OPEN SYSTEMS AND INDUSTRY STANDARDS IN FLIGHT SIMULATION: WHAT DO THEY PROMISE, CAN THEY DELIVER?

Bruce Johnson
Harris Computer Systems Division
Houston, Texas

Michael Caffey
FlightSafety International
Tulsa, Oklahoma

Mark Easter
FlightSafety International
Tulsa, Oklahoma

ABSTRACT

In the flight simulation industry today, computer industry standards and open systems architecture are dramatically influencing computer system selection, hardware/software design, and applications software development. What is an "open" systems simulator design? While open systems design has often been defined to mean the selection of a particular operating system and/or computer language for a host computer, it actually encompasses much more. A true open systems design impacts both hardware and software across all the systems and components that constitute a simulator.

Industry standards are offering very enticing promises of lower systems cost and complete portability of code. Yet how genuine are these promises and will flight simulation manufacturers and end-users truly benefit from a design that fully embraces open systems and industry standards? Beyond delivery, how will open systems and industry standards affect the logistical support of future simulators and training devices?

This paper will explore these issues and provide some answers to these questions. It reports of and draws upon the recent experiences of the Simulator Systems Division of FlightSafety International during their development of a completely portable simulator design. This design effort utilized industry standards to produce a flight simulator that is portable across multiple host computer platforms. The design effort involved months of development work on three different computer platforms (a Concurrent 8000, a Harris Night Hawk 4000, and an IBM RS/6000™). Far from a trade study, the design effort culminates this year with the delivery of a simulator to a FlightSafety customer.

Both the benefits and consequences of a standards-based design will be discussed based on the lessons learned in this effort. In addition, the trends in industry standards will be evaluated to predict their effect on future simulation development efforts.

ABOUT THE AUTHORS

Mr. Bruce Johnson is a Regional Analyst with Harris Computer Systems Division working in their Houston, Texas office. He has spent the past eight years either working in or consulting to the simulation and training industry. He received his Bachelor of Science Degree in Engineering and Computer Science from the University of Florida in 1981.

Mr. Michael Caffey is a Principal Engineer at FlightSafety International specializing in operating systems and Ada software design. He has been the lead system architect on several simulation projects and is currently involved in FlightSafety's portable simulator design project. He holds a Bachelor of Science Degree in Electrical Engineering from the University of Tulsa.

Mr. Mark Easter is a Staff Engineer at FlightSafety International specializing in UNIX® operating systems and simulation software design. He is the lead software engineer in FlightSafety's portable simulator design project. He holds a Bachelor of Science Degree in Broadcast Engineering from Bob Jones University.

OPEN SYSTEMS AND INDUSTRY STANDARDS IN FLIGHT SIMULATION: WHAT DO THEY PROMISE, CAN THEY DELIVER?

Bruce Johnson
Harris Computer Systems Division
Houston, Texas

Michael Caffey
FlightSafety International
Tulsa, Oklahoma

Mark Easter
FlightSafety International
Tulsa, Oklahoma

INTRODUCTION

Though the terms industry standards and open systems are generally associated with computer systems of the nineties, the roots of standardization actually stretch back several decades. Over twenty years ago computer industry standards such as ASCII, FORTRAN and RS-232 enabled engineers and programmers to design and write "semi-portable" applications. In those days it was a fairly simple task to keep track of industry standards due to their meager numbers. Since no-one truly expected code to be portable anyway, the few existing industry standards of that era were often met with relative ambivalence.

Today, however, expectations are quite different. Industry standards and open systems are influencing the way that products are designed and developed in virtually all application areas from embedded flight control to desktop publishing. Engineers and programmers are expected to design and code applications that are portable to a variety of platforms. Flight simulation manufacturers and training system contractors are no exceptions. The end-users of simulators and training devices are not only requesting specific computer systems in their specifications but also are expecting code portability to the next generation of computer hardware. They want choices, they want the combination of hardware and software products to work together, and they want to be able to upgrade part of the system in the future without a major redesign.

With more standards there are more choices. Unfortunately, to make intelligent choices a learning process must also take place. It is dangerous to assume that all standards are good and should be followed. It is equally treacherous to ignore standards altogether. Computer vendors, simulation manufacturers, and training contractors are all beginning to realize that standards compliance is a desired feature of their products.

