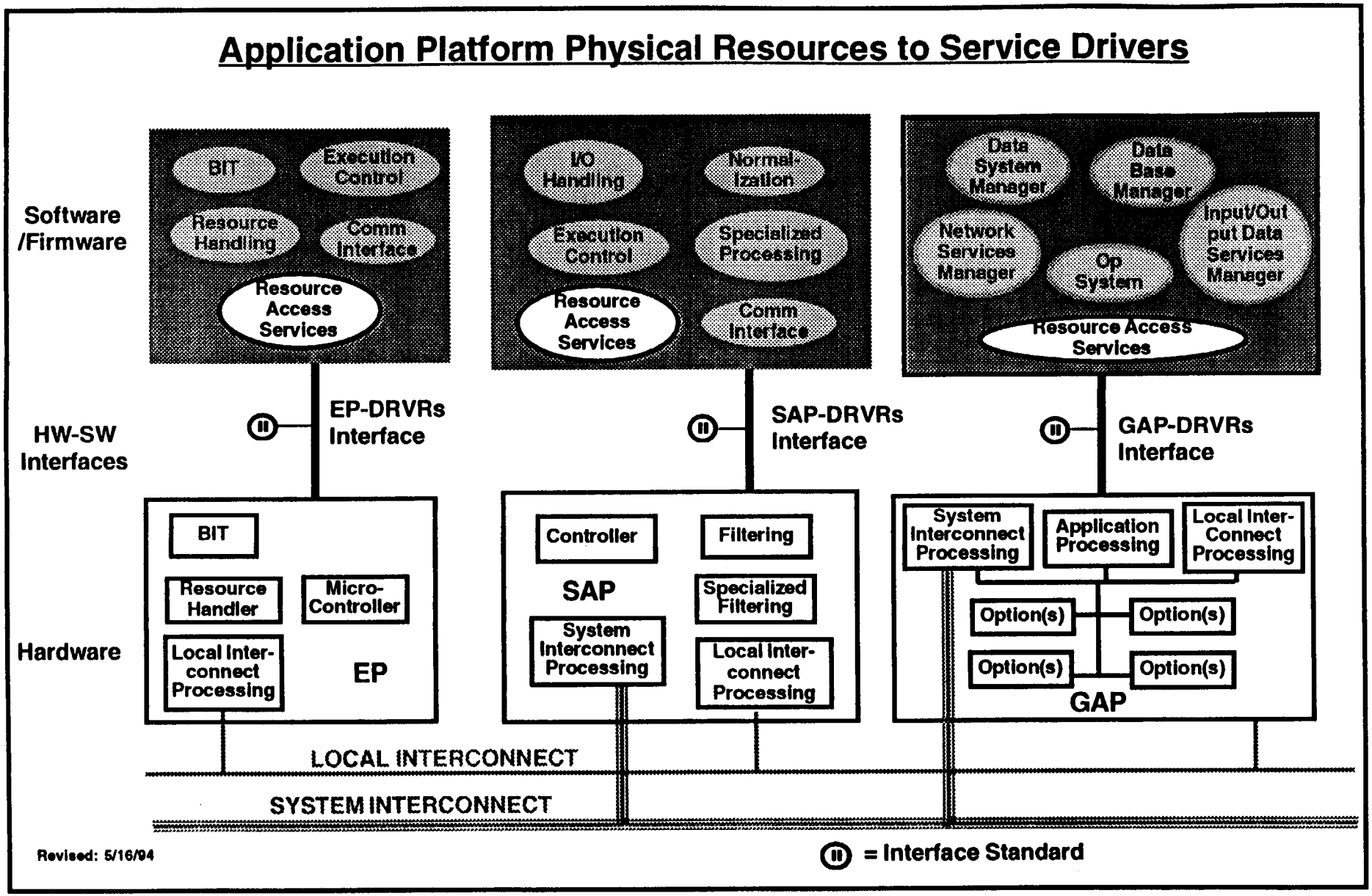


Figure 5-10. Class 2D Resource Access Services-to-Physical Resources Direct Interfaces

services performing the same function (such as drivers in the OS, DSM, etc.) to the hardware instruction set architecture (ISA) and register usage. With regard to the model, these interfaces are internal to each processing element. This class shall [2] define the interfaces for low level service drivers that interact with the physical resources for each of the processor types (EPs, SAPs, and GAPs). All the drivers for all processor types shall [3] be contained in a SDSS sub-architecture.

An example of the Physical Resources to Resource Access Services interfaces needed for the physical resources to be accessed by low level services such as drivers is shown in Figure 5-11. The interfaces are shown in black and labeled, and everything else has been grayed out to highlight items of interest.

All types of error processing on data transmissions for flight or safety critical functions through the class 2D interface shall [4] be allowed except those employing retransmission. For those mission critical or service functions that have no hard deadline requirements, error processing employing retransmission shall [5] be allowed.

5.4.4 CLASS 2L - RESOURCE ACCESS SERVICES-TO-RESOURCE ACCESS SERVICES LOGICAL PEER INTERFACE REQUIREMENTS

Resource Access Services -to-Resource Access Services interfaces shall [1] be defined as shown in Figure 5-12. This interface shall [2] be a peer to peer information/data exchange and coordination interface between Resource Access Services modules within a processing element or between separate processing elements. The circled arrows from Resource Access Services to itself represents internal peer to peer interfaces inside the Resource Access Services. Figure 5-13 is an example of Resource Access Services (low level service drivers)-to-Resource Access Services interfaces. This class shall [3] define the interfaces that enable information/data exchange and coordination between the low level service drivers.

All types of error processing on data transmissions through the class 2L interface shall [4] be allowed except those employing retransmission. For those mission critical or service functions that have no hard deadline requirements, error processing employing retransmission shall [5] be allowed.

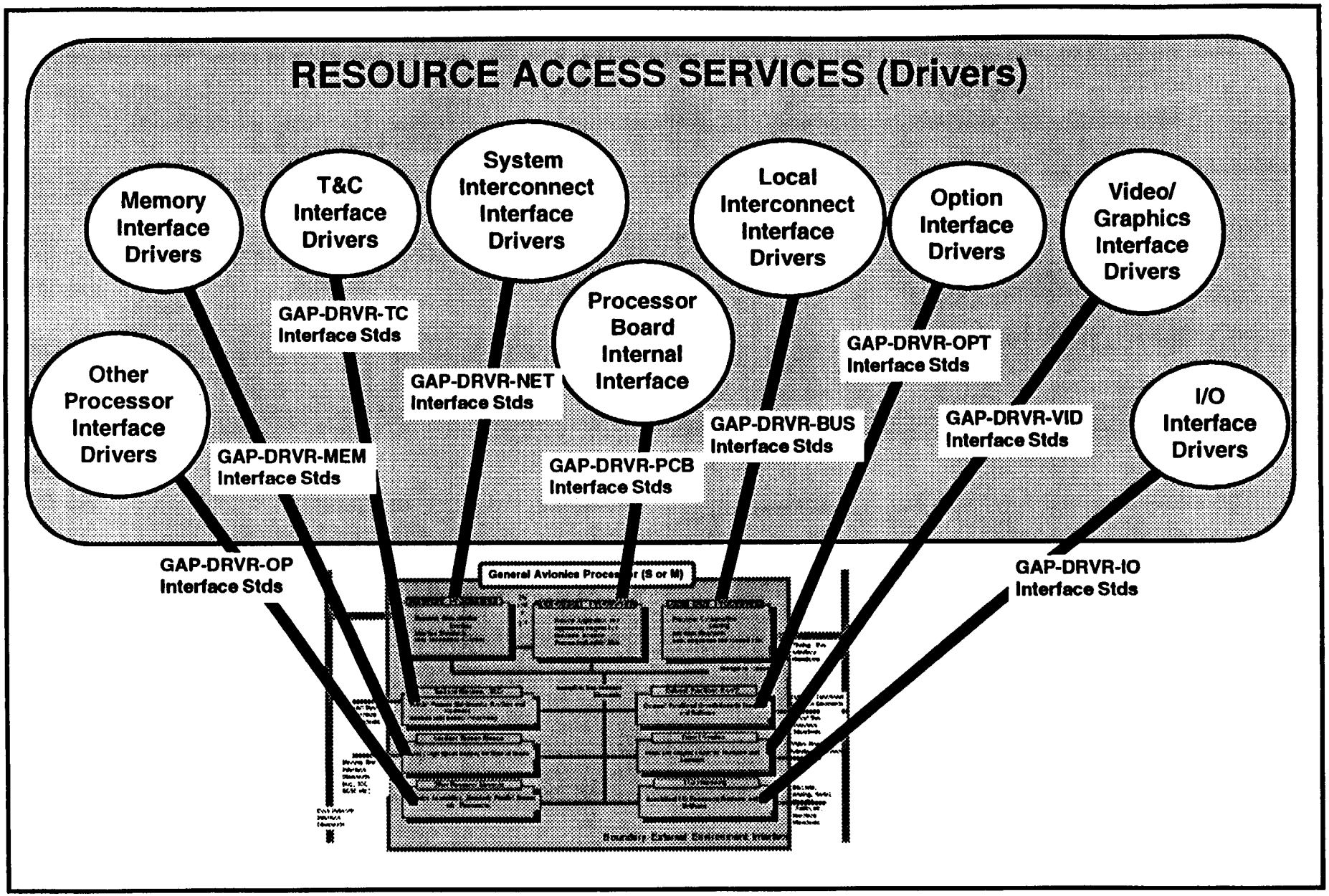


Figure 5-11. GAP to Resource Access Services (Class 2D Example)

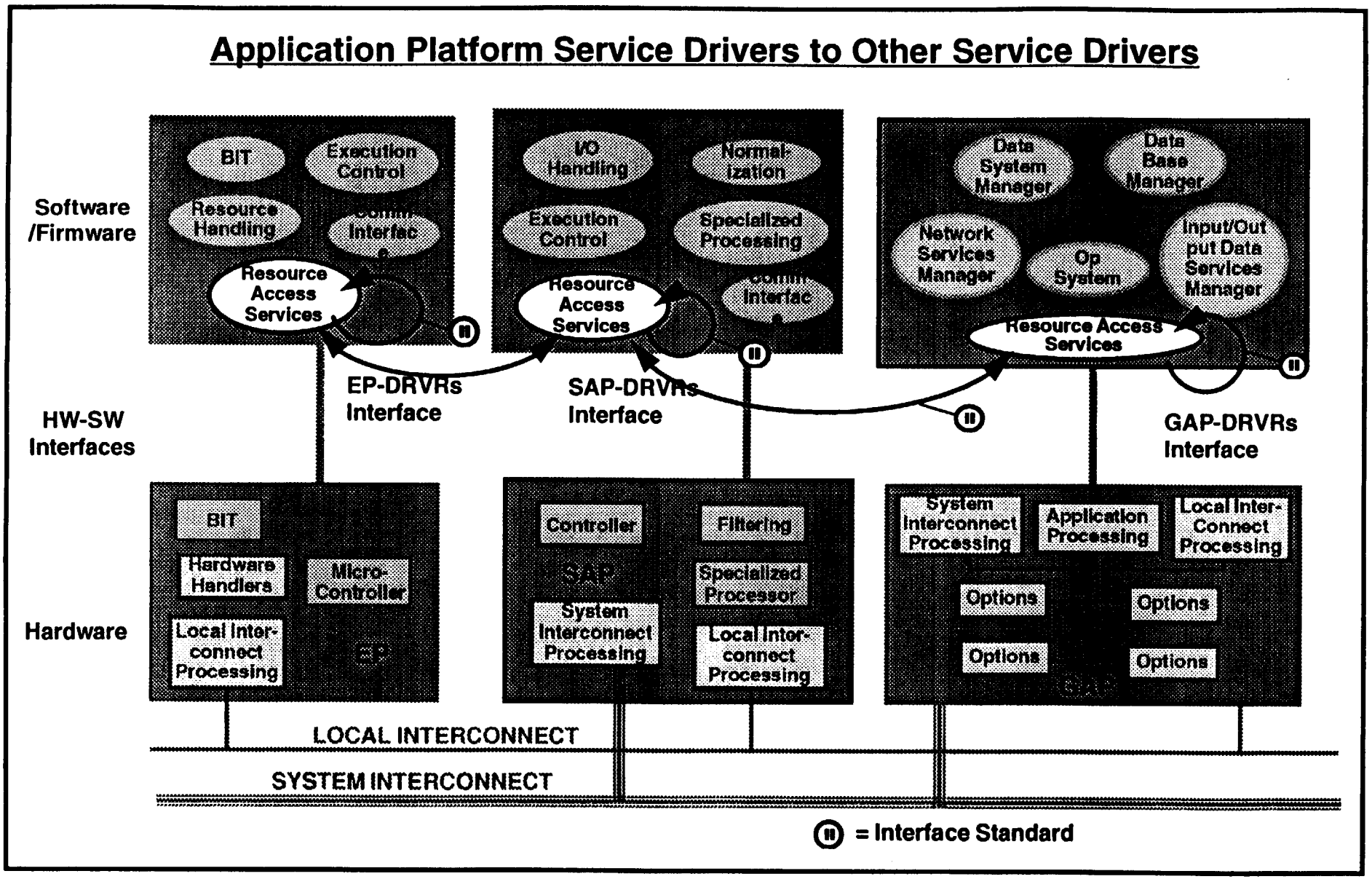


Figure 5-12. Class 2L Resource Access Services-to-Resource Access Services Logical Peer Interfaces

