ABSTRACT

NEXT GENERATION SATELLITE ORBITAL CONTROL SYSTEM

by
Thomas Gerard Nowak

Selection of the correct software architecture is vital for building successful software-intensive systems. Its realization requires important decisions about the organization of the system and by and large permits or prevents a system's acceptance and quality attributes such as performance and reliability. The correct architecture is essential for program success while the wrong one is a formula for disaster.

In this investigation, potential software architectures for the Next Generation Satellite Orbital Control System (NG-SOCS) are developed from compiled system specifications and a review of existing technologies. From the developed architectures, the recommended architecture is selected based on real-world considerations that face corporations today, including maximizing code reuse, mitigation of project risks and the alignment of the solution with business objectives.

NEXT GENERATION SATELLITE ORBITAL CONTROL SYSTEM

by
Thomas Gerard Nowak

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer Science

January 2005

APPROVAL PAGE

NEXT GENERATION SATELLITE ORBITAL CONTROL SYSTEM

Thomas Gerard Nowak

Dr. Ali Mili, Thesis Advisor — Date
Professor of Computer Science, NJIT

Dr. Alexander Thomasian, Committee Member — Date
Professor of Computer and Information Science, NJIT

Dr. Robert M. Klashner, Committee Member — Date
Assistant Professor of Information Systems, NJIT

BIOGRAPHICAL SKETCH

Author: Thomas Gerard Nowak

Degree: Master of Science

Date: January 2005

Undergraduate and Graduate Education:

- Master of Science, Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2005

- Bachelor of Science, Electrical Engineering,
  Manhattan College, Riverdale, New York, 1985

Major: Computer Science

Patents:

- US patent # 6,154,15. General Frame-Based Compression Method.

- US patent # 6,597,892. Automated ground system with telemetry initiated command assistance.

To my wife, Kate and children, T.J., Chad and Kevin

ACKNOWLEDGMENT

I would like to express my sincere appreciation to Dr. Ali Mili, who not only served as my research supervisor, providing valuable insight, resources, and intuition, but also gave me support, encouragement, and reassurance. Special thanks are given to Dr. Alexander Thomasian and Dr. Robert M. Klashner for their astute recommendations and active participation in my committee.

Many of my fellow associates at SES-Americom and SES-Astra are deserving of recognition for their support, specifically Phil Schramm, Pascal Wauthier and the entire Computer Systems Engineering group.

I also wish to thank my wife, Kate, for her continued love, support and encouragement over the years.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND DEFINITIONS

AGI	Corporation that markets the STK product
DEC	Digital Equipment Corporation
Epoch	Specific point in time
FLD	Flight Dynamics Software
Fleet	Collection of geosynchronous satellites
FTP	File transfer Protocol
GUI	Graphical User Interface
ISI	Integral System Corp. Markets satellite control software
JDK	Java Development Kit
NG-SOCS	Next Generation Satellite Orbital Control Software
Object	An entity that has state, attributes and services
OOP	Object Oriented Programming
OS	Operating System
PC	Personal Computer
PC-SOCS	PC Satellite Orbit Control System
SES	Company full name is Societe Europeenne des Satellites
SES-Americom	One of the divisions of SES, also known as Americom
SES-Astra	One of the divisions of SES, also known as Astra
SRS	Software Requirements Specification
STK	Satellite Tool Kit, application produced by AGI Corp.
TCO	Total Cost of Ownership
UML	Unified Modeling Language

VAX	Virtual Address Extension. A computer platform
VMS	OS that runs on VAX and DEC Alpha computers
Servlet	A small program that runs on a server

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of this dissertation is to determine the most appropriate architecture for the Next Generation Satellite Orbital Control System (NG-SOCS) that will be jointly developed SES-Americom and SES-Astra.

The NG-SOCS system will be based upon two legacy systems presently in use, Flight Dynamics System (FLD) and PC Satellite Orbital Control System (PC-SOCS). This effort included the compilation of system requirements, an investigation into existing technologies, investigation of merger strategies, development of approaches to deal with heterogeneous platforms and different programming languages, proposing candidate architectures and the plan to ensure that the proposed solution meets the system requirements.