So, either by choice or by decree, open systems and industry standards are now affecting the way that the simulation and training industry designs and develops their products. In order to keep up with the changing marketplace and the evolving requirements of customers, it is imperative that simulator manufacturers and training contractors learn to maximize the benefits that open industry standards can offer them.

WHAT IS OPEN SYSTEMS ARCHITECTURE AND HOW DOES IT APPLY TO THE DESIGN OF A FLIGHT SIMULATOR OR TRAINING DEVICE?

An open industry standard typically has the following characteristics:

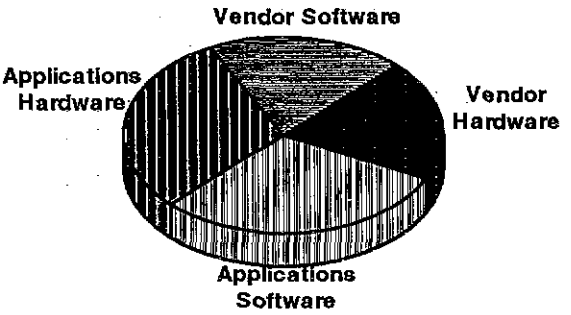
- Platform (Vendor) Neutrality
- Multiple Source Availability
- Wide Industry Acceptance/Demand
- Controlled by an Industry Consortium (Preferred)

The standards described later in this paper and used by FlightSafety in their design effort largely adhere to these criteria. Open industry standards are the building blocks used to produce an open systems architecture.

Open systems architecture, while often associated with simply the use of the UNIX® operating system (OS), actually encompasses much more. It includes elements of both hardware and software; from the Instructor Operator Station (IOS) to the visual system. It stretches from the early stages of design and development, through product delivery and

acceptance, then on into years of support, maintenance, and upgrades.

Figure 1



As illustrated in Figure 1, only a portion of any given simulation or training product is actually purchased as Commercial Off-The-Shelf (COTS) from a vendor. While the OS may be a real-time UNIX variant and the computer hardware itself based on industry standard VME, the majority of the end product will ultimately be provided by the systems integrator (in this case a simulator manufacturer or training contractor). Delivering industry standard simulators is a shared responsibility of both vendors and simulation and training contractors; vendors delivering industry standard systems and contractors using these standards in the development of their simulators and trainers. The true benefits of open systems will not be realized unless both vendors and integrators are committed to delivering industry standards in their products.

The success of the UNIX operating system in virtually all application areas has shown how the push for standardization has affected computer selection. Its success testifies to the industry trend of placing as much emphasis on standardization as is placed on features and/or performance.

Performance, specifically real-time performance, has been the primary reason why the UNIX operating system was slow to be used in simulation and training applications. Today, however, with real-time UNIX operating systems on the market that deliver true deterministic performance, the flight simulation industry is reaping the rewards of designing with open systems. Twenty years ago it was the norm for simulators and trainers to operate off proprietary operating systems (often modified by the simulation manufacturer) with applications written in assembly language. To demonstrate how things have changed, over the past year most of the full flight simulators delivered to commercial airlines have had host computers that operate under a version of the UNIX operating system with the application code devoid of ANY assembly language.

WHICH STANDARDS ARE IMPORTANT TO THE SIMULATION AND TRAINING INDUSTRY

Many publications exist from a variety of sources, that define the existing standards pertaining to the computer industry (for example, from 88open®). For brevity, only the computer industry standards that appear to be getting wide use and/or attention in the flight simulation and training industry will be discussed here.

Operating System Standards

The operating system standards that seem to be having the most widespread affect on the way that code is designed and developed in the simulation and training industry are those from the Institute of Electrical and Electronic Engineers (IEEE) POSIX working groups. POSIX (Portable Operating System Interface for UNIX - the X ties it to UNIX) is a series of standards originally set-up by a technical subcommittee of UniForum (a non-profit UNIX trade association). Since that time, the IEEE and the International Standards Organization (ISO) have taken over the standardization effort in order to give it wider recognition. Much has been said to criticize the POSIX standardization effort because it has all the drawbacks of design by committee - a lengthy, political, and complex process. Yet still the end result will provide programmers with a portable set of system services to use in their code development.