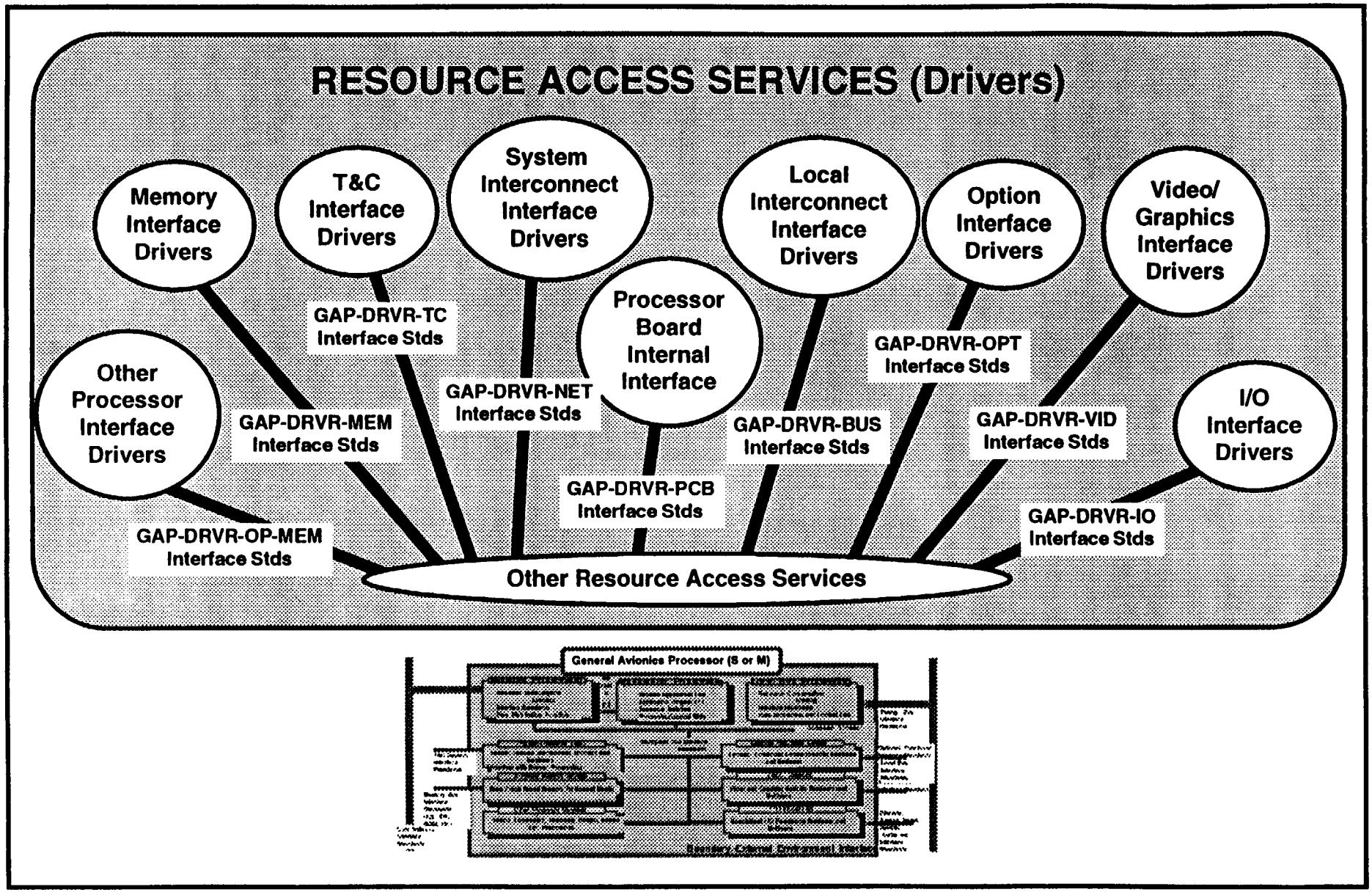


Figure 5-13. Class 2L Resource Access Services to Resource Access Services Example

5.4.5 CLASS 3D - DATA SYSTEM SERVICES-TO-RESOURCE ACCESS SERVICES DIRECT INTERFACE REQUIREMENTS

DSS to Resource Access Services Direct interfaces shall [1] be defined as shown in Figure 5-14. These interfaces shall [2] consist of the interfaces from DSS to the Resource Access Services or other services performing the same function (such as drivers in the OS, data system manager, etc.) to the hardware ISA and register usage. With regard to the model, these interfaces are internal to each processing element. The physical resource elements are grayed out to show that these elements are a repeat of Figure 5-10; the black elements represent the new capabilities and interfaces added by this interface class. This class shall [3] define the interfaces for low level service drivers that interact with the physical resources for each of the processor types (EPs, SAPs, and GAPs). All the drivers for all processor types shall [4] be contained in a sub-architecture.

All types of error processing on data transmissions through the class 3D interface shall [5] be allowed.

5.4.6 CLASS 3X - OPERATING SYSTEM SERVICES-TO-NON OPERATING SYSTEM SERVICES DATA SYSTEM SERVICES DIRECT INTERFACE REQUIREMENTS

The OSS-to-Non-OSS DSS interface shall [1] be defined as shown in Figure 5-15 and shall [2] consist of the interfaces between the OSS and the four other Non-OSS services that comprise the DSS. The interfaces are shown in black and labeled. Their grouping into class 3X facilitates design of OSS's and privileged interfaces needed to insure effective OSS performance. Class 2D and 3D provided the service drivers to isolate the physical resources, Class 3X provides the remainder of the direct operating system interfaces to local system services needed to operate the computer system.

The DSS sub-architecture consists of at least elements from the set of the DSM, DBM, IOSM, OSS, and NSM. Class 3X shall [1] provide the direct interfaces between the OSS to other local DSS applications for effective local interprocess communications and support. These interfaces are direct interfaces because they enable OSS code to interact with service code in other local entities. Although the OSS are a subset of the DSS, they shall [2] also provide direct low level OSS access not provided by the higher level DSS interface for those users requiring this type of interface. Class 3X interfaces shall [3] meet derived requirements based on the need of an application to support users.

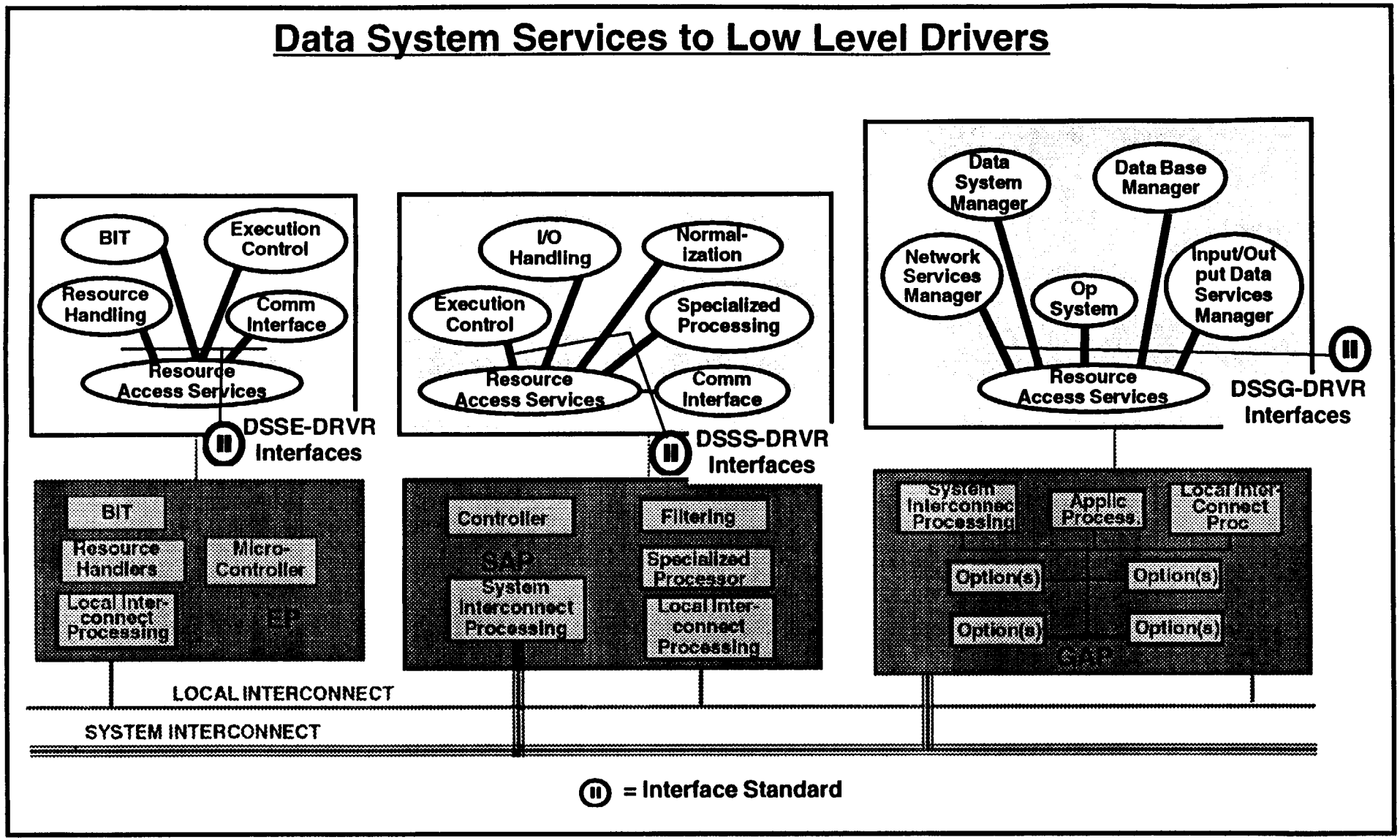


Figure 5-14. Class 3D Data System Services-to-Resource Access Services Direct Interfaces

