SAE Aerospace
An SAE International Group

AEROSPACE STANDARD

SAE AS4893

Issued 1996-01
Reaffirmed 2011-04

Generic Open Architecture (GOA) Framework

FOREWORD

The Generic Open Architecture (GOA) development was initiated to develop a framework which can be used to classify interfaces needed in airborne avionics systems. At the time of the development of the GOA, development of such a classification was considered a crucial part of transitioning open systems standards to military avionics. However, it was determined during the development of the GOA that the GOA Framework is applicable to domains other than avionics. For that reason the framework is entitled Generic Open Architecture instead of the original name, Generic Open Avionics Architecture (GOAA).

The GOA effort was fortunate that a suitable base document existed as a starting point for its definition. The base document for the GOA standard is the Space Generic Open Avionics Architecture (SGOAA), NASA CR-188269. The SGOAA was 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 NASA CR-188269. The contributions of Mr. Ben Doeckel of LESC who participated in early development of the concepts for the avionics architectures and standards represented in the SGOAA is acknowledged. Special acknowledgment is also given to Mr. Dave Pruett of the Johnson Space Center for his support of the Advanced Architecture Analysis, assistance in the development of the avionics architecture and constructive criticisms of the SGOAA.

The GOA is an evolution of the SGOAA. This evolution occurred through the work of several diligent people who made up the SAE AS-5 GOA Task Group. This standard was prepared under the direction of:

Chuck Roark — Chairman, AS-5 Committee
Texas Instruments Communications & Electronics Systems
7839 Churchill Way, MS 3933
Dallas, Tx 75251

Terry Rasset — Chairman, AS-5 GOA Task Group
McDonnell Douglas Aerospace
5301 Bolsa Avenue, MS 46S-1
Huntington Beach, Ca 92647

TO PLACE A DOCUMENT ORDER: Tel: 877-606-7323 (inside USA and Canada)
Tel: 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
SAE WEB ADDRESS: http://www.sae.org

1. SCOPE:

1.1 Scope:

This SAE Aerospace Standard (AS) establishes a Generic Open Architecture (GOA) Framework for application independent hardware/software systems. This document defines the interface classes for the GOA Framework. Supplemental documents define the guidelines for applying the GOA Framework to specific applications.

1.2 Purpose:

The purpose of this document is to provide a framework to identify interface classes for applying open systems interface standards to the design of a specific hardware/software system. This framework is used to define an abstract architecture based on a generic set of interface points. The generic set of system interface points facilitate identification of critical interfaces.

It is intended that the GOA Framework be specialized for varying domains. A domain specific implementation of the GOA Framework will increase the chance that components/capabilities produced independently will "plug and play" and evolve affordably within the domain. The GOA Framework provides a basis for commonality for both vendors and users of components/ capabilities. Application of the GOA Framework will impose constraints on individual domains and implementations. This will increase the likelihood that independently produced products will interoperate.

Application of the GOA Framework together with the appropriate open system interface standards is expected to provide the following benefits to future programs:

a. Provide the basis for establishing a set of specifications, standards and procedures that will become common to all elements of a major system.

b. Ensure that future systems can be upgraded and maintained with minimal redesign impact to the existing system by establishing the interfaces required to enable modular replacement of hardware and software.

c. Promote availability of multiple sources of needed software and hardware, especially commercial off-the-shelf components.

d. Provide a pool of hardware and software modules for multiple program commonality and re-use.

e. 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.

1.3 Application Guidance:

This document is intended to be used by both system designers and system implementors in the development of open systems architectures. Domain specific guidelines should be developed to provide clarification for application of the GOA Framework.

1.4 Document Structure:

This document is structured as follows:

a. Section 1 defines the scope and purpose of this document.

b. Section 2 lists documents referenced within this document, and definitions and terminology used in this document.

c. Section 3 presents GOA Framework requirements.

d. Section 4 provides applicable notes to the GOA Framework standard.