With POSIX, rather than worrying about the inconsistencies between UNIX system variants, a programmer can simply follow the POSIX standard and be assured of code portability. Even some non-UNIX operating systems, such as DEC VAX/VMS™, are committed to compliance with the POSIX standards.

The POSIX standardization effort encompasses a large variety of operating system features including some that are appropriate for only specific application areas such as supercomputing and transaction processing. The following POSIX draft standards are more general in nature and are gaining interest in the flight simulation and training industry:

1003.1	- System Services and C Language Bindings
1003.2	- Shell and Utilities
1003.4	- Real-Time Extensions
1003.4a	- Threads Extension
1003.5	- Ada Bindings to 1003.1
1003.9	- FORTRAN Bindings to 1003.1
1003.20	- Ada Bindings to 1003.4

Table 1

To date, only one of the POSIX standards has completed the standardization process of the IEEE/ISO - 1003.1 (commonly referred to as POSIX.1). Consequently, the other "standards" are actually "draft standards" until they receive formal IEEE/ISO approval. The POSIX.1 standard defines a general operating system interface including system subroutines, error conditions, and return codes. Several independent accreditation labs exist that can certify an operating system with POSIX.1 compliance. In order to be certain that an operating system complies to POSIX.1, independent certification is certainly a good idea. Flight simulation manufacturers that are committed to standardizing their code and writing portable applications should not only select a POSIX.1 compliant OS but also see to it that the standard is utilized in their code development.

Another of the POSIX standardization efforts that has gained the interest of the flight simulation community is POSIX 1003.4 - Real-time Extensions. POSIX.4 defines a set of extensions to the UNIX operating system that adds a great deal of real-time functionality to a "standard" UNIX system. POSIX.4 standardizes the following real-time features:

- Synchronization (Counting Semaphores)
- Clocks and Timers
- Memory Locking
- Memory Mapped Files and Shared Memory
- Priority Scheduling
- Interprocess Message Passing
- Synchronized I/O
- Asynchronous I/O
- Real-Time Files
- Real-Time Signals (Asynchronous Event Notification)

With POSIX.1 and POSIX.4, virtually all of the system-level software of a flight simulator or training device can be designed and coded in a platform independent fashion. As an example, a cyclic executive commonly used in flight simulators could be written with the process primitives of POSIX.1 and the real-time signals and clock interfaces of POSIX.4. Unlike POSIX.1, however, POSIX.4 is still evolving and is not yet an approved standard of the IEEE/ISO. Though several operating system vendors have claimed compliance with draft revisions of the POSIX.4 standard, compliance at this time guarantees little in the way of true portability. Those involved with the POSIX.4 standardization process believe that it will be approved by the IEEE late this year. Within a few months after that, expect OS and computer system vendors to be providing truly POSIX.4 compliant products.

Programming Language Standards

A typical simulator or training device designed today utilizes at least two different higher-order languages. While much of the industry continues to develop applications under FORTRAN, both Ada (mostly for government contracts) and C are seeing widespread use as well. All three of these languages now have industry standards associated with them:

FORTRAN	- ANS x3.9-1978 (FORTRAN 77) often supplemented with MIL-STD-1753
C	- ANS x3.159-1989 (ANSI C)
Ada	- MIL-STD-1815A

Table 2

To ensure portability of source code language standards must be adhered to by all engineers and programmers writing code on any given program. If language extensions are used and porting is necessary, it could require both trivial syntactical changes and major design changes. As an example, the CASE statement

often found in FORTRAN compilers is not part of either ANS x3.9-1978 or MIL-STD-1753. It is, however, not difficult to translate it to a "standard FORTRAN" IF/THEN/ELSE statement. Conversely, other known FORTRAN extensions (BIT data types for example) are not easily translated to standard FORTRAN and could severely limit code portability if used. Certainly a tradeoff exists between features and standardization. Yet if true portability is the priority, then all non-standard, language extensions should be eliminated from source code.