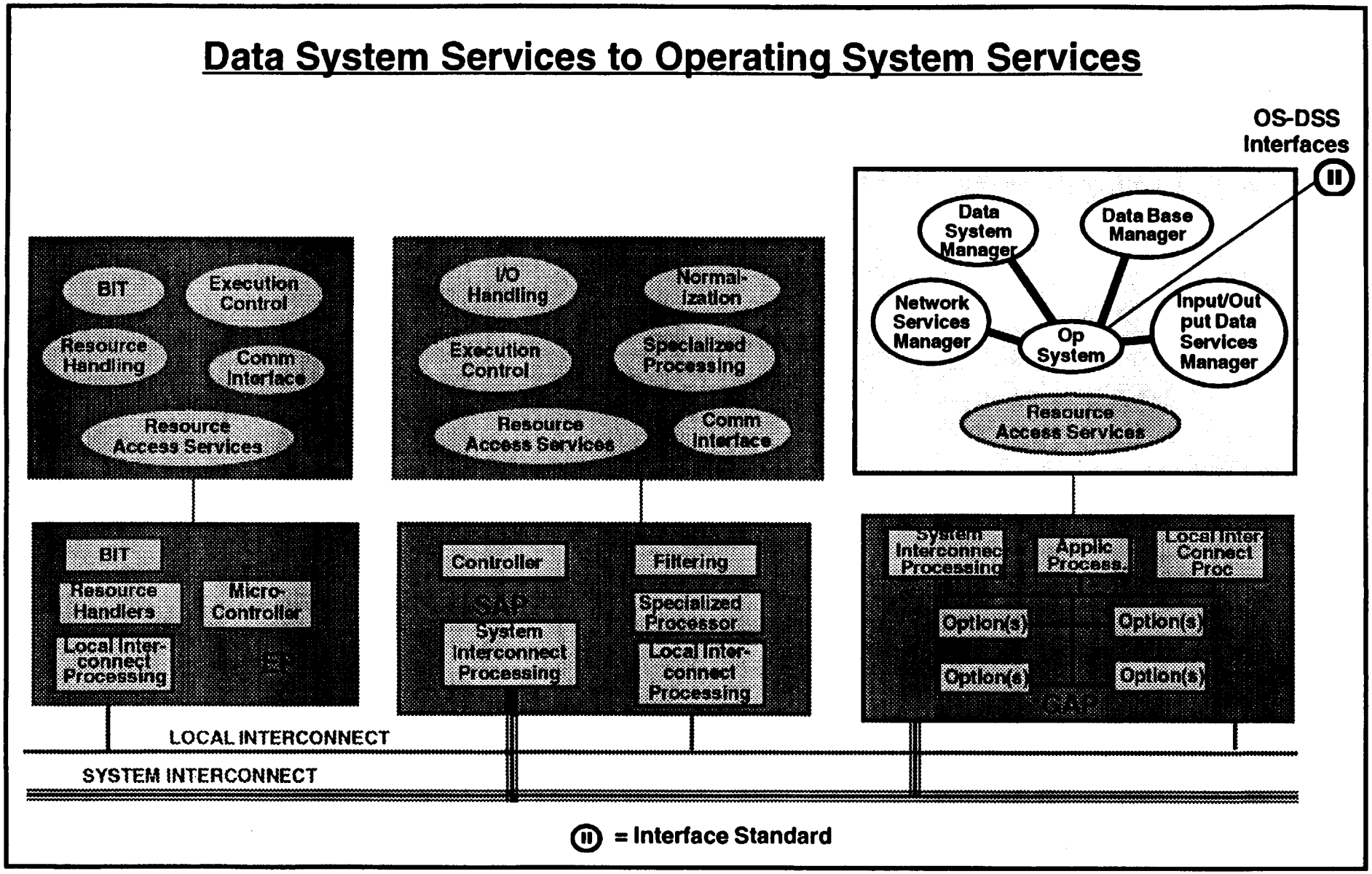


Figure 5-15. Class 3X Data System Services Operating System Services Privileged Interfaces

For class 3X interfaces of flight, safety or mission critical functions with hard deadlines to either the DSM or to Device Drivers, all types of error processing on data transmissions through the class 3X interface shall [4] be allowed except those employing retransmission. For those mission critical or service functions that have no hard deadline requirements, error processing employing retransmission shall [5] be allowed.

5.4.7 CLASS 3L - DATA SYSTEM SERVICES -TO-DATA SYSTEM SERVICES LOGICAL PEER INTERFACE REQUIREMENTS

DSS to DSS interfaces shall [1] be defined as shown in Figure 5-16. This is the peer to peer interface of DSS in one processing element (GAP, SAP or EP) interfacing with the system services in the same processing element or remotely to an external processing element to coordinate operations in a distributed environment. Since Classes 1D to 3X isolated the physical resources and system services in each processor, Class 3L shall [2] provide the interface capability for services in one processor to interact with services in the same or another processor. Class 3L interfaces shall [3] meet derived requirements based on the need of an application to support users in a multi-processing environment.

An example of the DSS interfaces needed to support local operations and logical access to other GAP DSS is shown in Figure 5-17. The black-line interfaces are the primary interfaces between the local services. Local services and remote services shall [4] have a common logical architecture. For distributed processing systems, a circular interface between each service entity and itself shall [5] be defined as shown in Figure 5-17, since each service must be able to communicate with remote versions of itself in other nodes. Remote interfaces to the special avionics processor and the embedded processor services shall [6] also be defined and specified.

For flight critical functions, error processing shall [7] not be allowed on data transmissions across the class 3L interface. For safety critical, mission critical or service functions, error processing not employing retransmission shall [8] be allowed.

5.4.8 CLASS 4D - DATA SYSTEM SERVICES-TO-APPLICATIONS DIRECT INTERFACE REQUIREMENTS

DSS to applications interfaces shall [1] be defined as shown in Figure 5-18. This is the direct interface within a processing element between the applications and the DSS (language bindings/specification) to allow provision of needed services. Since Classes 1 to 3 isolated

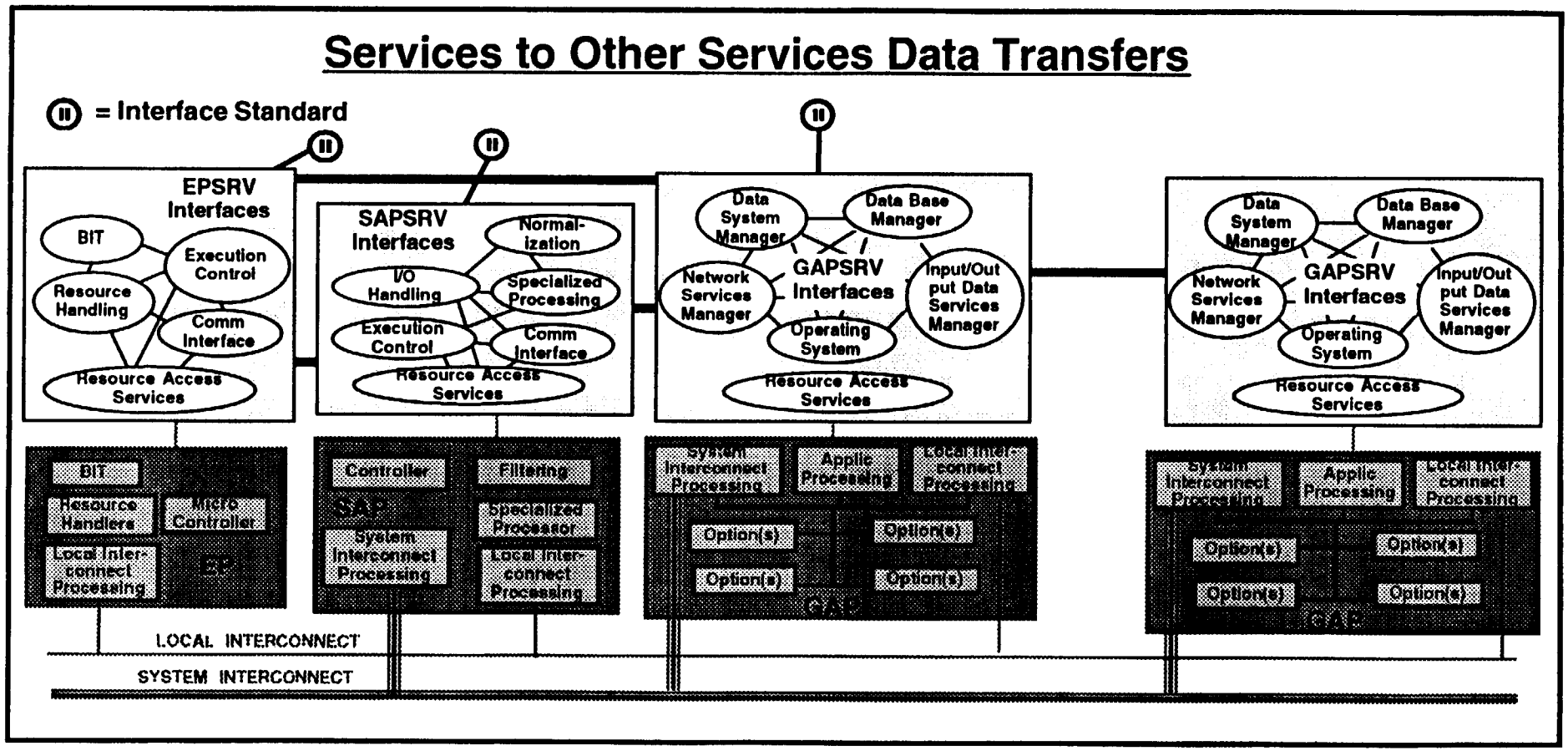


Figure 5-16. Class 3L Data System Services-to-Data System Services Logical Peer Interfaces

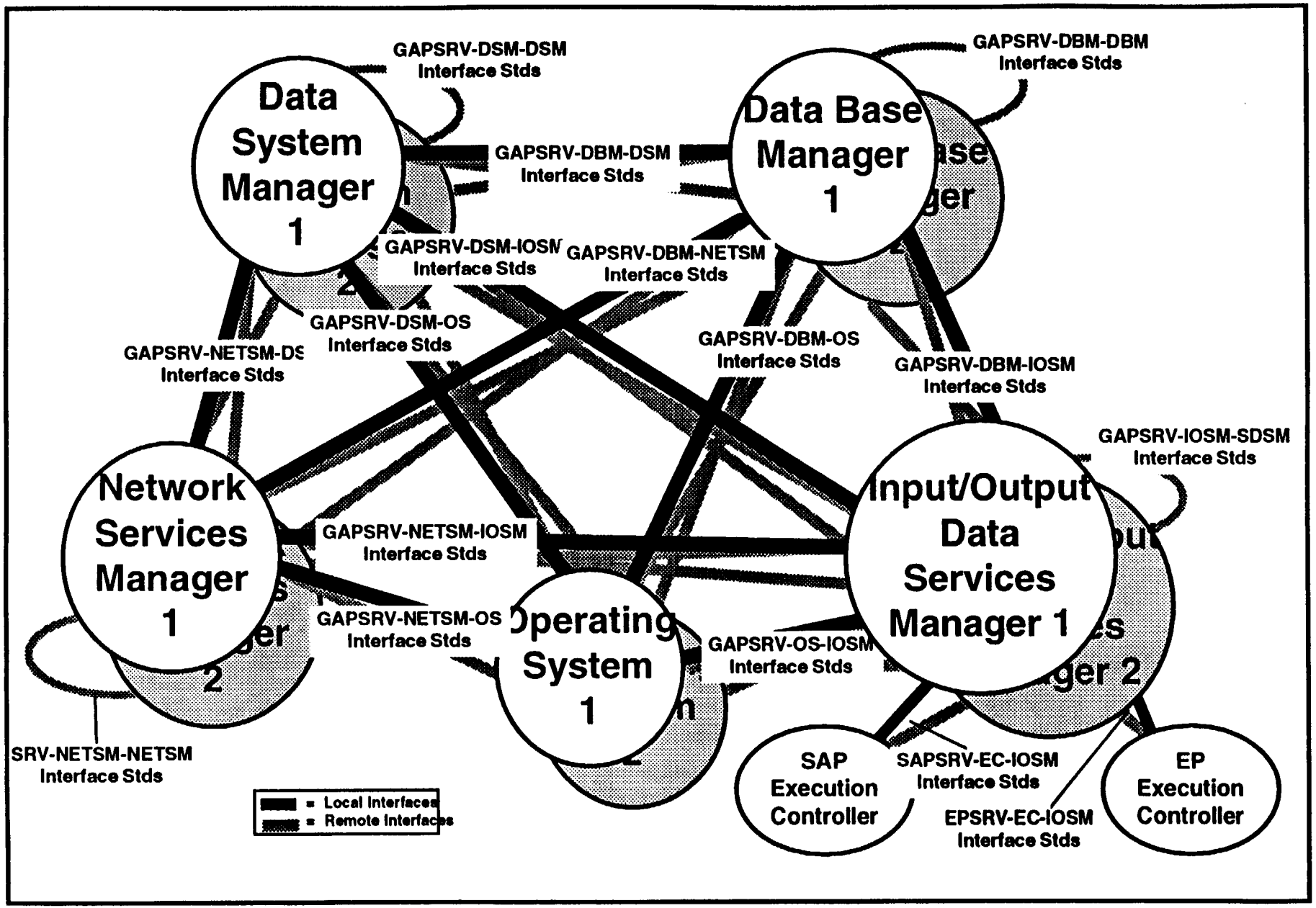


Figure 5-17. DSS Services to Other or Remote Services (Class 3L Example)