2. REFERENCES:

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:

ISO	International Organization for Standardization 7498: Information Processing Systems - Open Systems Interconnection - Basic Reference Model
MIL-STD-499B	"Draft Systems Engineering Standard", AFMC, 1994
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:

SGOAA	Space Generic Open Avionics Architecture (SGOAA) Standard Specification, LESC-30354-C (NASA CR-188290), Lockheed Engineering & Sciences Company, June 1994

2.3 Other Publications:

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, April 1993

2.4 Definitions:

2.4.1 APPLICATION: Application is the capability (service/function) provided by a system specific to the satisfaction of a set of user requirements. [Derived from POSIX91]

NOTE: Application refers to the entire system function that may be implemented in part by application software.

2.4.2 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]

2.4.3 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] API is a GOA 4D interface.

2.4.4 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] Application Software is a GOA layer.

2.4.5 ARCHITECTURE: Architecture describes the components, organization, and interfaces of a system.

2.4.6 ARCHITECTURE FRAMEWORK: Architecture Framework is a skeleton that serves to organize a systems architecture without dictating the detailed implementation. The GOA Framework emphasizes the interface between interconnecting components.

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

2.4.8 DIRECT INTERFACE: Direct Interface is an interface that defines a service/consumer relationship between adjacent GOA layers in the GOA model. Direct Interface is the connection between an entity sending (or receiving) data with another entity receiving (or sending) data for transmission of that data along the routing path. (See logical interface.)

2.4.9 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.

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

2.4.11 EXTERNAL ENVIRONMENT INTERFACE: External Environment Interface (EEI) is the interface between the application software or 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. [Derived from POSIX91] EEI are GOA 1D, 1L, 2L, 3L, and 4L interfaces that are external to a platform.

2.4.12 GOA INTERFACE CLASS: GOA Interface Class is a direct or logical interface as defined in Section 3 of this document.

2.4.13 GOA LAYER: GOA Layer is an abstract set of entities, associated data attributes and essential services connected by GOA direct interface(s).

2.4.14 INTERFACE: Interface is the shared boundary between functional units.

2.4.15 LOGICAL INTERFACE: Logical Interface is an interface that defines a peer-to-peer relationship between entities within the same GOA layer of the GOA model. 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. (See direct interface.)

2.4.16 OPEN SPECIFICATION: Open Specifications are public specifications that are maintained by an open, public consensus process. The public consensus process for open specifications must be maintained and accepted by an open forum. [Derived from POSIX91]

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

a. To be ported with minimal changes across systems conforming to the same open specifications
b. To interoperate with other applications on local and remote systems
c. To interact with users in a style that facilitates user portability

2.4.18 OPERATING SYSTEM SERVICES AND EXTENDED OPERATING SYSTEM SERVICES:

Operating System (OS) Services and eXtended Operating System (XOS) Services are secondary GOA layers within the System Services GOA layer. The OS Services consist of core “executive” type functions that are responsible for managing system resources such as processor execution, memory, timing, and input/output for use by applications and System Services. The XOS Services consist of modular extensions to the OS Services as well as higher level functions (e.g., database manager) that are shared by multiple applications.

2.4.19 PHYSICAL RESOURCES: Physical Resources are those functions based in hardware in the application platform. Physical Resources is a GOA layer.

2.4.20 PLATFORM: See Application Platform definition.

2.4.21 PROTOCOL: Protocol is a set of semantic and syntactic rules that must be followed to perform communications functions within a communications system. (Derived from [POSIX 91]).

2.4.22 RESOURCE ACCESS SERVICES: Resource Access Services are those low level services (e.g., drivers) which enable the System Services to interact with physical resources. Resource Access Services is a GOA layer.

2.4.23 SERVICE: Service is the work performed for a subsystem , application, or end user. An end user can be any entity that uses the service.