The proposed architectures were not restricted to the use of existing legacy hardware architecture and were free to consider fundamental changes to the existing system architecture. In particular, the new system must allow for the capability of stand alone operation of core system functionality from computers (mostly laptops) which are frequently not connected to the corporation's computer network, yet support a centralized paradigm in order to ensure that all critical satellite information is available to the entire departmental personnel.

In addition, the possible use of third party software to satisfy portions of the system requirements were examined. Presently, two vendors claim to provide

off-the-shelf software systems for satellite orbital control. It was ensured that the proposed solutions support mechanisms that would allow for the potential use of portions of these products sometime in the future.

1.2 Background Information

In general, a satellite orbital control system is used to determine the precise orbit of a satellite, plan for orbit correction maneuvers and provide support tools as is depicted in Figure 1.1. The PC-SOCS and FLD systems are in use at SES due to the merger of SES-Astra and GE-Americom in 2002. The two systems presently employ different architectures, programming languages and computing platforms to accomplish these tasks.

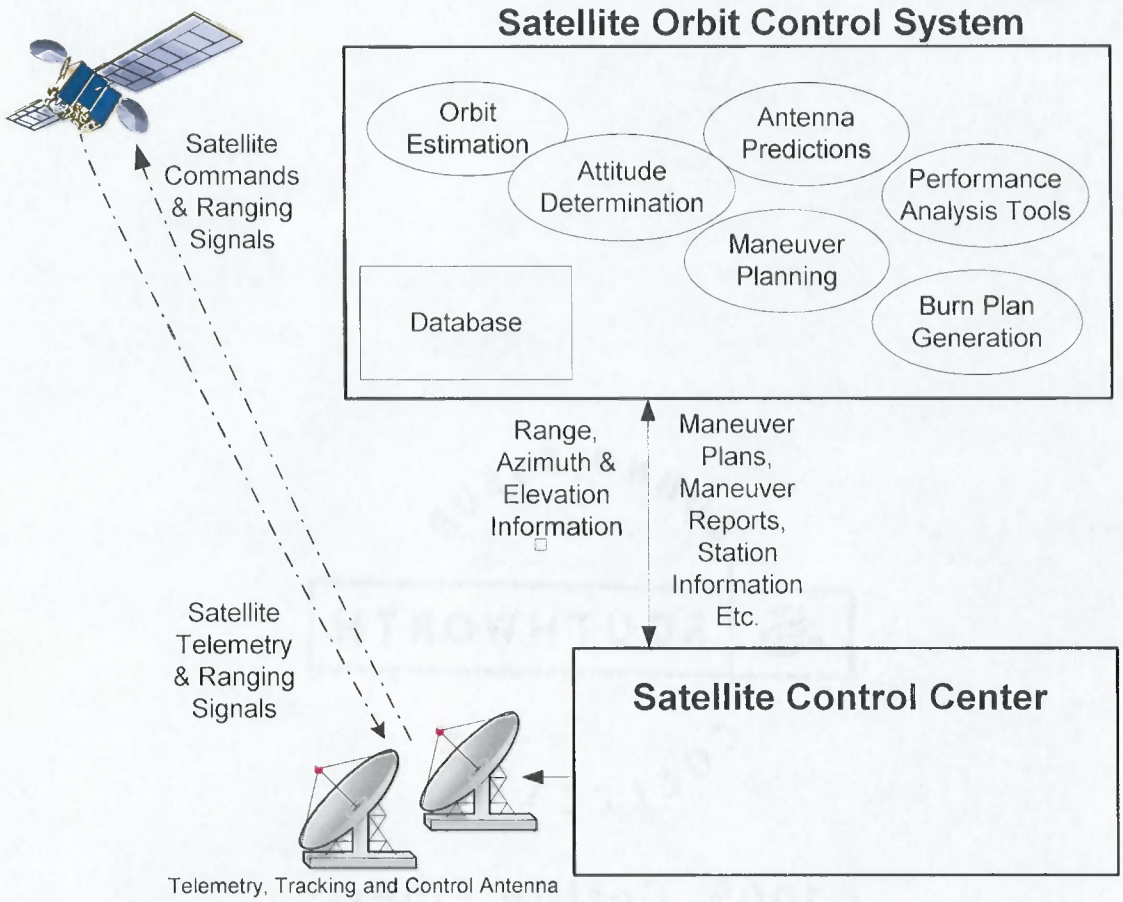

Figure 1.1 Generic Satellite Orbit Control System.