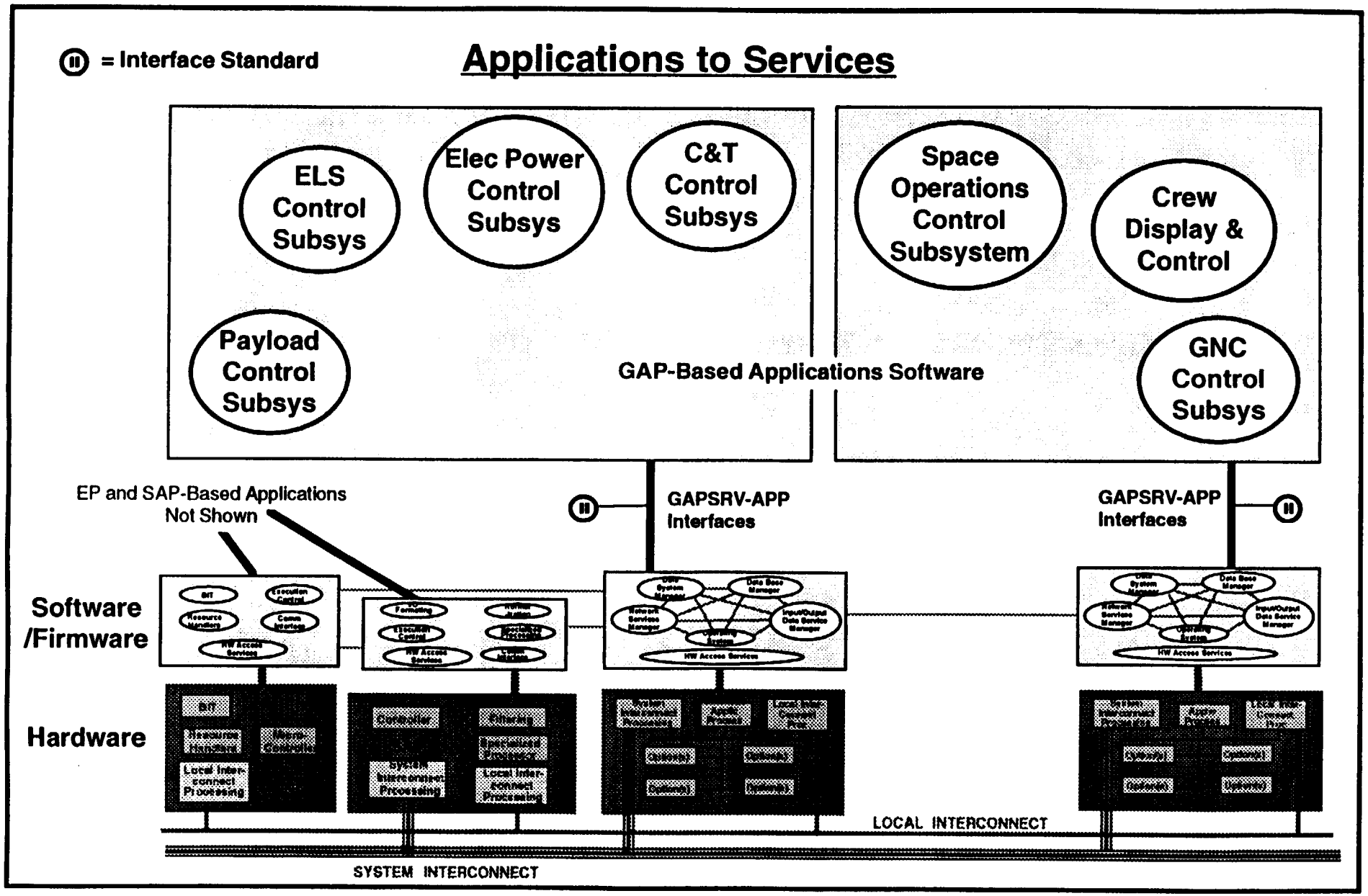


Figure 5-18. Class 4D Data System Services-to-Applications Direct Interfaces

the physical resources and system services in all the processors, Class 4D shall [2] provide the interface capability for services in any processor to interact with an application executing in the processor. Class 4D interfaces shall [3] meet derived requirements based on the need of an application to support users in a multi-processing environment.

An example of the DSS to applications interfaces is shown in Figure 5-19. The applicable interfaces are shown in black and labeled, and everything else has been grayed out to highlight items of interest.

This interface shall [4] be a standard access interface to the any of the DSS, which is a function of the service and independent of any one application. The preferred interfaces are to the DSM and to the IOSM which would provide access to the other three DSS. The IOSM shall [5] be capable of providing access to other services as well as directly to the application or sensor providing the source of data. The DSM shall [6] be capable of providing control interfaces to other control subsystems.

Error processing on data transmissions across the class 4D interface shall [7] not be allowed. Transfer of error processing results over the class 4D interface to Onboard Health Management processing shall [8] be allowed.

5.4.9 CLASS 4L - APPLICATIONS-TO-APPLICATIONS LOGICAL INTERFACE REQUIREMENTS

Applications to applications interfaces within a single system shall [1] be defined as shown in Figure 5-20. This shall [2] be a peer to peer information exchange and coordination interface between applications modules. Applications shall [3] not communicate directly. All applications to applications communications shall [4] be implemented by use of system services through at least a Class 4D standard interface to system services. This interface may be between applications within a processing element or between applications in separate processing elements. The grayed out parts of in Figure 5-20 represent the material covered in Classes 1D to 4D, the black parts of the figure are the new interface definitions added in Class 4L. Since Classes 1D to 4D isolated the physical resources, system services and applications in any processor, Class 4L shall [5] provide the interface capability for an application in any processor to interact with another application executing in any processor. Class 4L interfaces shall [6] meet user and derived requirements based on the need of multiple applications to support users in a multi-processing environment.