2.4.24 SOFTWARE: Software is the programs, procedures, rules, and any associated documentation pertaining to the operation of a system containing one or more programmable processors. [Derived from POSIX91]

2.4.25 SOURCE: Source is the originator of data passed across an interface.

2.4.26 SYSTEM: System is the composite of equipment, material, computer software, personnel, facilities and information/procedural data that satisfies a user need [SYSB-1]. A system is an integrated composite of people, products, and processes that provide a capability to satisfy a stated need or objective [MIL-STD-499B].

2.4.27 SYSTEM SERVICES: System Services (SS) provides a comprehensive set of common services in support of applications. The System Services isolates the Application Software from the Application Platform’s Physical Resources. The System Services is a GOA layer that contains two secondary GOA layers: OS Services and XOS Services.

2.4.28 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.

2.5 Terminology:

2.5.1 IMPLEMENTATION DEPENDENT: Implementation Dependent means the implementation is to define and document the requirements for architectures, software constructs, correct data values or behavior. (Derived from [POSIX91])

2.5.2 INFORMATIVE: Informative is providing or disclosing information which is only instructive and not preceded by a "shall" (Derived from [POSIX91]) Such material poses no requirements, and is primarily located in Section 4, Notes.

2.5.3 NORMATIVE: Normative is prescribing or directing a norm or standard; used in standards to indicate text which poses requirements. (Derived from [POSIX91]) Each section of this document is normative, except for sections 4. Non-normative parts of each section are explicitly indicated.

2.5.4 SHALL: Shall, in implementation, indicates a requirement for implementors.

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

3. ARCHITECTURE INTERFACE DETAILED REQUIREMENTS:

The GOA Framework establishes a structure for hardware, software and interfaces that implement the functions for systems in varying domains. The GOA Framework is a set of nine interface classes partitioned into logical and direct interface classes. This description of the nine required classes includes both system service software and Application Software for systems and subsystems. Interfaces in this framework are valid for both one platform and multi-platform architectures (if applicable).

The framework's nine interface classes are partitioned into 4 logical and 5 direct interface classes. Logical interfaces define what information is exchanged while direct interfaces define how the information is transported. Figure 1 illustrates a logical interface between Application Software entity A and entity B with the direct interfaces that provide the services to accomplish the information exchange between them. These interface detailed requirements define the GOA Framework's nine interface classes.

This framework is used to organize system (and lower level) requirements and define how those requirements are applied at an appropriate system level to determine the logical and direct interface points. In a given instance of this framework, system logical data flow requirements should be created for each communicating entity addressing the data attributes needed by that entity or needed to be provided to some other entity. The data flow requirements of the logical interface identify the source of the data and the end-user needing the data, as well as the characteristic attributes required of the data and the coordination needed by them.