FLD is composed of various modules and input/output files. The modules are written in FORTRAN 90 and ADA mainly due to the various legacy concerns at the time of development of FLD. Additional FORTRAN modules were devised as part of a SES-Astra relationship with a German aeronautics company in the early 1990s and represent many man-hours of mathematical algorithm development and verification. ADA modules were used to tie the system together in a near real-time architecture.

PC-SOCS is composed of C and assembly language modules that were

initially developed by GE-Americom and Integral Systems (ISI) to visually determine and depict the satellite orbits. The choice of architecture was due primarily to the fact that ISI had a government contract to provide a similar system at the time.

CHAPTER 2

REQUIREMENTS COMPILATION AND DEVELOPMENT

The initial steps of this effort focused on the collection and documentation of the NG-SOCS system requirements. Existing FLD and PC-SOCS specification were obtained and disseminated to the NG-SOCS stakeholders. A series of meetings were then held with the different stakeholders to develop and shape the requirements. From the original specifications and stakeholder inputs, the system requirements were finalized. This specification is contained within the Appendix of this document.

The specification followed the recommend software requirements documentation practices of IEEE Std. 830-1984. Stakeholders included the original system developers, business managers and the end users of the system (satellite controllers, satellite analysts, earth station managers and satellite engineers).

Obtaining the final set of requirements was a difficult process for many of the reasons that are identified by Sommerville (124-125). Some stakeholders were sometimes not sure what they needed from the new system or found it difficult to articulate what they required. Specifically, this was the case when discussing the user interface and graphical representation of much of the data. In some cases, it was agreed that the best course of action would be to generalize a requirement and then agree to use a rapid software prototype development

approach to elicit end user feedback at the appropriate time, later in the development cycle.

Also, as acknowledged by Markus (430), end users will resist a new system that is envisioned to be "non-user friendly". This could explain the difficulties that were experienced while trying to obtain agreement from the stakeholders on the manners in which end-users will interact with the new system.

In addition, conflicting requirements from different groups had to be overcome. For example, the centralized system approach, where the main application resides on the server vs. the distributed model where users can run the full featured application when not connected to the central server. In this case, architectures that support both requirements had to be developed.

As discussed in Zwass (550-558), the development of high quality, reliable and accepted systems hinges, to a large degree, on the proper elicitation of stakeholder requirements. For this effort, numerous passionate exchanges of viewpoints provided a foreshadowing of poor end-user acceptance levels if the developed system did not address the concerns that were raised in these meetings.

CHAPTER 3

EXISTING ARCHITECTURES

3.1 Flight Dynamics System (FLD) Architecture

FLD, depicted in Figure 3.1, is composed of various modules and input/output files. The modules are written in FORTRAN 90 and ADA primarily due to the various legacy concerns at the time of development of FLD. FORTRAN modules were developed as part of a SES-Astra relationship with a German aeronautics company in the early 1990s and represent many man-hours of mathematical algorithm development and verification. ADA modules were used to tie the system together in a near real-time architecture.