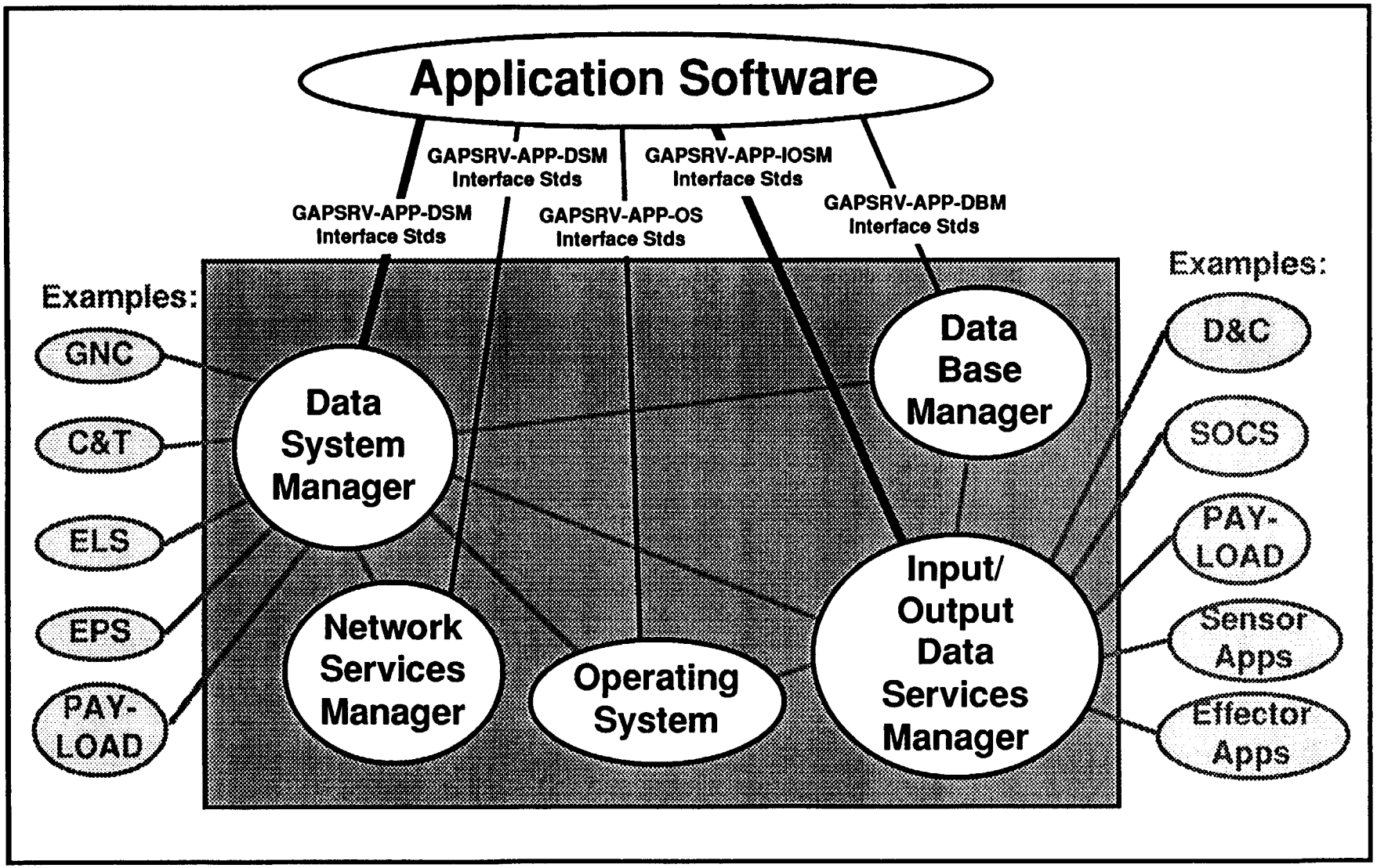


Figure 5-19. Services to Applications Interfaces (Class 4D Example

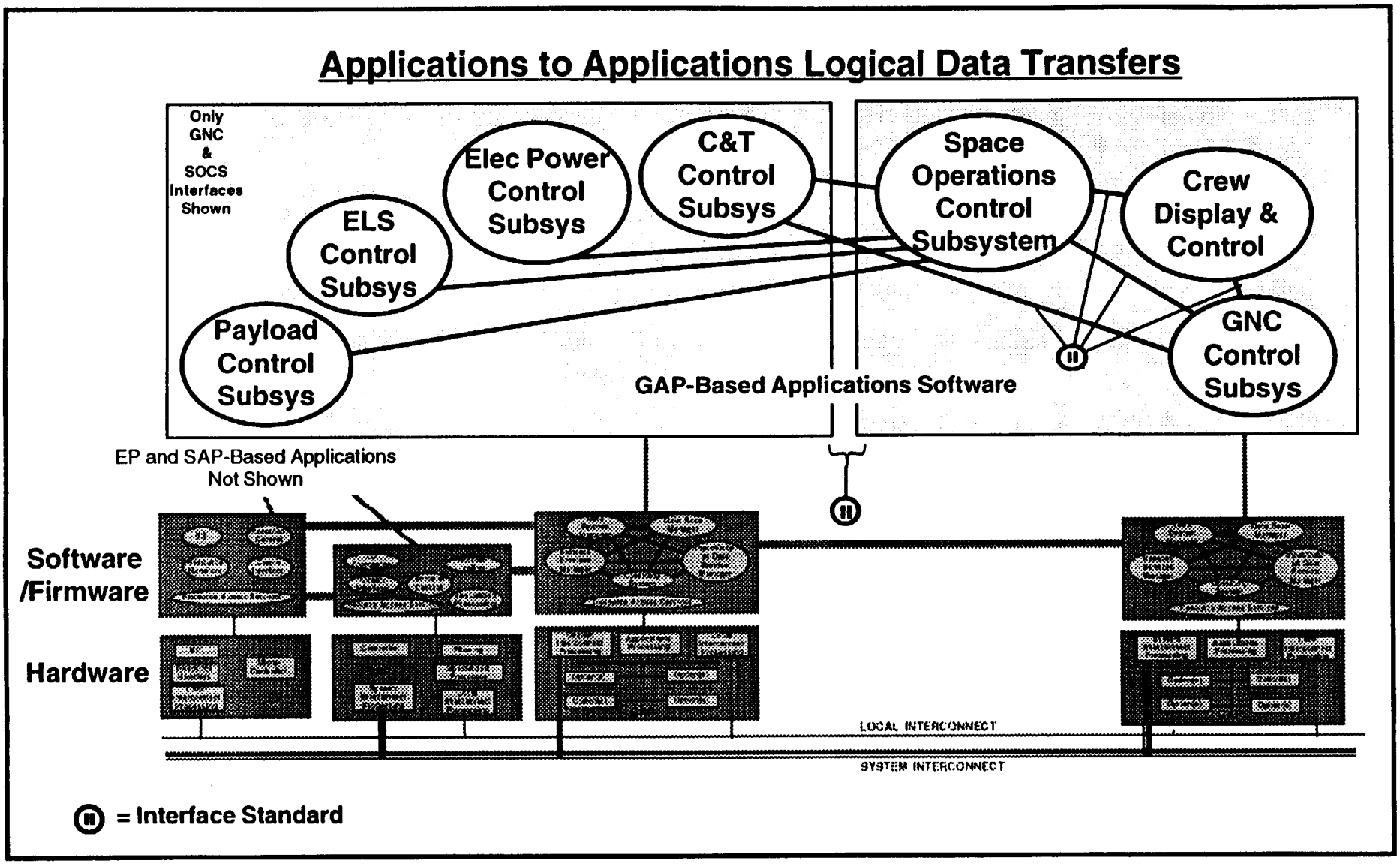


Figure 5-20. Class 4L Applications-to-Applications Logical Peer Interfaces

Applications to applications logical interfaces may also include interfaces between applications in two different systems or vehicles. System A applications to system B applications interfaces shall [7] be defined as shown in Figure 5-21. The grayed out parts of the figure represent the material covered in Classes 1D to 4L (within one system), the black parts of the figure are the unique interfaces that are provided by Class 4L for inter-system interfacing. Since Classes 1D to 4D isolated the physical resources, system services and applications in any system, Class 4L shall [8] provide the interface capability for an application in one system to interact with an application executing in another system. Class 4L interfaces shall [9] meet user and derived requirements based on the need of multiple applications to support users in a multi-system environment. Class 4L interfaces shall [10] be defined to meet the overall mission and operational control requirements across multiple facilities and vehicles.

Error processing on data transmissions across the class 4L interface shall [11] not be allowed. Transfer of data between Onboard Health Management entities shall [12] be allowed.

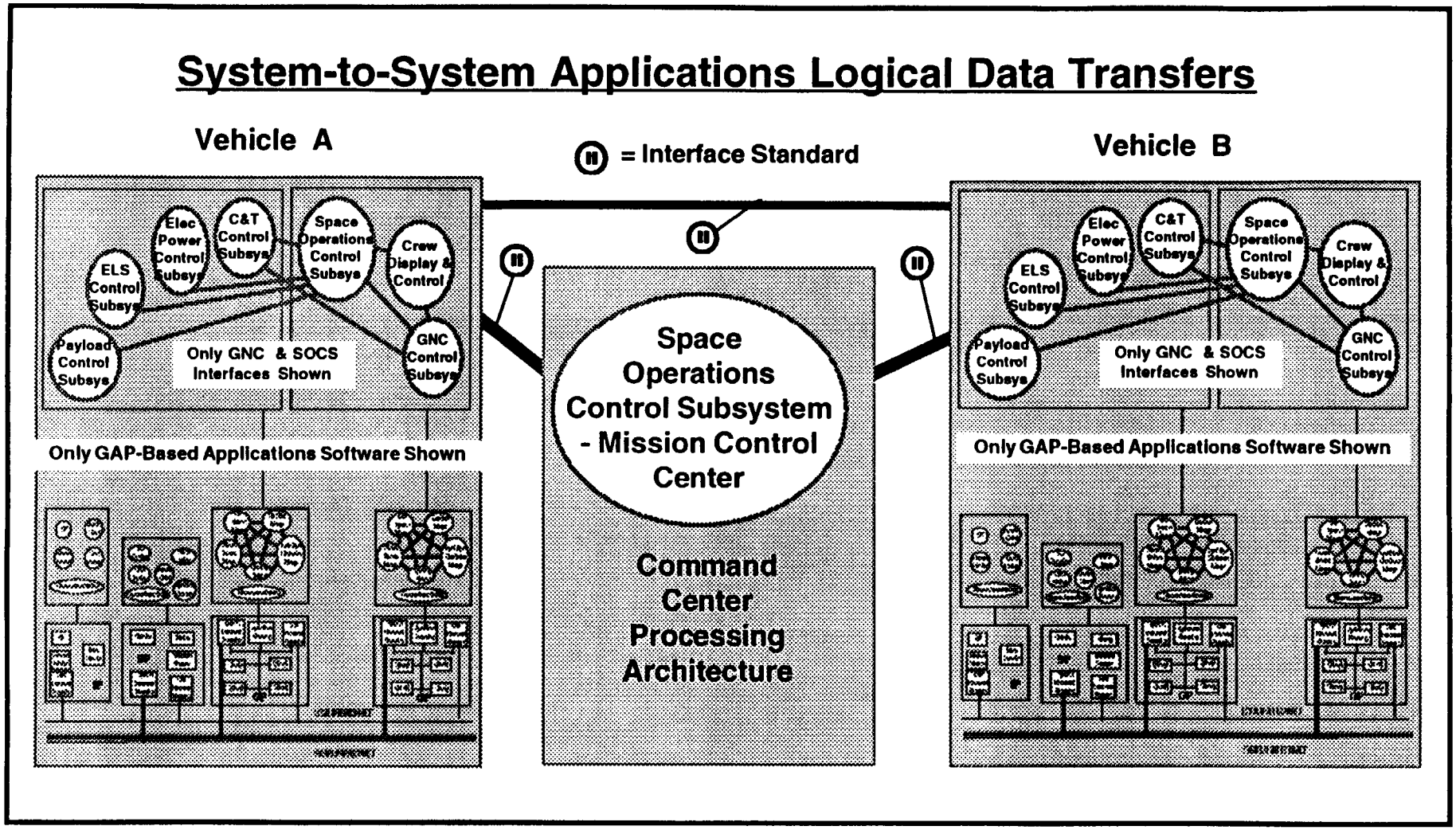


Figure 5-21. Class 4L System A Applications-to-System B Applications Logical Interfaces Example

6. NOTES

(This section contains information that may be helpful, but is not mandatory)

6.1 AVIONICS SYSTEM NOTES

6.1.1 AVIONICS GENERAL

Avionics provide for information acquisition, transmission, and storage of analog or digital signals and include the sensors, intra-platform communications, processing physical resources, software and subsystems, data storage, human-machine interface subsystems, and response actuator controls used in the vehicle.

6.1.2 MODE CONSIDERATIONS

Modes govern how the system operates in response to human commands. Mission ready mode means the system has all elements working as specified or "green". Operationally ready mode means the system can function but can not accomplish a desired mission, for instance when an aircraft is configured for a reconnaissance mission but is needed for a bombing mission, or when a spacecraft is can be launched but has no payload installed. Degraded mode means the system is "soft broke", but can perform a subset of its required functions. An example is when an aircraft radio is not working so not all functions can be performed but the aircraft can still fly and drop bombs. Red-tagged mode means that the system cannot operate at all, for instance when an aircraft fuel system is polluted or a wheel is broken on the ground.

6.1.3 SGOAA MODELS

The complete SGOAA consists of six classes of models as illustrated in Figure 6-1. The requirements for implementing the first three of these model classes is addressed in this standard. Use of the Generic Performance Model, Generic Dependability Model and the Generic Programmatic Model classes are not a requirement for the development of avionics systems using the SGOAA Standard. These models are recommended for consideration of use in any avionics system development program. The bulleted lists of models below each box in Figure 6-1 are examples of specific existing or in development models expanding on the SGOAA.

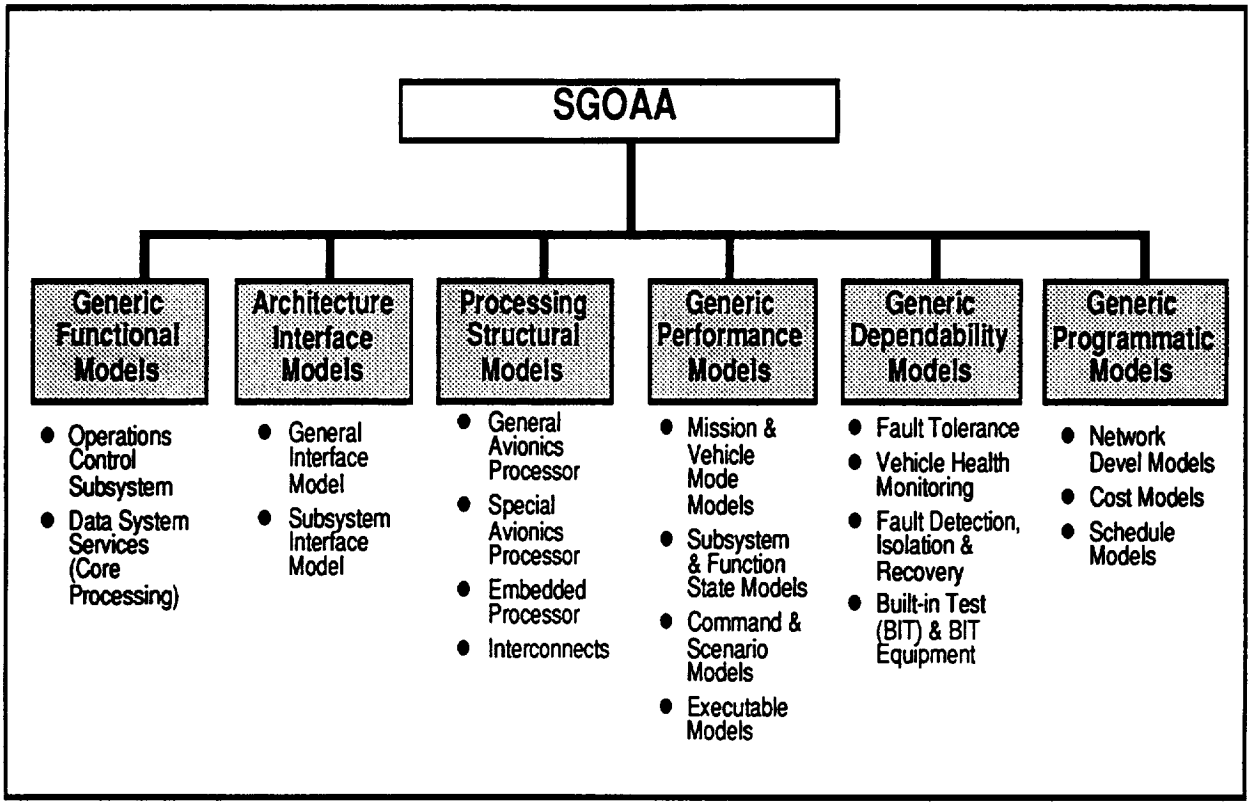


Figure 6-1. Key Elements of the SGOAA

6.1.4 ARCHITECTURE INTERFACE MODEL

The Architecture Interface Model is summarized in Figures 5-7 and 6-2. Figure 5-7 presented the Reference Architecture Interface Model, and Figure 6-2 presents an overlay of the reference model on the generic avionics structure assembled in this standard. In Figure 6-2, Level 1 is the Physical Resources Level, Level 2 is the Resource Access Services Level, Level 3 is the DSS level and Level 4 is the Application Level. Both figures also show the relationships of this architecture interface model to the POSIX P1003.0 interfaces.

6.1.5 COMPONENT PARTITIONING CRITERIA

The SGOAA processing components will typically be partitioned based on the criteria shown in Table 6-1. Any single physical resource being considered for use as a GAP, SAP or EP may fit any of these three categories, depending on the use to which the item is to be placed.