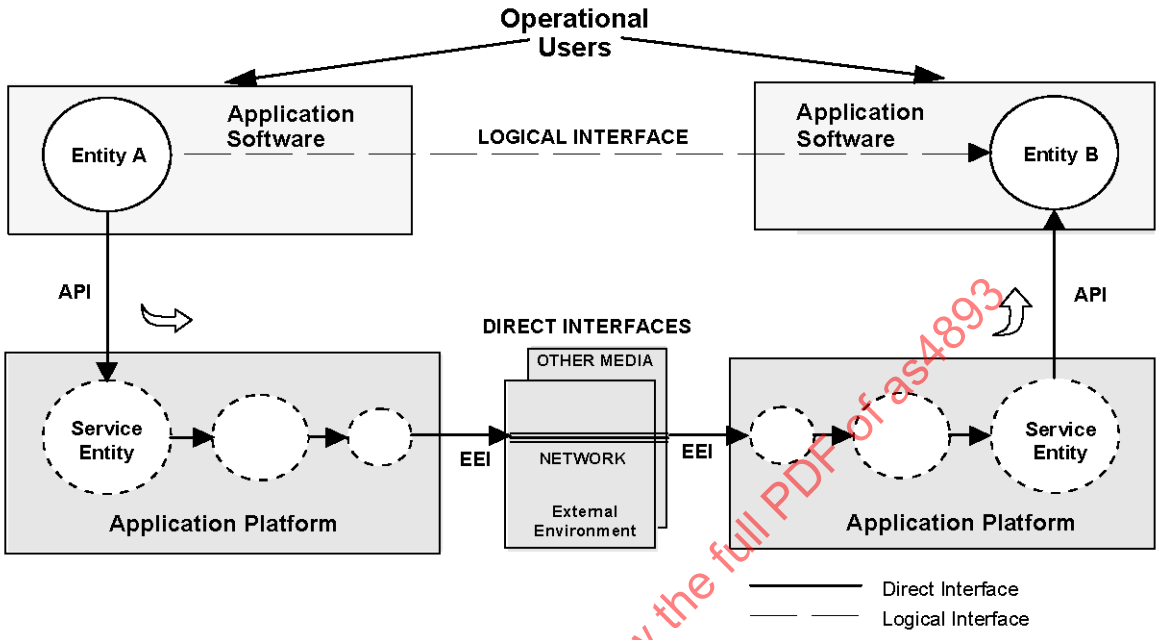

FIGURE 1 - Logical Interfaces are Implemented by Means of Direct Interfaces

3. (Continued):

Logical data flow requirements should not be concerned with the mechanism for implementing the data interchange. Implementation related requirements for the interfaces belong to a direct interface class which defines the mechanisms provided to flow the data from the source to the end-user. Sources of such design requirements for the interfaces, application platform hardware and application platform services should be derived from the Application Software requirements and their logical data attribute requirements based on the user's needs.

3.1 Architecture Framework Requirements:

The GOA Framework consists of nine classes of interfaces as shown in Figure 2, Figure 3, and defined in Table 1. Figure 2 shows the GOA Framework within the context of a POSIX environment, while Figure 3 shows the GOA Framework within the context of two separate application platforms. These classes are the levels of interfaces from Physical Resource up to systems of Application Software which are to be completely defined in an architecture developed in accordance with this standard. Definition of each interface class shall [1] be in accordance with the requirements contained in the following paragraphs. An architecture that is partitioned within the definition of the nine GOA classes of interface will be consistent with the GOA Framework.