It is interesting to note that all three languages (FORTRAN, C and Ada) are currently undergoing (or have recently undergone) additional standardization efforts that will be providing more capabilities and features to these language in the near future:

FORTRAN	- FORTRAN 90 (ISO/IEC 1539:1991)
Ada	- Ada 9x (MIL-STD-1815B)
C	- ANSI C++ (ANS X3J16 Committee)

Table 3

These standards will lead to many changes in the way that code is designed and developed in the nineties. They, hopefully, will provide standards-based features that limit reliance upon proprietary extensions of language processors.

Networking and Interface Standards

Standard networking and networking protocols have impacted the training and simulation industry in many different ways. The interoperability of computer systems has changed dramatically with the implementation of industry standard networking with products such as Ethernet® and the Transmission Control Protocol/Internet Protocol (TCP/IP).

Industry standard protocols are not only being used in simulation and training development environments but also in the training devices themselves. Because of their low cost and high availability, engineers are findings ways to include Ethernet interfaces into their real-time applications. A real-time Ethernet interface will typically be designed to omit the higher layers of the TCP/IP protocol in order to improve throughput performance. For example, instead of using TCP, the User Datagram Protocol (UDP) might be used. Some computer system vendors even provide a low overhead, "raw" IEEE 802.2/802.3 interface at the TCP/IP link level. While such an interface actually permits data transfers at near the limit of the Ethernet's performance capability (10 megabits/second), the software may not be portable to all Ethernet platforms.

With the price coming down and the availability going up of Fiber Distributed Data Interfaces (FDDI), expect industry standard protocols to appear in real-time interfaces to an even greater extent. There is also quite a bit of work going on to provide FDDI over Shielded Twisted Pair (STP) copper cable in order to lower its implementation costs even further.

The following representative I/O bandwidths are attainable utilizing Ethernet and FDDI with industry standard protocols (in Kilobytes per second - KB/sec):

TCP/IP over Ethernet	750 KB/sec
UDP/IP over Ethernet	1150KB/sec
Raw IEEE 802.3 over Ethernet	1250KB/sec
TCP/IP over FDDI	2000 KB/sec
UDP/IP over FDDI	3500 KB/sec
Raw IEEE 802.3 over FDDI	4500 KB/sec

Table 4

While Ethernet and FDDI can currently deliver quite impressive I/O throughput values, throughput is only a part of what is typically needed in real-time systems. Ethernet and FDDI I/O can be very non-deterministic and provide no guaranteed response times. In addition, Ethernet and FDDI device drivers can often add tremendous latencies to operating system real-time responsiveness.

One of the true needs of the real-time computing industry is for a streamlined, deterministic, standards-based, networking protocol. Some work is being done in this area with the proposed Xpress Transfer Protocol (XTP) from Protocol Engines, Incorporated. The

XTP definition provides the following "real-time oriented" capabilities:

- Message priority and scheduling
- Reliable multicast mechanism
- Real-time reliable datagram capability
- Selective retransmission and acknowledgment

Work is currently being done at several locations to assess the suitability of XTP for tactical simulators such as the U.S. Army's Close Combat Tactical Trainer (CCTT).

Display and Graphics Standards

Display and graphics standards are utilized quite extensively in both development and target environments of flight simulators and training devices. Products that use the X-Window System™ and OSF/Motif™ have added to the interoperability and portability of graphics-oriented, user-interface software. Whether in Computer Based Trainers (CBT), Instructor Operator Stations (IOS), or RADAR/Sensor simulators, standards-based graphics components are appearing more and more.

In the development environment, the X-Window system has permitted graphics to migrate to the desktop level with its availability on workstations, X-terminals, and even PCs. For target environments (i.e., simulators), the graphics standards permit portability of graphics applications to a variety of platforms. This portability leads to a "vendor neutral" approach in the design of all graphics oriented software as well.