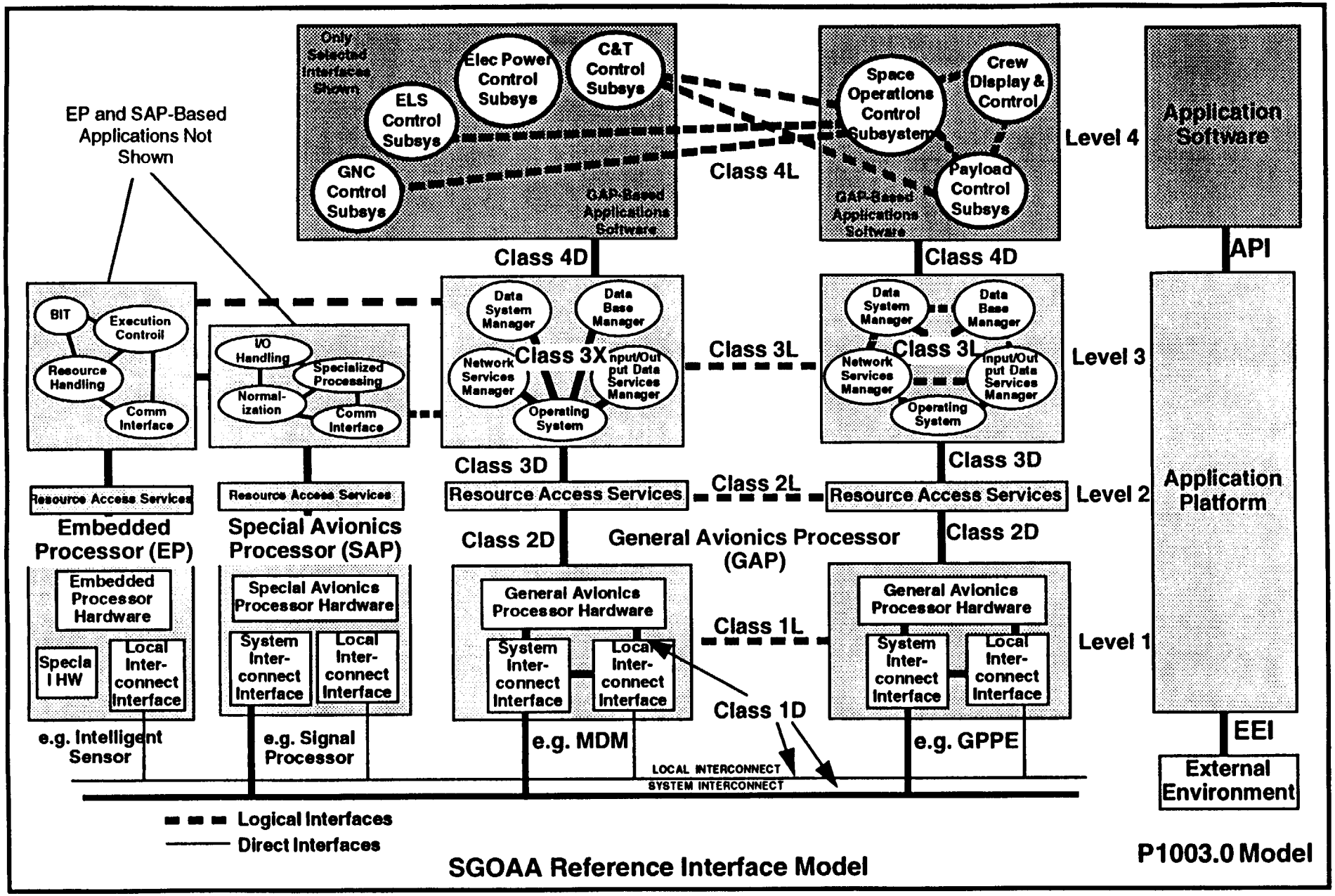


Figure 6-2. Generic Avionics Architecture Interface Model

Table 6-1. Component Partitioning Criteria

Component	Criteria
GAP	o Multi-purpose and multi-function o Loose coupling to sensors and effectors o Multiple inputs and outputs o Long latency (relatively)
SAP	o Limited purpose and function o Medium coupling to sensors and effectors o Multiple inputs and single output o Medium latency (relatively)
EP	o Single purpose and function o Tight coupling to one sensor or effector o Single input and output o Very short latency

6.2 REQUIREMENTS NOTES

6.2.1 DATA PROCESSING SUBSYSTEM

Data processing subsystem requirements are inherited onto lower level subsystems. A data processing subsystem is setup and controlled by a runtime operating system.

6.2.2 INTERSYSTEM APPLICATIONS INTERFACE

Intersystem applications interface requirements address *logical* requirements of the user across the interface.

6.2.3 CONTROL SUBSYSTEM

Control subsystem requirements are inherited onto lower level control subsystems.

6.2.4 MODULAR ARCHITECTURE

Modular architecture requirements changes should not cause disproportional changes in the design, and design changes should be limited to one or a few design modules. Requirements changes should not affect the architecture of the system, unless the change is one for the architecture directly.

6.2.5 Direct Interface

Direct interface requirements are normally a design issue unless the physical implementation has implications for the logical use or need for data, only then should the direct implementation be specified as a requirement. For instance, a service such as a Reports Generator getting data from a DBM might not need to know the inter-network addressing of the DBM, but the NSM providing the data would need to know the routing requirements of the physical resources services.

6.3 REQUIREMENTS CHARACTERISTICS DESIRED

6.3.1 ROBUSTNESS

Robustness must enable a system to operate in conditions not originally foreseen by the specification without catastrophic failures, without exhibiting behavior that disturbs the rest of the system, by failing (if necessary) in a "graceful" manner by terminating cleanly and safely.

6.3.2 SYSTEM SERVICES

System services must have common and standardized interfaces serving many applications.

6.3.3 SERVICE FUNCTIONS

Service functions are usually widely replicated in support of many control or data processing subsystems. This wide replication of functionality is a key determining characteristic in defining an individual process as a service in this methodology. Services are critical to system operation, not to mission or vehicle operation per se. An example of a service function is a Report Generator since many applications and control subsystems must generate reports; here, they call on the report generator service which knows how to look up the table defining the applications/control report, how to format the format for completion, how to find the data to fill the report fields with, and how to route the report for distribution based on a predefined distribution list. High level system services are services such as timing, distributed data handling and fault tolerance, which may have different needs when viewed as a multi-processing system than when considered as a single processor system.

6.3.4 TAILORING

Tailoring of the SGOAA may result in subsets of requirements applicable to a mission or program, but the resulting system must retain architectural interface compatibility.

6.3.5 SYSTEM CHARACTERISTICS

System characteristics that determine the nature and requirements for a system resource architecture are the number of processors, their type and topology, the speed and size of shared memory available, the local memory of each, the bandwidth and access to communications media, and the interfaces available for use by people, applications and platform system services in the physical resources.

6.3.6 ONBOARD HEALTH MANAGEMENT

Onboard health management facilitates tradeoffs between requirements for and design of reliability, maintainability, error processing, and fault treatment. Compliant architectures should also facilitate defining requirements for and interactions between health management, logistics, supportability, and safety management.

6.4 ARCHITECTURAL CHARACTERISTICS DESIRED

6.4.1 SYSTEM SOFTWARE ARCHITECTURE

A system software architecture must describe the set of system functions performed by the applications, and the structure of the platform system services that enable the applications to perform their tasks. The functionality described by the system software architecture are the tasks which are required of the system to meet the needs of operational users.

6.4.2 APPLICATION PLATFORM

The AP provides services at its interfaces that, as much as possible make the specific characteristics of the platform transparent to the application.

6.4.3 APPLICATION PROGRAM INTERFACE

The API is primarily in support of application portability, but system and application interoperability are also supported by the communications API.

6.4.4 ARCHITECTURE LOCATION INDEPENDENCE

Architecture location independence is desirable, meaning an architecture compliant with this standard should be independent of whether control functions are implemented onboard the vehicle or offboard the vehicle (e.g., in support control facilities).

6.4.5 FAULT TOLERANCE TRANSPARENCY

Fault tolerance transparency is desirable, meaning applications in an architecture compliant with this standard should not require knowledge of the redundancy of its platform or its direct interfaces.

6.4.6 ADAPTABLE REDUNDANCY

Adaptable redundancy is desirable, meaning that an architecture compliant with this standard should allow more than one configuration of the architecture to provide different levels of redundancy. This allows another level of commonality beyond that of physical resource and software modules.

6.5 DIRECT AND LOGICAL INTERFACE NOTES

6.5.1 CLASS 2D DIRECT INTERFACE

The Class 2D Resource Access Service to Physical Resources Direct Interface drivers are (obviously) physical resource dependent, but this enables the architecture to begin partitioning out of the physical resource dependencies, which is a key in providing for technology upgradability in the future.

6.5.2 HEALTH MANAGEMENT INTERFACE

Note that interface drivers specifically defined for health monitoring are not required. Each required driver will collect all data associated with its physical resource element and format it for conveyance to the appropriate operating system interfaces; if health monitoring capabilities have been implemented in the associated physical resources, then this data will be collected along with all other data.

6.5.3 CLASS 3L LOGICAL INTERFACES

Class 3L Data System Services-to-Data Services Logical Peer Interfaces provide the interface capability for services in one processor to interact with services in the same or another processor. They are the heart of multi-processor capability needed in modern space avionics

systems. EP services can interact with SAP and GAP services; SAP services can interact with GAP services; 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).

6.5.4 CLASS 3X DIRECT INTERFACES

The definition of Class 3X Operating System Services to Non-OSS Direct interface consists of the input/output handler calling conventions and context switch conversions between the system service drivers on one processing element interfacing with one or more system services to provide for local information exchange.

6.5.5 CLASS 4D DIRECT INTERFACES

Class 4D Data System Services-to-Applications Direct Interfaces provide the capability for services in any processor to interact with an application executing in the processor. 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 Class 4D interfaces are direct interfaces because the applications code is interacting with the service code.

6.5.6 CLASS 4L LOGICAL INTERFACES

The Class 4L Applications-to-Applications Logical Interfaces are logical interfaces because the application originating data is not directly interacting with applications that will use the data. Class 4L interfaces are also the interface for exchange of information between the space avionics system and another avionics system for overall command and control. This interface is at the mission level and may be an information exchange between the ground or between separate space vehicles.

Class 4L also provides the basic multi-system capability to meet multiple actual user requirements in multiple systems, facilities or vehicles. Applications can operate in any system's processor (e.g., the Mission Control Center GAP or workstation) to cooperate with applications in another system's processor (e.g., the Lunar Transfer Vehicle GAP).

6.6 IMPLEMENTATION CHARACTERISTICS

In implementation, tailoring of the generic architecture to the unique requirements of the mission and system application is critical to successful use of this architecture. Profiles need to be applied. Such usage must recognize the scaleability, recursiveness and interaction between target and development environments in order to take full advantage of the utility of this architecture.

6.6.1 ARCHITECTURE SCALEABILITY

The generic system architecture model can be scaled to apply to the size and definition of system being used in any type of physical resources/software processing system. It is equally applicable to systems at the vehicle level, rack or black box level, module or board level, or chip level, as shown in Figure 6-3. In this figure, examples of a system interconnect, local interconnect and internal interconnect change as the scale of system application changes. The use of embedded processor, general avionics processor and special avionics processor also changes. While only intended as an example, this figure makes clear that when two more engineers are applying the this architecture to a specific project, mission or system, great care must be taken to insure all discussions reference the same level of "system".

6.6.2 ARCHITECTURE RECURSIVENESS

The generic architecture interface model can be recursively applied to different usage's of layers (i.e., physical resources, drivers, OS, etc.) between the SGOAA classes in the architecture. As shown in Figure 6-4, from an external view, a physical resource such as an ethernet board is a monolithic block. However, from within the block, ethernet "physical resources" may consist of a microcosm of the entire interface model with board drivers, a OS kernel/RTE, some ethernet services and some ethernet processing applications all resident on the board. All of these microcosmic elements are transparent to an outside user passing through the interface.

