Open Systems Architecture in a COTS environment*

Alan R. Stottlemyer
Kevin M. Hassett

ABSTRACT

A distributed architecture framework has been developed for NASA at Goddard Space Flight Center (GSFC) as the basis for developing an extended series of space mission support data systems. The architecture is designed to include both mission development and operations. It specifically addresses the problems of standardizing a framework for which commercial off-the-shelf (COTS) applications and infrastructure are expected to provide most of the components of the systems. The resulting distributed architecture is developed based on a combination of a layered architecture, and carefully selected open standards. The layering provides the needed flexibility in mission design to support the wide variability of mission requirements. The standards are selected to address the most important interfaces, while not over constraining the implementation options.

KEYWORDS

System architecture, Standards, Distributed systems

INTRODUCTION

NASA/GSFC is a primary developer of space mission systems for NASA-supported unmanned missions with Earth orbiting spacecraft. There has been an increasing need to reduce the time for system development, and the overall life-cycle cost of such missions, by as much as an order of magnitude. To achieve these gains, an entirely new approach was needed. As a part of developing the new approach, a new system architecture was required which would enable these gains.

The architecture must enable maximum re-use for system components, and enhanced ability to integrate such components effectively and efficiently. Because the

number of such space missions is relatively low, component re-use only within NASA missions would not achieve the cost reductions needed, and a high level of use of commercial software and hardware is required. Further, the architecture must maximize the available choices and avoid dependence on any single vendor for critical components.

Existing standards for integration have addressed infrastructure levels, and provided little definition for application level interfaces and functionality. Past experience demonstrates that standardization at the application level frequently had the effect of severely limiting flexibility and use of COTS components because the standards were not widely supported. The Open Systems standards need to allow greater freedom to choose computer platforms, information repository and application products, and commercial mechanisms supporting application interconnection and data transport.

Over the last year and a half, the architecture has evolved rapidly, and been validated through an associated prototyping program. The resulting architecture is presented in the following sections, with a complete documentation in reference 1.

THE LAYERED ARCHITECTURE APPROACH

The system architecture combines three primary features. First, a layered design is used which enhances flexibility by information hiding. Second, the software client-server concept is preserved as a characteristic of a specific component interaction, rather than of the component as a whole. Third, the hardware architecture is based on both three-tier client-server and peer-to-peer configurations.

The layered design is shown in Figure 1. The design is based on a combination of the Open System Interconnect (OSI) communications reference model and the National Institute for Standards and Technology (NIST) Open System Environment (OSE) reference model. The OSI model is used because of the distributed system architecture. The OSE model provides a good foundation for mapping the structure within a computer platform. The combination is necessary to provide the overall structure, with some elements mapped to specific OSI layers and others to OSE layers.

On the left side of the diagram are shown the OSI reference layers (in parentheses), and the architecture elements mapped to them. On the right, the OSE model of platform environment and platform services is followed. The “network transparency” element serves to integrate the two into a common application program interface (API).