I/O Bus Standards

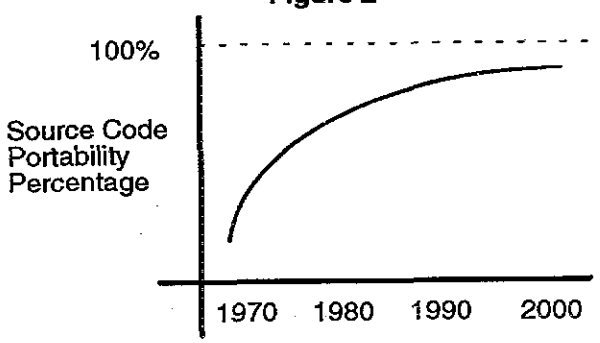
Products such as the VME bus have clearly demonstrated the advantages of designing flight simulators with open architecture. The industry standard VME bus offers thousands of products from literally hundreds of vendors. While certainly other I/O buses are used in simulators and trainers (for example MULTIBUS™ II, Micro Channel™ etc.), none can come even close to offering the product availability of VME. To illustrate, try determining how many different vendors provide FDDI or ARINC 429 interfaces for either Multibus II or Micro Channel and then compared that with the number of vendors that provide VME interfaces. It is safe to say that a engineer is severely limiting his choices for I/O if a VME bus is not included in the host computer system.

New revisions proposed for the VME bus standard (IEEE 1014-1987) will virtually assure its leadership position for the remainder of this decade. These revisions include some extensions that will provide increases to the I/O bandwidth capabilities of VME. These extensions include VME-64 (64 bit data transfers) and Source Synchronous Block Transfer (SSBT). These two extensions should allow a VME bus to practically sustain an I/O bandwidth of nearly 50 megabytes per second.

MAJOR PROMISES OF OPEN SYSTEMS DESIGN

The major promises of open industry standards are portability and interoperability.

Figure 2



The graph in Figure 2 depicts how the evolution of industry standards has affected the portability of source code. The graph illustrates how source code has grown in portability over the years as industry standards have evolved. It also illustrates that while portability is converging on the 100% level, it is uncertain at this time if 100% portability is a goal that will ever be achieved.

In the days when code development was performed on a single monolithic computer system, there was little need for computer system interoperability. Today, however, with virtually every engineer and programmer having a Personal Computer (PC) or workstation on their desk, computer system interoperability is

essential. Networking and file sharing are commonplace in today's distributed architectures and most computer systems can provide seamless integration of utilities and data to/from multiple platforms.

To what extent can these promises be realized in the simulation and training industry? Can the design and development of simulators in a real-world environment truly benefit from standards? The Simulation Systems Division of FlightSafety International, Inc. has been involved in precisely this type of development. Since the inception of their design effort, the goal has been to develop code that is nearly 100% portable across multiple platforms. This development effort has not been without its difficulties and compromises yet it has remained largely successful. Specific details of their development effort are provided in later sections of this paper.

WHAT DIFFERENCE DOES IT MAKE TO THE END-USER?

As an end-user it may be difficult to discern just exactly what the benefits are of purchasing a simulator or training device that maximizes the use of industry standards. Certainly, to the end-user, the following questions are paramount:

- Will a standards-based system be more reliable?
- Will a standards-based simulator be more maintainable?
- Will a standards-based simulator improve the actual training process?

Unfortunately, the true empirical data needed for answering these questions will not be available for quite some time (actual data typically lags behind technology by several years). However, based on what is known today much can be said to predict the answers to these questions.

In general, the use of industry standards in the hardware design of a simulator or training device can be expected to improve reliability based on simple component availability criteria.

A good example of this is in the area of devices designed for an industry standard peripheral bus, such as the Small Computer Systems Interface (SCSI). SCSI, ANSI standard ANS-STD-X3.131, is used by hundreds of different vendors. If a computer system is designed to use 5 1/4 inch disk drives that interface over a SCSI bus, there are at least 30 different manufacturers that provide products to choose from. A computer system manufacturer has the freedom to select the most reliable of the available disk drives to increase the reliability of the complete system. In addition, since the disk drive market is extremely competitive, reliability is a feature that is closely monitored and constantly improved.

Likewise, software reliability is improved by utilizing industry standard operating systems and utilities. A larger installed base for software products leads to the identification and correction of a greater number of problems. Reliability is increased since greater use of the software yields a more stable product. UNIX source code from AT&T is distributed to hundreds of different locations throughout the world which leads to massive amounts of scrutiny and testing of the product. Code stability is quickly obtained and easily maintained, resulting in a more reliable product for the end-user.