6.6.3 ARCHITECTURE TARGET DEVELOPMENT

The architecture applies equally to any physical resources/software processing system. In the example shown in Figure 6-5, it applies to a development environment, where development software such as a compiler is an application, which must have knowledge of the host, the target and the programming language being used. Within the development environment, there are services, OS elements and drivers. The target code is operated on by

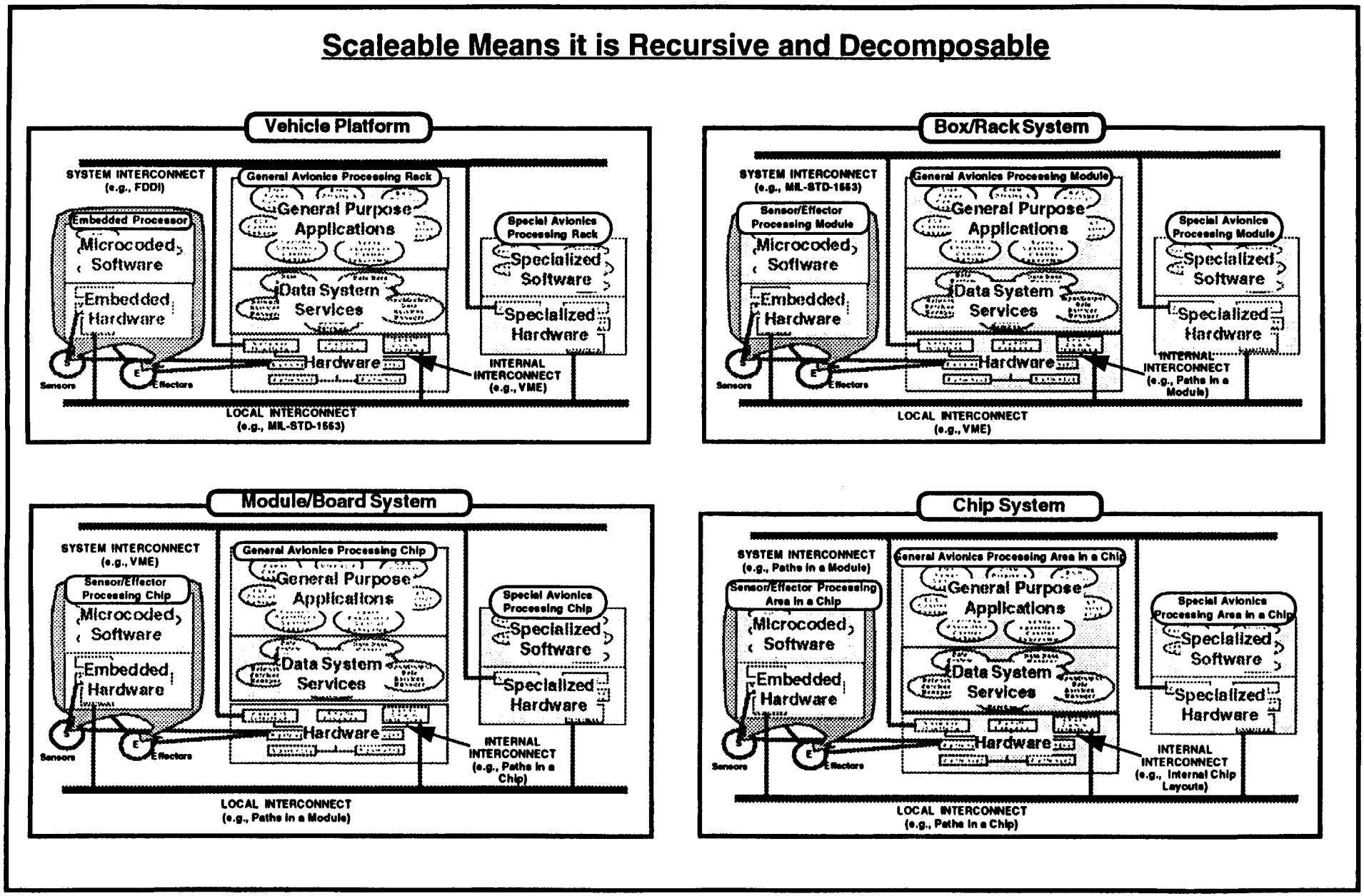


Figure 6-3. The Generic System Architecture Model Is Scaleable

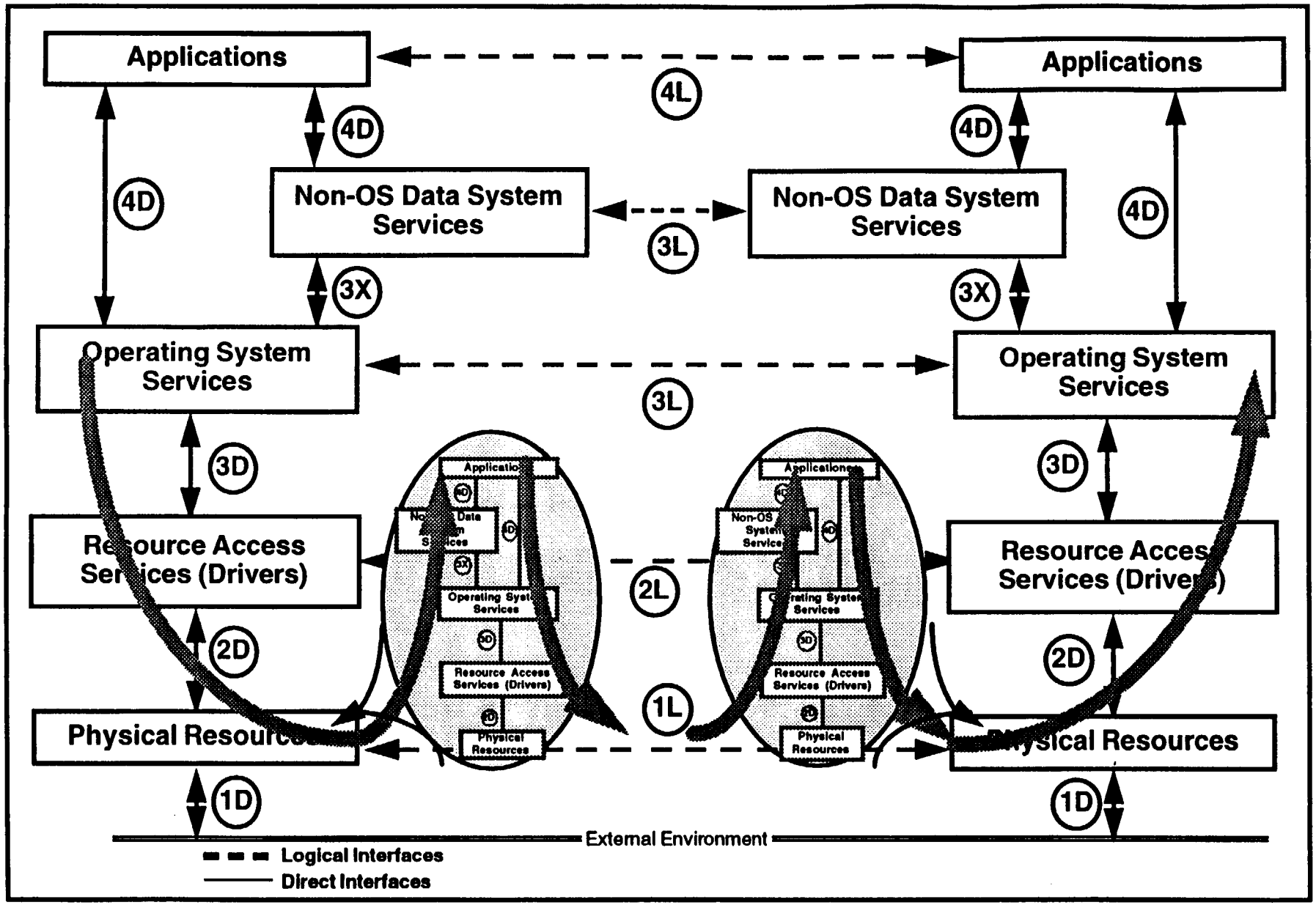


Figure 6-4. The Architecture General Interface Model Is Recursive

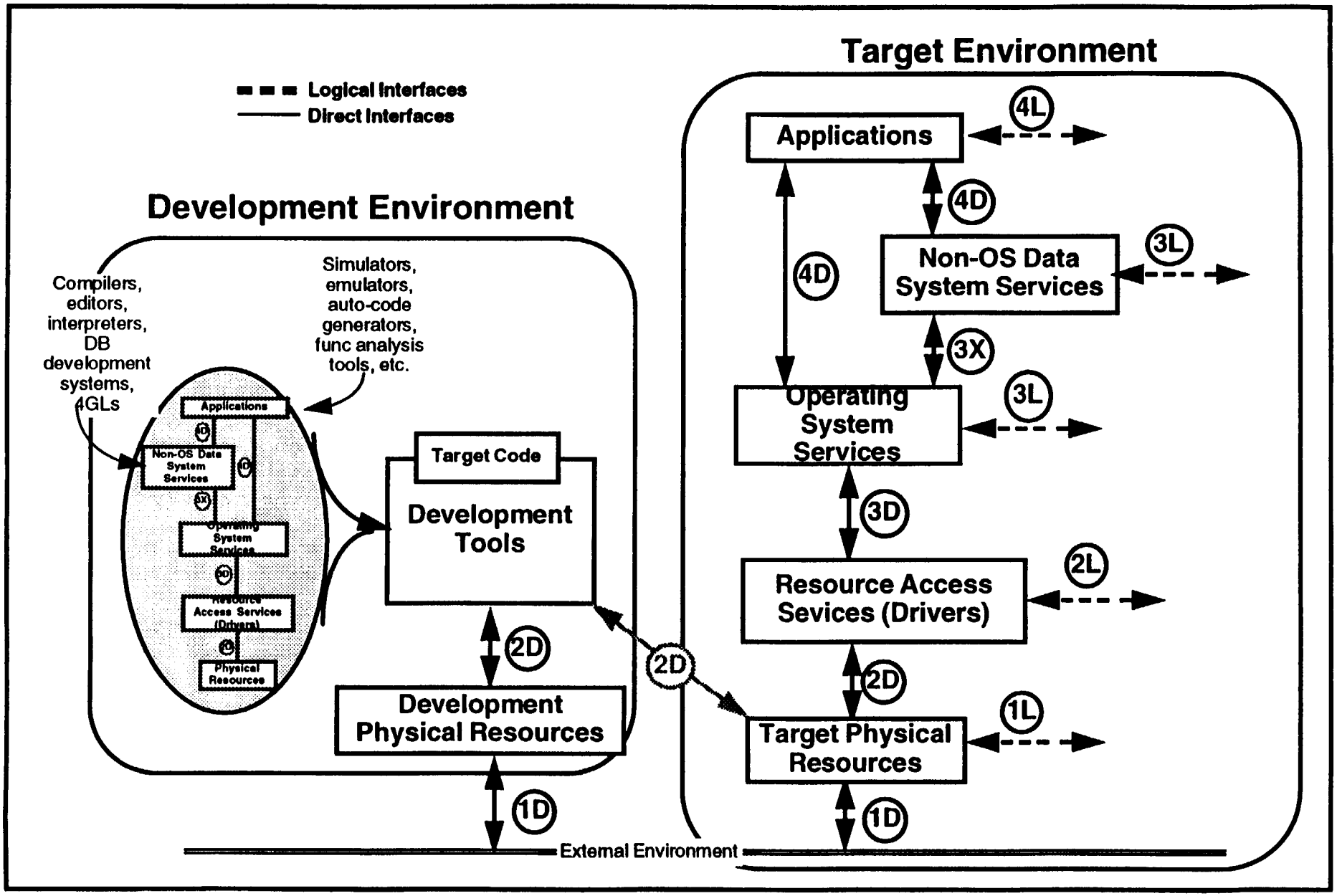


Figure 6-5. The Interface Model Applies to Both Target and Host Development Environments

the development environment and transferred upon completion to the target physical resources to execute. From the view of the target environment, this development environment is transparent.

6.7 TERMINOLOGY NOTES

In the space systems avionics arena, a data system is terminology used to refer to the set of operating system elements and supplementary or common services controlling processing resources in a host platform. This is similar to and includes the terminology system executive and distributed executive as used in PAVE PILLAR, global executive as used in parts of the F-22 Program, or common system services (e.g., an operating system) as used often in POSIX and other parts of the aircraft avionics arena. This standard differentiates between data system and DSS, in that a data system includes DSS, command processing, and other common capabilities (see Space Data System and Space Data System Services definitions in section 3.2).