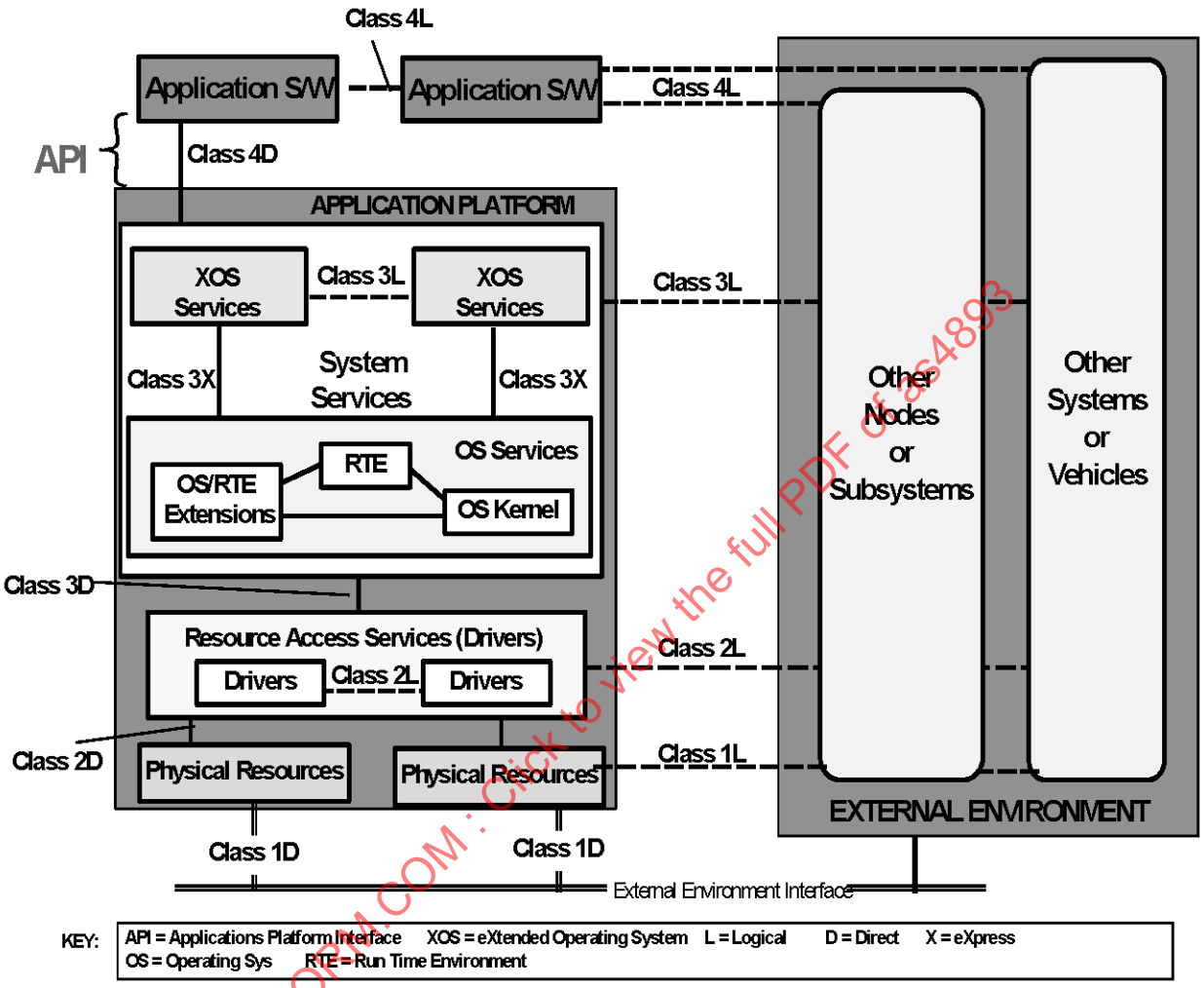

FIGURE 2 - GOA Framework - View 1

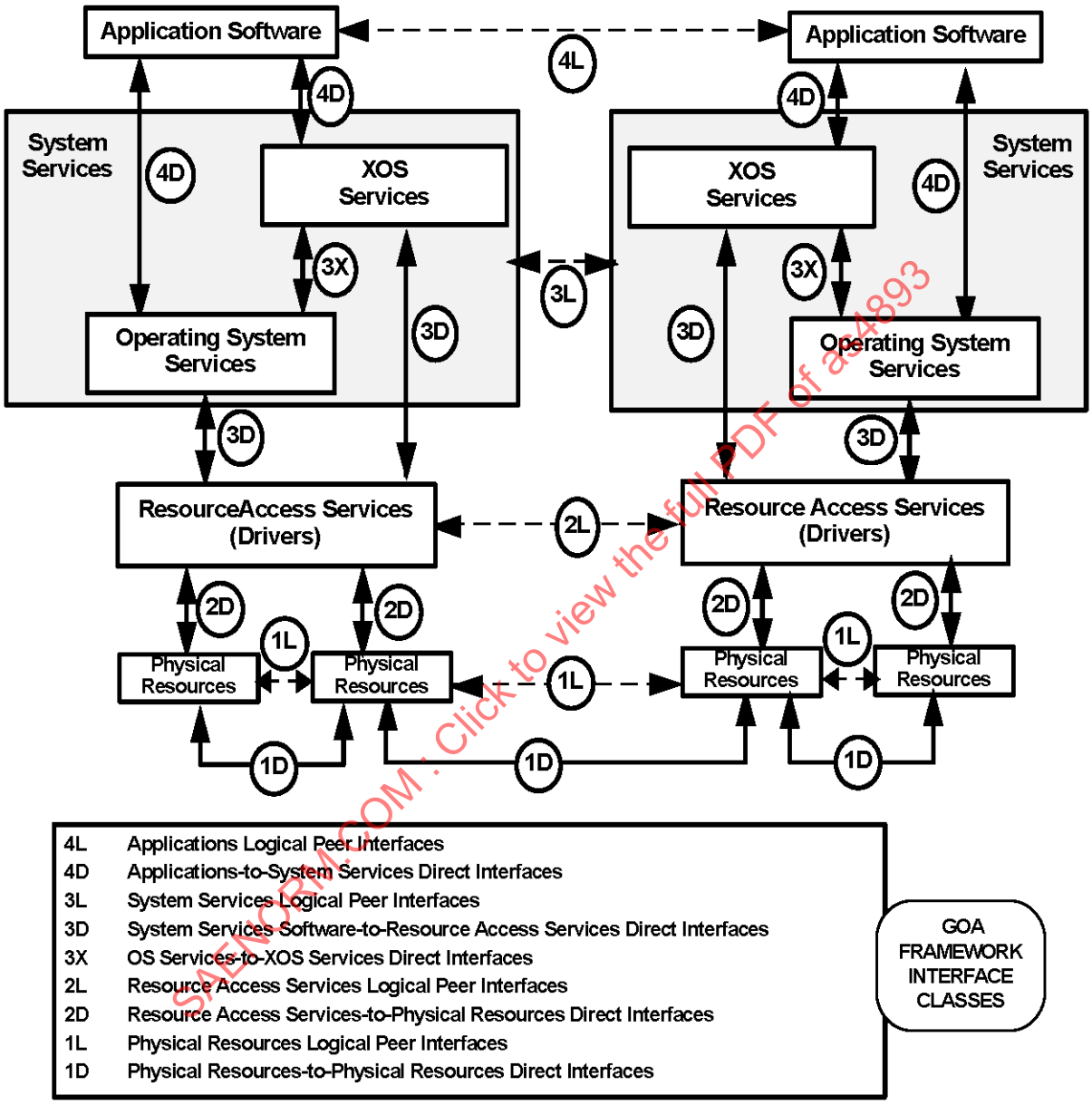

FIGURE 3 - GOA Framework - View 2

TABLE 1 - GOA Interface Classes

Class	Description
4L	Applications Logical Peer: Class 4L applications logical peer 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, including host/target environments.
4D	System Services-to-Applications Direct: Class 4D is the API with System Service code.
3L	System Services Logical Peer: Class 3L System Services logical peer 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 or host/target environments 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.
3X	OS Services-to-XOS Services Direct: Class 3X OS Services to XOS Services direct interfaces are the direct connections between OS Service code and XOS Service code sets, which enable OS Services to conduct privileged and other interactions with local XOS service code.
3D	System Services-to-Resource Access Services Direct: Class 3D System Services to resource access services direct interfaces are the direct connections between System Service code and Resource Access Service code sets. This enables System Services to receive and interpret data packets, and pass them on to other service code which will process them locally. It also enables formulation and transmission of data packets.
2L	Resource Access Services Logical Peer: Class 2L Resource Access Services logical peer interfaces are the requirements for establishing a data interchange interface between Resource Access Services in one node with those of the same or another local/remote node or host target environment which enable low level peer data exchange such as between drivers for different hardware.
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.
1L	Physical Resources Logical Peer: Class 1L Physical Resources logical peer interfaces are the requirements for establishing a data interchange interface between Physical Resources, such as those that enable bus or communications link boards to address their peers in another node or system .
1D	Physical Resources-to-Physical Resources Direct: Class 1D Physical Resources to Physical Resources direct interfaces are the direct connections between Physical Resources, such as those needed to enable buses and communications links to address processors or needed to enable processors to address memory registers.

3.1 (Continued):

The GOA Framework is a hierarchical model. There are four primary GOA layers within the GOA Framework: Application Software, System Services, Resource Access Services, and Physical Resources. The System Services GOA layer consists of two secondary GOA layers: Operating System Services and eXtended Operating System Services.