The training process itself can also be affected by industry standards. As an example, the IOSs of flight simulators are often cited as an area in need of standardization. IOSs can typically differ from simulator to simulator and a standardized man-machine interface offers many benefits. A standards-based IOS design could begin by specifying the use of some existing industry standards. For example, a graphical-user interface (OSF/Motif) and a host to IOS interface (Ethernet with TCP/IP) could be the beginnings of a standardized IOS. Certainly allowing the instructor to focus more time on the training and less time on operation of the IOS itself will benefit the training process.

Many other areas of training could indirectly benefit from utilizing industry standards in simulator design and development.

OVERVIEW OF THE FLIGHTSAFETY DEVELOPMENT EFFORT

FlightSafety International is currently involved in the development of a portable flight simulator design. This development effort targets an FAA

Phase II Lear 31 simulator that is being built for Singapore International Airlines. For this project, the host computer software is to be 100% portable to any UNIX computer system that complies with a basic set of widely recognized industry standards. A Harris Night Hawk 4400 computer system is being used as the host computer for this project, though other computer systems are used (IBM RS/6000, Sun SPARC™, Concurrent 8000) to help ensure portability to a variety of computer platforms.

Typically a number of independent computer systems are used in flight simulators. The focal point of the simulation is, most often, the host computer system, where all of the aircraft modeling software resides. For most simulators, the host computer software is written in a mixture of C and FORTRAN, though Ada is used on certain simulators.

In simulator development efforts that preceded this project, a number of other computers were used to implement subsystems. Each subsystem generally required very little modification from one simulator to the next. Software for the subsystems was written in C, FORTRAN, and assembly language. The subsystems interface to the host computer system through several different types of interfaces depending on the requirements of the subsystem. Direct Memory Access (DMA) interfaces were used for the subsystems that require high speed real-time data transfers to and from the host computer, while Ethernet and RS-232 were used for the less demanding interfaces. All of the subsystems were VME-based. Table 5 lists the subsystems used in these simulators and the type of interfaces that were used between the subsystem and the host computer.

Subsystem	Interface
Digital Control Loading and Digital Motion	DMA
Cockpit I/O	DMA
Avionics Interface Computer	DMA
Instructor Operation Station	Ethernet
External Voice Processor	RS-232
Visual	(Changed from simulator to simulator.)

Table 5

Base Standards

At the outset of this project, it was decided to specify a set of standards that all simulation software would adhere to whenever possible. Extensive investigations were performed into current computer standards and a base set of standards was selected that a computer system must comply with in order to be used. All of these standards are widely accepted and well controlled. The standards address the operating system interface, programming languages, networking, user interface, and computer hardware standards. All of the host computer system software is designed to comply with these standards. Table 6 lists the base standards that were selected.

Operating System Interface:
- POSIX 1003.1
- AT&T System V Common Object File Format (COFF)

Programming Languages:
- ANSI C (Required for ALL simulators)
- FORTRAN 77 (Required for all FORTRAN-based simulators)
- MIL-STD 1815A Ada (Required for all Ada-based simulators)

Networking:
- EEE 802.3 Ethernet
- TCP/IP
- Sun NFS and NIS

User Interface:
- X Windows Version 11 Release 4
- MOTIF
- Curses
- Korn Shell

Hardware:
- VME I/O Subsystem
- IEEE Floating Point

Table 6

Operating System Standards

In this project's host computer system there are two broad classes of software: systems software and aircraft modeling software. The aircraft modeling software performs all operations that are associated with simulating a particular

type of aircraft. This software is written in FORTRAN 77 and is modified for each type of aircraft that is to be modeled. The aircraft modeling software generally requires no direct interaction with the operating system, thus by using standard FORTRAN 77 the aircraft modeling software for a particular type of aircraft can be easily ported to various types of computer systems with almost no modifications.

The systems software provides support for the aircraft modeling software. The real-time executive, the I/O processing software, and the general purpose library routines are examples of the systems software. The systems software is written in ANSI C and is generally unmodified from one simulator to the next. This software is very dependent on operating system services.