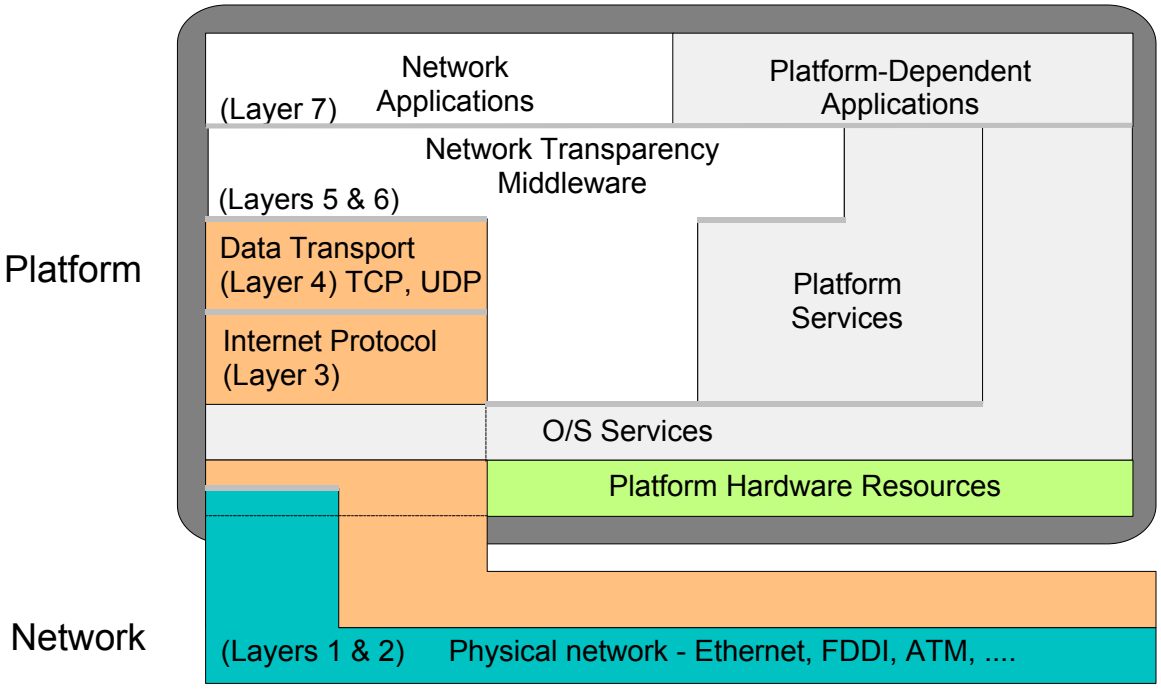

Figure 1. Layered Architecture

The network transparency layer provides a common, network independent, environment for all applications which use that interface. A set of common global services are provided through this interface to support application management, intercommunication, and data transport. The general support services are modeled after those supplied by OMG's CORBA. However, not all services are required for a given system, and a CORBA implementation is not required.

Some applications will be constructed to use platform specific interfaces. In some cases this is done to enhance application technical performance. In other cases it results from either market pressures, or simply the lack of an interface to network standards. Such applications are supported by a standard interface to platform services, such as defined by the NIST Application Portability Profile (reference 2), X/Open UNIX, or Microsoft Windows WIN32 API.

The software application relationships are client-server for a given interaction. However, a software component can support both client and server interactions, and simultaneously when multithreaded. Some components will be constructed as pure servers or clients, but others will support both relationships. In general, both simple client-server (C/S) and three-tier application designs are expected. Additionally, some applications in our domain

are better designed as data driven pipe and filter (P&F) style. Figure 2 illustrates these types of application relationships.

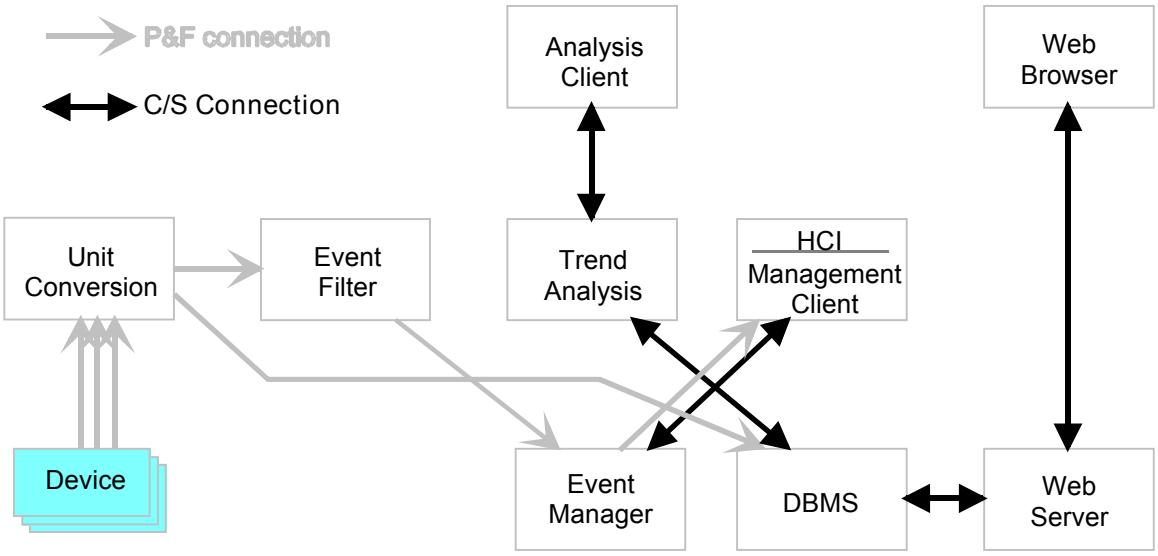

Figure 2. Application Relationship Styles

The hardware architecture combines client-server models with peer to peer. Some components, e.g., network servers, database servers and print servers, will be server platforms in larger systems, but may be combined with other allocated functions in standard workstations for smaller systems. Similarly, there will be application-algorithm processing platforms in larger systems, built to enhance the capability for massive computations required, such as image processing from earth sensing instruments. There are likely to be true client platforms, in the form of either individual PCs or X-terminals, for all systems. However, for many smaller systems, there will be a common workstation platform for all except remote clients. In this latter case, the peer to peer network may work better.