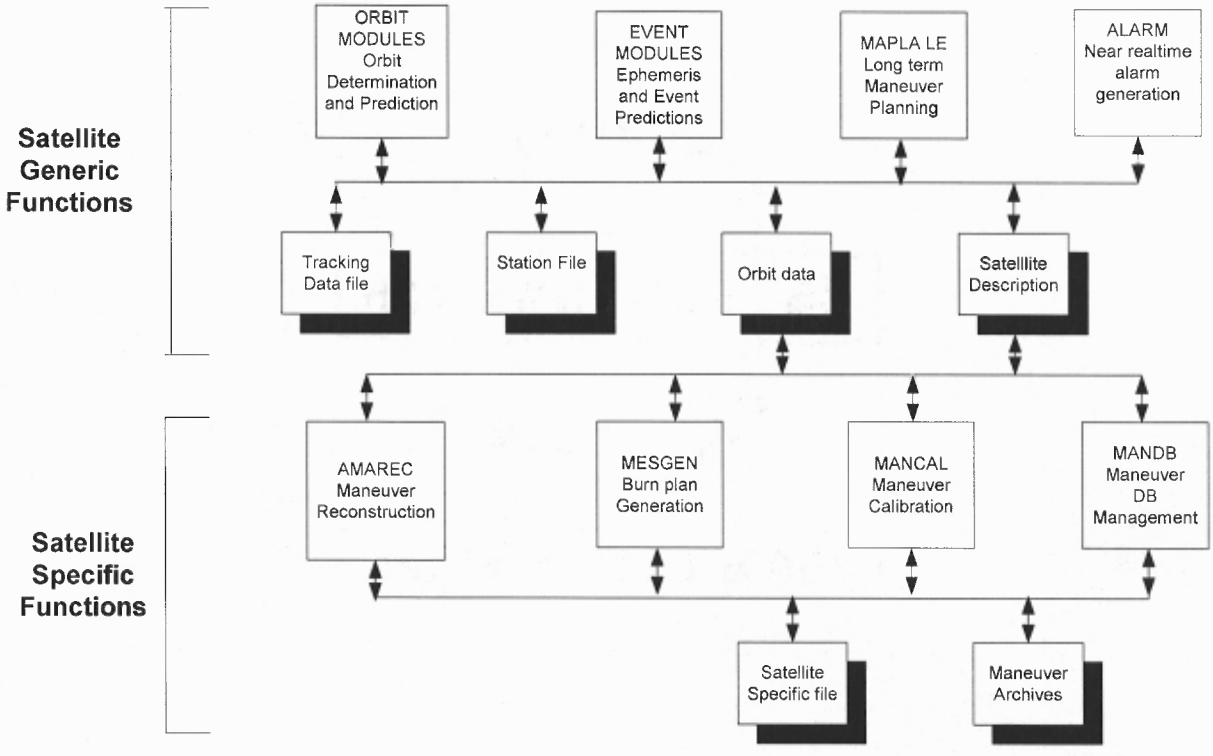

Figure 3.1 Flight Dynamics System (FLD) Architecture.

3.2 PC-Satellite Orbital Control System (PC-SOCS) Architecture

PC-SOCS, depicted in Figure 3.2, is composed of C and assembly language modules that were initially developed by GE-Americom and Integral Systems to visually determine, depict and control the satellite orbits. The choice of architecture was due primarily to the fact that Integral Systems had a government contract to provide a similar system at the time.