In order to maximize the portability of the systems software, POSIX 1003.1 was selected as the base standard for interfacing to the operating system. Whenever possible, POSIX 1003.1 system services and library routines are used to interface to the operating system. If an essential capability is not provided in the POSIX 1003.1 standard, a hierarchy of standards is followed to determine what system service is to be used to provide the capability. The standards hierarchy is defined as follows:

1. POSIX 1003.1
2. AT&T System V Release 4
3. OSF1
4. BSD 4.3
5. Vendor-Specific

It is important to realize that just developing software on a standards-based computer system does not guarantee that the software will be easily portable to other computer systems. Software must be designed with portability in mind. Virtually every vendor's implementation of UNIX contains features from several different versions of UNIX, as well as, some completely proprietary features. For example, almost every system that is based on AT&T System V Release 4 (SVR4) also contains numerous enhancements from the Berkeley Standard Distribution (BSD) implementation of UNIX. Unfortunately, the specific set of BSD enhancements that are included in an implementation of UNIX can vary greatly from one vendor to the next. It was found it to be particularly helpful to have several different UNIX systems available for software development. Three different UNIX systems have been used during this particular software development effort. Use of these systems has helped to identify many key portability issues.

There are certain tradeoffs associated with developing portable software. There is sometimes significantly more effort required to develop a portable solution to a problem than a non-portable solution. For example, many real-time computer companies offer proprietary ways of scheduling real-time software. These proprietary features reduce process dispatch latencies (PDL) and simplify the task of developing a real-time executive. However, use of such features reduces portability. Instead of using these features this project elected to develop a portable simulation executive that used standard UNIX interprocess communication (IPC) mechanisms. Use of this approach increased the development effort required and slightly increased the amount of processing power required to implement the executive. However, the resulting executive is now almost 100% portable to any UNIX platform. It should be noted that this does not mean that vendor specific enhancements are useless in a UNIX environment. At higher iteration rates it would have been necessary to use proprietary features.

A significant part of a simulator's systems software is associated with I/O interfaces. As shown in Table 5, there are several types of I/O interfaces from the host computer to other computers comprising the simulator. It is fairly easy to achieve portability in some of these interfaces. For example, the Ethernet interface to the IOS is implemented using TCP/IP and is very easily ported to different UNIX systems. Likewise the RS-232 interface was implemented using the POSIX "termio" interface that is supported on all POSIX compliant systems. However, for the real-time interfaces that require high levels of throughput and fast interrupt response times, portability is much more difficult to achieve. For instance, DMA interfaces might be implemented using a VME card that emulates a DEC DR11W interface. Then typically several blocks of data would need to be sent over each of these interfaces at 30 Hz. The interface protocol used between the host computer and the subsystems also requires the host computer

to handle interrupts quickly in order to implement these interfaces. To accomplish all of this, kernel resident device drivers must be written to implement the communications protocol used for each of these interfaces. Standardizing on the use of a particular VME interface card simplifies the process of porting these device drivers to other UNIX systems, however there is still a significant amount of porting effort required in this area due to differences in the way device drivers are implemented on various UNIX systems.

The DDI/DKI (Device Driver Interface/Device Kernel Interface) standard developed by AT&T is the best attempt yet to standardize UNIX device drivers. While the Harris Night Hawk supports the DDI/DKI standard, many other computer systems currently do not. Until more systems support DDI/DKI, relatively little portability can be achieved in device driver software. This is the only portion of this project's software that is not expected to be easily portable to other UNIX platforms.

One other standard that has been found to be very important is the Common Object File Format (COFF) standard. This standard addresses the format of the object files produced by the compilers on UNIX systems. UNIX systems that support COFF also contain a library of C functions that can be used for extracting information from the COFF files. The significance of COFF to real-time programmers is that COFF format object files contain all of the symbolic information required to determine the addresses of data contained in the executable programs. Using COFF, it is possible to develop a portable real-time debugger. Note that the library routines isolate your application software from the actual object file format. Thus a vendor's object file format can differ from the actual COFF format without affecting your software if the vendor preserves the library routine interface.