THE APPROACH TO STANDARDS USAGE

Standards selection is both critical to success, and a potential obstacle. Standards are needed to support changes in system configurations, but use of poorly supported standards may, in fact, preclude access to COTS components. The approach to establishing standards was to establish a very small core of widely supported, mandatory standards for global compatibility. The mandatory standards are supplemented by a more extensive set from which to select for a given system implementation. This provides compatibility within

a mission system, and also the flexibility to support the differing requirements between mission systems.

The core standards include the Internet Protocol (IP) suite for network support, and both UNIX and Windows NT for platform support. The overall intent was to keep the number of mandatory standards limited to those with truly global support, and to select only those for which there was a clear benefit to establishing a standard. The flexibility within the set has been given at least preliminary trial in the prototyping environment. A complete discussion of our standards approach is provided in references (3) and (4).

IMPLEMENTATION CONCEPT

The supporting mission system implementation concept is that of configuration by selection from available capabilities, and not one of providing a large, universal capability set for all systems. This is driven by the wide range of mission requirements. Large mission systems, such as those supporting the Mission to Planet Earth, are expected to manage up to a Terabyte of new data per day. Additionally they may require close coordination of activity on multiple spacecraft. By contrast, a small mission system may be acquiring about 100 Megabytes per day, and supporting only a single spacecraft. In another dimension, some mission systems must support distributed near real-time operations, while others have all such needs constrained to localized hardware components.

The resulting designs can range from a large network of high performance platforms distributed around the world, to a few PC platforms attached to a local radio frequency subsystem with Web connections for remote access. By using the capability selection model, the low end systems can be built with low cost COTS components to match their needs, and, where appropriate, have common components with the largest of systems.

Initial prototypes have provided validation of some areas of the approach. Implementations using COTS products with message and file servers for network transparency have been successful. A small set of prototypes using a CORBA integration mechanism have been performed, linking both COTS and legacy applications. The CORBA integration has proven successful in both function and cost. Presentation of these results in full is provided in reference 5.

Avoiding over-specification has also proven valuable. COTS products not identified specifically with space applications have been productively included into the prototyping environment. Analysis support and data manipulation packages have proven useful in the

same way that spreadsheets can be used for many applications. Products covering non-UNIX platforms have been identified in the prototyping effort, validating this approach to low cost mission system development.

CONCLUSION

Use of this architecture for constructing mission systems provides clear benefits. Specification of an infrastructure based on IP, and on UNIX and Windows NT platforms, provides a common foundation for which there is a rich set of available commercial products. Use of the layered approach, and the network transparency component, has led to lower cost integration approaches, based on commercial transparency support, and with direct application integration. Standards for network transparency support, e.g., DCE, CORBA and Java run-time services, are proving to be usable with real commercial support, and with available applications using the interfaces.

The initial prototyping experience has been very successful, indicating the viability of the architecture. However, much work remains before we understand the full implications of the approach. More prototypes and a pilot project are planned to enhance our knowledge. Identification of more effective products and those to cover unusual needs will occur in this process. Tuning of both product selection and the implementation process will be needed to reliably achieve the cost and schedule goals.

ACKNOWLEDGMENTS

Support by NASA under contracts NAS5-31500 and NAS5-31000 is gratefully acknowledged. In particular, collaboration with Gary Meyers and Robert Connerton of the Renaissance team at GSFC during the entire architecture development activity was particularly valuable. Discussion and comments from my coworkers at CSC was essential to success. Particularly valuable assistance was received from my co-author, Kevin Hassett, and from team members Keith Hogie and Edward Stokes.

REFERENCES

1. Renaissance Generic Architecture, 504-REN-96/003, Mission Operations and Data Systems Directorate, NASA/GSFC, May 1996

2. Application Portability Profile, OSE/1 Version 2.0, NIST Special Publication 500-210, 1993

3. Stottlemyer, Alan, and Hassett, Kevin, “The Role of Standards in COTS Integration Projects,” Proceedings of 32nd Annual International Telemetering Conference, San Diego, California, October, 1996.

4. Renaissance Standards, 504-REN-96/005, Mission Operations and Data Systems Directorate, NASA/GSFC, May 1996

5. Scheidker, Eric, Rashkin, Robert, Pendley, Rex, Werking, Roger, Bonnin, Magda, Copella, John, and Bracken, Mike, “IMACCS: A progress report of NASA/GSFC’s COTS-based ground support systems, and extensions into new domains,” Proceedings of 32nd Annual International Telemetering Conference, San Diego, California, October, 1996.