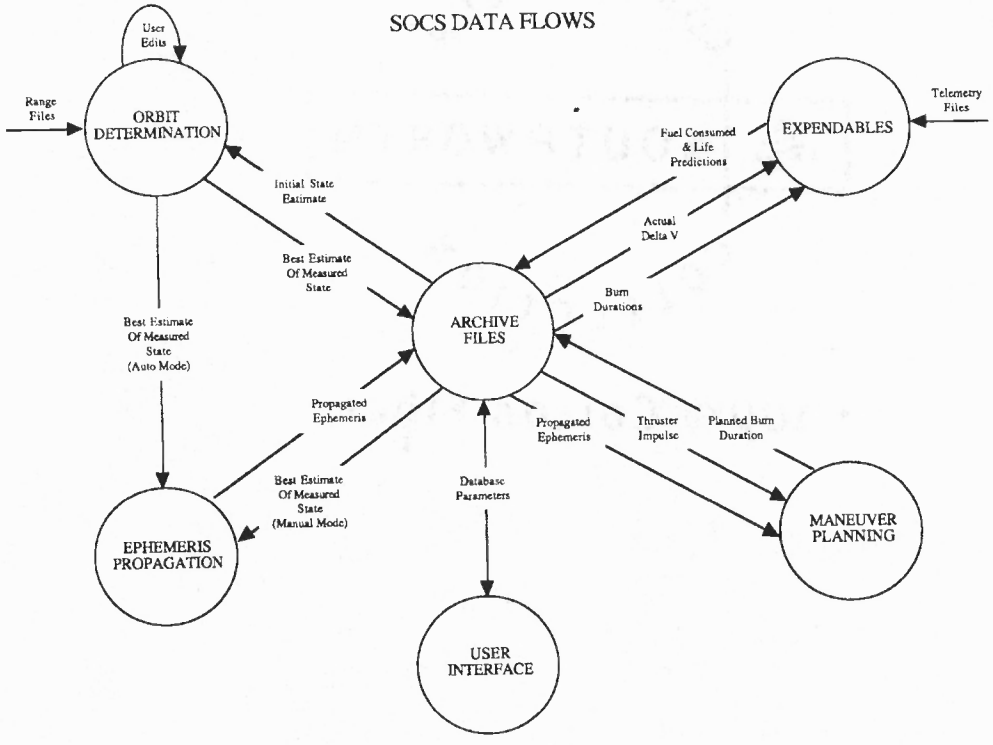

Figure 3.2 PC-Satellite Orbital Control System (PC-SOCS) Architecture.

CHAPTER 4

TECHNOLOGY INVESTIGATION

This chapter documents a study of the state of the industry that was completed in order to examine the existing tools, platforms and operating systems that were under consideration for use in NG-SOCS. The data contained in this chapter was used to support the architecture studies described later in this document.

4.1 Programming Languages

4.1.1 ADA Programming

Ada is a block-structured programming language originally designed for the US Dept. of Defense. As part of the initial concept, Ada designers emphasized software reliability and program safety. It was designed to contain several object-oriented programming features and intended to support large-scale programming efforts. Some of Ada's more powerful features include: nested procedures, nested packages, strong typing, multi-tasking, generics, exception handling, and abstract data types.

To support program safety and reliability, Ada was designed as a strongly typed language requiring that all data elements be declared as a particular type or subtype. Type enforcement is strictly enforced both within and between modules. Ada supports comprehensive exception handling and full complement of sequential control structures. A strict validation of all Ada compilers is adhered

to by use of an official test suite. Ada is strictly standardized and well documented as a language.

Ada was first standardized in 1983, with updates published in 1995. These two versions are generally referred to as Ada *83* and *Ada 95*.

4.1.2 C Programming

C is a high-level block structured language with good support for system programming. Originally designed as a systems programming language, it has proven to be a powerful and flexible language that can be used for a variety of applications. The first major program written in C was the UNIX operating system and for many years C has been tightly linked to the UNIX operating system. However, C has become an important language independent of UNIX, and is widely used in PC, Mac, mainframe and other computing environments.

C was extremely popular in academic and industrial computing from the late 1970s through the early 1990s, and still has a very large user community. Free and commercial C implementations are easily obtained; one of the most popular free implementations is the GNU C Compiler (gcc). C was standardized in 1990; currently the ANSI ISO/IEC 9899 defines the C language

While it is a high-level language, C is closer to assembly language than most other high-level languages. This closeness to the underlying machine language allows C programmers to write very efficient code. The low-level nature of C, however, can make the language difficult to use for some types of applications.

4.1.3 C++ Programming

The C++ programming language is derived from C and is a fairly complicated object-oriented language that is a popular language for application development on UNIX systems and PCs. The syntax of C++ is similar to that of C and contains a mixture of extensions and extra keywords needed to support OO features.

C++ offers a wide range of OOP features like multiple inheritance, strong typing, dynamic memory management, templates, polymorphism, exception handling and overloading. Some newer C++ systems also offer run-time type identification and separate namespaces. C++ also supports the expected features of an application programming language: a variety of data types including strings, arrays and structures, full I/O facilities, data pointers and type conversion. The C++ Standard Template Library (STL) provides a set of collection and abstract data type facilities.

C++ has dynamic memory allocation, but does not have Java like garbage collection. This allows developers to mis-use and potentially leak memory. C++ also supports hazardous low-level facilities like raw memory pointers and pointer arithmetic, which are useful when writing tight code, but can increase the time needed for software development and testing.

Some free C++ compilers are widely available, the most significant of which is the GNU C/C++ compiler, GCC. C++ was standardized by the ISO and ANSI in November 1997.