Programming Language Standards

ANSI C, FORTRAN 77, and MIL-STD 1815A Ada have been selected as language standards. The important thing to remember when dealing with these languages is that a vendor may add vendor specific enhancements to these languages. To ensure portability, constantly check to be sure that a language feature in use is defined by the language standard. With all of the enhancements that are added to these languages, it is easy to get vendor specific enhancements scattered throughout the code.

Networking Standards

An essential part of this project's software development environment is its Local Area Network (LAN). To support this, IEEE 802.3 Ethernet has been selected as the primary networking medium. In addition, All potential computer systems are required to support TCP/IP and Sun's Network File System (NFS) and Network Information System (NIS). It should be noted that while almost every UNIX computer supports some version of NFS and NIS, there are periodic updates to these packages and not every vendors' support of NFS or NIS will be up to date with the latest versions of these products from Sun. Thus, it is important to verify that potential computer systems support the NFS and NIS features that you use in your LAN.

User Interface Standards

To support graphics, the X-Windows System Version 11, Release 4 (X11R4) and the OSF/MOTIF Graphical User Interface (GUI) have been selected. These are by far the most widely supported graphics interfaces on UNIX computer systems. For text based applications, the UNIX "curses" interface is used, which hides terminal specific processing from the application code promoting portability.

The Korn Shell has been selected as the standard shell. The Korn Shell was selected for a couple of reasons. It is a superset of the Bourne Shell which is available on all UNIX systems. This allows Bourne Shell scripts to run under the Korn Shell with no modifications. In addition, the Korn Shell provides a user interface that is superior to both the Bourne Shell and the C Shell.

Hardware Standards

For the most part, this project avoided specifying particular standards for computer hardware. The only standards specified were that any potential computer system must support a VME bus and must also use the IEEE floating point format. Compliance with these two standards greatly reduces the effort required to

integrate the various computer systems that comprise the simulator. FDDI is being evaluated as a possible future replacement for the DMA interfaces in order to further increase application portability.

CONCLUSION

Though computer industry standards have been around for quite some time, their influence on computer system selection and hardware and software design and development has never been greater than what it is today. Genuine benefits can be gained for simulator manufacturers, training contractors, and end-users by specifying and designing to open-system architecture and computing. Based on what has been demonstrated on the Lear 31 simulator project, true portable simulator applications are now possible. In a computer system, standardization has become just as important as both performance and functionality.

Where is this leading the industry? For computer manufacturers, industry standards must be given a high priority in product development. Simulator manufacturers and training contractors must be able to provide products that enable the end-users to upgrade and change-out components without a major redesign. Already, in many cases, acquisition of a flight simulator involves selecting from a list of supported host computers. This end-user "menu-like" selection process will most likely evolve to other components of a simulator as industry standards continue to affect simulator design.

Industry standards and open architecture are more than the latest buzzwords. Computer system manufacturers, simulator manufacturers, training contractors, and end-users that can learn how to maximize the benefits of industry standards will greatly profit over the next decade and on into the next century.

Trademark Acknowledgment:
UNIX is registered trademark of UNIX System Laboratories, Inc.

Ethernet is a registered trademark of Xerox Corporation

OSF/Motif is a trademark of the Open Software Foundation, Inc.

X-Window System is a trademark of MIT

RS/6000 and Micro Channel are trademarks of International Business Machines Corporation

NFS and SPARC are trademarks of Sun Microsystems, Inc.

MULTIBUS is a trademark of Intel Corporation

88open is a registered trademark of 88open Consortium, Ltd.

VAX/VMS is a trademark of Digital Equipment Corp.

References:

1. Parkinson, D. "New Computing Standards for Flight Simulator (Simulator Open Systems)", IATA Flight Simulation Symposium, November, 1990
2. "Draft Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 1; System Application Program Interface (API) - Amendment 1: Real-time Extension (C Language)", Institute of Electrical and Electronics Engineers, Inc., 1992
3. Johnson, B. "How Computer Industry Standards are Affecting the Design and Development of Modern Day Flight Simulators" International Training Equipment Conference, April, 1992
4. "88open World of Standard Reference Guide", 88open Consortium, Ltd., 1992