3.1.1 Class 4L - Application Logical Peer Interface Requirements: The application logical peer interface shall [1] be a peer to peer information exchange and coordination interface between software applications. This interface may be between Application Software executing on the same application platform or between application software executing on separate application platforms. Since Classes 1D to 4D isolate the Physical Resources, System Services, and applications in any processor, class 4L shall [2] provide the interface capability for Application Software in any processor to interact with other Application Software executing in any processor. Application logical peer interfaces may also include interfaces between Application Software in two different systems.

3.1.2 Class 4D - System Services-to-Applications Direct Interface Requirements: System Services to applications direct interfaces shall [1] provide the direct interface between the Application Software and the System Services (language bindings/specification) executing on the same application platform to allow provision of needed services. Since Classes 1 to 3 isolate 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 Application Software executing in the processor.

3.1.3 Class 3L - System Service Logical Peer Interface Requirements: System Services logical peer interfaces are the peer to peer interface of System Services. The interface may be within a single application platform or between different application platforms to coordinate operations in a distributed environment. Since classes 1D through 3X isolate the Physical Resources and System Services in each processor, 3L shall [1] provide the interface capability for services in one processor to interact with services in the same or another processor. Local services and remote services shall [2] have a common logical interface.

3.1.4 Class 3X - OS Services-to-XOS Services Direct Interface Requirements: The OS Services to XOS Services direct interface shall [1] consist of the interfaces between the OS Services and the XOS Services that comprise the System Services. Class 3X provides privileged and other interfaces needed to insure effective XOS Services performance. Class 3X provides the interface necessary to extend a given operating system. This interface supports the modularized expansion of the system services without modifying the given operating system.

Class 3X shall [2] provide the direct interfaces between the OS Services to local XOS Services for effective local interprocess communications and support. Although the OS Services are a subset of the System Services, they shall [3] also provide direct low level OS Services access not provided by the 4D System Services interface for System Services functions requiring this type of interface.

3.1.5 Class 3D - System Services-to-Resource Access Services Direct Interface Requirements:

System Services to Resource Access Services direct interfaces shall [1] consist of the interfaces between the System Services to the Resource Access Services. An example of a System Services to Resource Access Services direct interface is the interfaces for low level service drivers that interact with the Physical Resources. For systems in which the Resource Access Services and System Services are implemented in software, the 3D interface defines a boundary between the hardware specific portion of the system and the hardware independent portion of the system.

3.1.6 Class 2L - Resource Access Services Logical Peer Interface Requirements: Resource Access Services logical peer interfaces shall [1] consist of the requirements for peer-to-peer information/ data exchange and coordination interface between Resource Access Services within the same application platform or between separate application platforms. This class shall [2] define the interfaces that enable information/data exchange and coordination between the low level service drivers.

3.1.7 Class 2D - Resources Access Service-to-Physical Resources Direct Interface Requirements:

Resource Access Services to Physical Resources direct interfaces shall [1] consist of the interfaces from the Resource Access Services to the hardware instruction set architecture (ISA) and register usage.

3.1.8 Class 1L - Physical Resources Logical Peer Interface Requirements: Physical Resources logical peer 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.

3.1.9 Class 1D - Physical Resources-to-Physical Resources Direct Interface Requirements: The Physical Resources to Physical Resources direct interfaces consist of the nuts and bolts, chips and wires of the system architecture model. This interface shall [1] consist of all the Physical Resources to Physical Resources interfaces within each application platform, as well as the Physical Resources interfaces to the external environment.

4. NOTES:

4.1 Relationship Between GOA, POSIX, and ISO OSI Models:

Figure 4 demonstrates how the ISO Open System Interconnect (OSI) communications model, the POSIX model, and the GOA Framework complement one another. Both the GOA Framework and ISO OSI communications model expand the view of the application platform within the POSIX model. The ISO OSI communications model deals solely with communication where as the GOA Framework is not limited to communication.