4.1.4 C# Programming

C# is an object-oriented language derived from C, with some similarities to C++, Java and Visual Basic. C# supports single inheritance, overloading, overriding, reflection, and polymorphism. Developed by Microsoft, the C# object model is designed to correspond directly with Microsoft's COM/DCOM object mode because it was the intent of the language to mainly support development on Windows operating systems.

Microsoft claims that C# offers the power and richness of C++ with the productivity of Visual Basic. C# does provide garbage collection and automatic memory management. GUI events and other external triggers are handled by a novel and complex form of delegation. Attributes support communication from the programmer to the complier and runtime environment via meta-data about code. Attributes are much richer than any similar capability in comparable languages yet also add a very complex side to the language if used by the programmer.

C# is strongly typed and provides extensive compile-time and run-time checking. Unlike Java and C++, C# hides some of the distinction between primitive types and object types by automatically 'boxing' and 'unboxing' primitive as objects. C# supports C-style unsafe pointers, but only within designate code sections. Like C, C# offers the expected complement of primitive types: integers, reals, booleans, and characters while supporting objects and arrays. Similar to Java, C# supports a string type that is an object and tightly integrated with the entire language.

An investigation into the availability of a C# complier determined that the only compiler which fully supports the C# language is provided by Microsoft via the Microsoft .net compilers. Microsoft does make a command-line compiler available for free, under the name of the ".Net Framework SDK".

4.1.5 FORTRAN Programming

FORTRAN was developed in the late 1950s and is still in use, especially in applications geared towards engineering and mathematics. Numerous FORTRAN versions have been released over the years that have added more modern programming paradigms. The more popular version being FORTRAN I, FORTRAN II, FORTRAN IV, FORTRAN 77, and FORTRAN 90. The current standard is FORTRAN 95 (ISO/IEC 1539-1), and it includes modern structured programming features in a traditional FORTRAN framework.

FORTRAN has good support for mathematics, especially floating-point computation and lacks modular programming structures and implicit declarations. For these reasons, it has been a staple language in the scientific community for complex scientific calculations, engineering models, statistics, and signal processing. In addition, its relatively simple code structure and static data structures make it highly compatible to compiler optimization and targeting to special hardware (e.g. vector supercomputers).

FORTRAN is normally case-insensitive. An early limitation of the language was that the position of text on lines was significant, however, FORTRAN90 and later versions support free-form input.

Some free FORTRAN compilers are available, the most significant of which is the GNU FORTRAN compiler, g77. Free compilers for FORTRAN 90 or FORTRAN 95 are under development by GNU. FORTRAN was standardized by ANSI in 1978 and is under the auspices of the Committee for Information Technology Standards (INCITS).

4.1.6 Java Programming

Java is an object-oriented language similar to C++ yet simplified, in some respects, to eliminate language features that caused frequent C++ programming errors. Java was designed and developed by Sun Microsystems.

The feature set of Java is fairly broad: it has inheritance, strong type checking, modularity (packages), exception handling, polymorphism, concurrency, dynamic loading of libraries, arrays, string handling, garbage collection, and an extensive standard library. The Java standard library packages include extensive I/O facilities, a comprehensive GUI toolkit, collection classes, date/time support, cryptographic security classes, distributed computation support, and system interfaces.

The fundamental structural component of Java is the class. All data and methods in Java are associated with some class; global data or functions do not exist in Java as in C++. Developers of the language did not include features that would jeopardize the simplicity or safety of the language, for these reasons Java has no true pointers, no true multiple inheritances, no operator overloading and no macro preprocessor. To circumvent the serious shortcoming of the lack of

multiple inheritances, Java supports the definition and inheritance of multiple stateless "interfaces", which covers most of the areas where multiple inheritances is usually desired. Similarly, while Java contains no facility for generic functions, the need for such functions is greatly reduced since the language imposes a rooted class hierarchy whereby all object classes inherit from the root class 'Object'.

Java version 1.1 added significant features of reflection and object class manipulation to support object serialization I/O. The newest version of the language, Java 1.2 added nested classes, persistence, and reflection as well as many additional standard libraries.