6.8 PURPOSE OF PROFILES

As described in [POSIX91], profiles define the combinations of base standards and profiles (i.e., templates) for the purpose of:

- Identifying the baseline standards, together with appropriate classes, subsets, options, and parameters, that are necessary to at least accomplish interoperability, portability and other identified capabilities.
- Providing a system of referencing the various uses of baseline standards that is meaningful to both users and suppliers
- Enhancing the availability for procurement of consistent implementations of functionally defined groups of baseline standards that are expected to be the major components of real applications systems
- Promoting uniformity in the development of conformance tests for systems that implement the functions associated with the profiles

6.9 BIBLIOGRAPHY OF USEFUL DOCUMENTS

These are publications which offer insight into generic, open architectures and provide supplemental explanatory material for this standard.

[SP-M-001] "Contract End Item Specification for Data Management System, Vol. 1: Data Management System Requirements", Rev. E, (NASA Approval Pending), Feb. 14, 1992. Reference Document #1.

[SSP 30261] Section 3 Revision D "Data Management System Architecture Control Document Section 3: Data Management System" with Revisions D1 and D2, September, 1991.

[MDC H4187] " Software Requirements Specification for the Data Management System Data Storage and Retrieval", SSFP DR SY-34.1I, Contract No. 87916006, IBM, October 25,1991.

[MDC H4188] " Software Requirements Specification for the Data Management System Network Operating System", SSFP DR SY-34.1I, Contract No. 87916006, IBM, October 25,1991.

[MDC H4189] " Software Requirements Specification for the Data Management System Operating System/Ada Run time Environment", SSFP DR SY-34.1I, Contract No. 87916006, IBM, October 25,1991.

[MDC H4190] " Software Requirements Specification for the Data Management System Management", SSFP DR SY-34.1I, Contract No. 87916006, IBM, October 25,1991.

[MDC H4191] " Software Requirements Specification for the Data Management System Standard Services", SSFP DR SY-34.1I, Contract No. 87916006, IBM, October 25,1991.

[MDC H4542] "User's Guide (Software) for DMS Initial Release ", SSFP DR SY-40.1I, Contract No. 87916006, MDSSC, September 23, 1991. .

[IBM101] "Critical Item Development Specification for Mass Storage Unit", SSFP DR SY-06.2I, Contract No. 87916006, 153A101-PTIC, IBM, Oct. 9, 1992.

[IBM401] "Critical Item Development Specification for the Standard Data Processor", SSFP DR SY-06.2I, Contract No. 87916006, 152A401-PT1D, IBM, Oct. 9, 1992.

[IBM403] "Critical Item Development Specification for the Embedded Data Processor", SSFP DR SY-06.2I, Contract No. 87916006, 152A403-PT1D, IBM, Oct. 9, 1992.

[IBM404] "Critical Item Development Specification for the Network Interface Adapter", SSFP DR SY-06.2I, Contract No. 87916006, 152A404-PT1D, IBM, Oct. 9, 1992.

[PRU90] Pruett, D., "Avionics Software Open System Environment Reference Model", JSC, March 1990.

[WRA91] Wray, R.B., "Requirements Analysis Notebook for the Flight Data Systems Definition in the Real-time Systems Engineering Laboratory (RSEL)," Job Order 60-430, Contract NAS9-17900 for the JSC, LESC-29702, JSC CR-185698, December 1991.

[SA91] NASA Open System Architecture Study, Lockheed Sanders, August 27, 1991

DISTRIBUTION LIST FOR LESC-30354-C
SGOAA STANDARD - CONTINUED

OTHER LOCKHEED

LAD, P. O. BOX 17100, AUSTIN, TX. 78744-1016
CURTIS WELLS, O/ T2-10, BLDG 30F

LADC, P. O. BOX 250, SUNLAND, CA. 91041
ALEX LOEWENTHAL, DEPT. 25-14, BLDG 311

LAS Ontario, P. O. BOX 33, ONTARIO, CA. 91761-0033
C. R. (BOB) FENTON

LASC, 86 S. COBB ST.,MARRIETTA, GA. 30063
RICK HARWELL, D/73-D2, ZONE 0685
JOHN WEAVER, D/73-D1, ZONE 0685
HUDSON, ROCKY, D/73-D2, ZONE 0685
REED, MIKE, D/73-MA, ZONE 0081
COX, JIM, D/73-MA ZONE 0081

LFWC, P. O. BOX 748, FORT WORTH, TX 76101
PAUL DANIEL, MAIL ZONE 2640

LMSC, 1111 LOCKHEED WAY, SUNNYVALE, CA. 94088-3504
ROY PETIS, O/ 73-12, BLDG 564
JOHN McMORRIS, O/ 81-90, BLDG 157
DUWAYNE DICKSON, O/ 80-06, BLDG 154
MERLIN DORFMAN, O/ 62-80, BLDG 563
CHARLES TADJERAN, O/ 62-31, BLDG 150
RANDY FLEMING, O/ 73-12, BLDG 564
F. L. LORY, O/ 68-15, BLDG 104

LMSC (RD&D), 3251 HANOVER STREET, PALO ALTO, CA 94303-1191
BILL GUYTON, O/ 92-20, BLDG 254E
STEVE SHERMAN, O/ 96-10, BLDG 254E
TOM ARKWRIGHT, ORG 96-10, BLDG 254E
RAY MUZZY, O/ 90-21, BLDG. 254E

LOCKHEED-SANDERS, 95 CANAL ST., NASHUA, NH 03061
RAY GARBOS (NAM5D-5002)
JOHN MILLER (NCA 09-1106)
DAVE AIBEL
ANNETTE LANIGAN
JEFF E. SMITH (PTPZ-B002)
DUNCAN MOORE (MER 24)
WALT ZANDI

LOCKHEED CORP, 4500 PARK GRANADA BLVD, CALABASIS, CA 91399-0310
BART KRAWETZ
MICHAEL CARROLL

LSOC, 1100 LOCKHEED WAY, TITUSVILLE, FL 32780
L. J. (LEWIS) BOYD, O/ 32-40, (Z/LSO-183)
ARTHUR EDWARDS, O/ 11-42, BLDG. B/DX-D, Z/LSO-284)

DISTRIBUTION LIST FOR LESC-30354-C
SGOAA STANDARD - CONTINUED
OTHER GOVERNMENT

NIST/CSL, FRITZ SCHULTZ, BLDG 225, ROOM B266, GATHISBURG, MD. 20899

DoD

AMSEL-RD-C2-TS-I, FT MONMOUTH, NJ 07703
DOUG JOHNSON/ACC #66

ASC/ENAS, WRIGHT-PATTERSON AFB, OH 45433
FRED WILSON

MARTIN FREED, (ASC/ENASC), 5565 BARBANNA LANE, DAYTON, OH 45415

ASC/YFMXT, WRIGHT-PATTERSON AFB, OH 45433-7003
MR. BYRON STEPHENS/JIAWG

HQ USAF/SCS, 1250 AIR FORCE, PENTAGON, WASHINGTON, D. C. 20330-1250
COL ROBERT HANLON

NAVAL AIR WARFARE CENTER, AIRCRAFT DIVISION, WARMINSTER, PA 18974-0591
RICHARD J. PARISEAU/CODE 102A
RICHARD S. MEJZAK/CODE 2021

ROME LABS/OCTS, GRIFFIS AFB, NY 13441-5700
RICHARD WOOD

NAVAIR, (AIR-56M), 1421 JEFFERSON HIGHWAY, ARLINGTON, VA, 22243
CMDR LARRY JAHNKE

SAE/AEROSPACE

SAE/ASD, SAE INTERNATIONAL, 400 COMMONWEALTH AVE, WARRENDALE, PA. 15096
BARBARA ROTH (FILE) RICH VANDAME

BOEING CORP, PO BOX 3999, SEATTLE, WA 98124-2499
RICHARD FLANAGAN
AL COSGROVE

CTA INC, SUITE 310, 18333 EGRET BAY BLVD, HOUSTON, TX 77058
MR DAVID COOPER

COMPUTING DEVICES INTL, 8800 QUEEN AVENUE SOUTH, BLOOMINGTON, MN 55431
JIM JAMES, M/S BLCS1D
DOCK ALLEN, M/S BLCW2S

DISTRIBUTION LIST FOR LESC-30354-C
SGOAA STANDARD - CONTINUED

C.S. DRAPER LABS, 555 TECHNOLOGY SQUARE, CAMBRIDGE, MA 02139
J. BARTON DEWOLFE/MS 61

E-SYSTEMS, P. O. BOX 12248, ST. PETERSBURG, FL. 33733-2248
JIM BRADY/MS29

E-SYSTEMS, P.O. BOX 660023, DALLAS, TX 75266-0023
TIM SMITH/MC 4-47310

EER SYSTEMS INC., 3027 MARINA BAY DR., SUITE 105,
LEAGUE CITY, TX 77573
RAY HARTENSTEIN

FAIRCHILD SPACE, 20301 CENTURY BLVD., GERMANTOWN, MD. 20874
JOHN SCHNEIDER, FLIGHT DATA SYSTEMS

GE, CORPORATE RESEARCH & DEVELOPMENT, PO BOX 8, SCHENECTADY, NY 12301
DAVID W. OLIVER/BLDG K-1, RM 3C3

GOODRICH AEROSPACE, SIMMONDS PRECISION PRODUCTS INC., AIRCRAFT SYSTEMS, PANTON ROAD, VERGENNES, VT 05491
JOHN D. BLAIR

HONEYWELL INC, 3660 TECHNOLOGY DR, MINNEAPOLIS, MN 55418
MR RON FRAZZINI

HUGHES AIRCRAFT, PO BOX 92426, LOS ANGELES, CA 90009-2426
JOHN GRIFFITH, RE/RI/B500

MCDONNELL DOUGLAS CORPORATION, 1801 E. ST ANDREW PLACE, SANTA ANA, CA 92705
TERRY RASSET/MS A208

MITRE CORPORATION, 202 BURLINGTON ROAD, BEDFORD, MA 01730-1420
WILLIAM T. BRANDOM/D-96
JACK SHAY/DIRECTOR OF SYSTEMS DEVELOPMENT

NAVMAR APPLIED SCIENCES CORP, 65 WEST STREET, SUITE C200, WARMINSTER, PA 18974
MR. DOUG D'AVINO

NORTHROP B-2 DIVISION, 8900 E. WASHINGTON BLVD., PICO RIVERA, CA 90660
JIM NELSON/DEPT T216/GB

DISTRIBUTION LIST FOR LESC-30354-C
SGOAA STANDARD - CONCLUDED

PARAMAX SYSTEMS CORP, PO BOX 64525, ST PAUL, MN 55164-0525
MR DARYLE HAMLIN/MS U1F15

RESEARCH ANALYSIS AND MAINTENANCE INC., 512 AUDOBON ST., LEAGUE CITY, TX 77573
ROGER EVANS

ROCKWELL INT'L CORP.- SPACE TRANSPORTATION SYSTEMS DIV., 12214 LAKEWOOD BLVD., DOWNEY, CA. 90241
BURTON SMITH, M/S FA20

ROCKWELL INTL CORP.- ROCKETDYNE DIV., 6633 CANOGA AVE, P. O. BOX 7922, CANOGA PARK, CA. 91309-7922
ANTHONY THOMPSON, D1055-LB33

SBS ENGINEERING, 5550 MIDWAY PARK PLACE, NE, ALBUQUERQUE, NM 87109
MR. DEREK HEAD

SOFTWARE ENGINEERING INSTITUTE, CARNEGIE MELLON UNIVERSITY, PITTSBURGH, PA 15213
B. CRAIG MEYERS

TEXAS INSTRUMENTS, 6550 CHASE OAKS BLVD, PO BOX 869305, PLANO, TX 75086
DR. CHUCK ROARK/MS 8481

TRW, HOUSTON, TX 77058
DOUG RUE (NASA MAIL)

WESTAR CORP. 6808 ACADEMY PKWY EAST, NE, BLDG C, SUITE 3, ALBUQUERQUE, NM 87109
CHRIS DE LONG

UNIVERSITIES & OTHERS

UHCL, UNIVERSITY OF HOUSTON - CLEAR LAKE, 2700 AY AREA BLVE. - BOX 444, HOUSTON, TEXAS 77058
CHARLES HARDWICH

UNIVERSITY OF TEXAS AT AUSTIN, PO BOX 8029, 10000 BURNET ROAD, AUSTIN, TX 78713-8029
GRANVILLE OTT/APPLIED RESEARCH LABORATORIES

NCOSE TGCC, 1907 BELLMEADE, HOUSTON, TX 77019
ED SMITH