Java is often compiled to platform-independent byte-codes. A Java Virtual Machine (JVM) running on the target computer interprets these byte-codes. A strict definition of the byte-code format exists thereby supporting portability to multiple platforms because JVMs exist for most of the popular operating systems, like UNIX, Macintosh OS and Windows.

Compiled byte-code can also be converted directly into machine language instructions by a just-in-time (JIT) compiler. The advantage is that Java programs compiled a JIT usually run much faster than when the bytecode is executed by an interpreter.

In additional to the normal application development support, Java can also be used to develop embedded programs, called 'applets', for web browsers and other Java-enabled platforms. This capability continues to play an important part in Java's acceptance and proliferation. A key aspect of the acceptance of this

approach is that Java's standard library package includes a security manager to restrict the capabilities of Java applets to ensure safety of end-users computers.

Commercial Java compilers and development environments are readily available. The most popular are products from Sun, Symantec and Microsoft. In addition, a GNU Java compiler is available at no cost.

4.1.7 Visual Basic Programming

Microsoft developed Visual Basic as an application development tool. Visual Basic uses an advanced structured dialect of the standard Basic programming language.

Visual Basic supports the following primitive data types: integers, reals, strings, booleans, currency, dates, and object references. It is a loosely typed language since variables may be declared, but typing is not required. Composite data types such as arrays and user-defined records are available. Control structures include conditional and iteration constructs and rudimentary error handling capabilities.

Visual Basic was initially designed to be an interpreted language, however newer implementations include native code compilers. It is not presently considered an advanced language appropriate for rigorous scientific computation, yet that seems to be wearing down as more advanced enterprise applications are being developed based on Visual Basic. It is a commercial product that runs only on Microsoft Windows platforms. A limited capability subset edition is available for free.

4.2 Categories of Computing Platforms and Operating Systems

Supercomputers are used primarily for specialized scientific computing applications that require enormous amounts of mathematical calculations. Examples of such applications are weather forecasting, animated graphics, fluid dynamics, and nuclear energy research.

Mainframes are large centralized computers that in the past provided the bulk of business computing through time sharing applications. Mainframes and comparable systems, namely computer clusters, are still useful for large scale tasks, such as centralized billing systems, inventory systems and database operations. One important difference between supercomputers and mainframes is that supercomputers are designed to run just a few large intense applications concurrently whereas a mainframe is designed to execute many different applications concurrently.

The term 'server' usually refers to a computer or groups of computers used for tasks like application serving, intranet serving and/or internet serving. Servers can be sized to act as mainframe replacements yet are often dedicated to just a few applications or tasks. The term server can also refer to a particular application that is performing a task or managing resources rather than referring to the entire computer itself.

Workstations are more powerful versions of personal computers. Generally targeted for use by only one person, workstations generally run a more powerful version of a desktop operating system and run on more powerful hardware.

Real time operating systems are specifically designed to handle events that happen in real time. Real time systems are grouped according to the response time that is acceptable (seconds, milliseconds, microseconds) and according to whether or not they involve systems where failure can result in loss of life.

4.3 Operating Systems

4.3.1 Windows

Windows 32 bit operating systems were originally designed and marketed for higher-reliability business needs. Versions include Windows NT 3.1, NT 3.5, NT 3.51 and NT 4.0. Microsoft then moved to combine their consumer and business operating systems with the release of Windows 2000, Windows XP and Windows Server 2003 and the soon to be released Windows Longhorn. Windows 2003 has had some recent success in running larger business class server systems for larger applications than was possible with previous versions of Microsoft operating systems.

4.3.2 VMS and OpenVMS

Virtual Memory System (VMS) is a multi-user, multitasking, virtual memory operating system that runs on Digital Equipment Corp (DEC) VAX and Alpha minicomputers and workstations. VMS was introduced in 1979 along with the first VAX minicomputer. VMS has undergone many changes over the years and it is still widely used in legacy systems.

The OpenVMS operating system is a descendant of the VMS operating system. OpenVMS is a multi-user, multiprocessing operating system designed by DEC that is now owned and supported by Hewlett-Packard. OpenVMS is designed for use in time-sharing, batch processing and transaction processing systems. It has been ported to the DEC Alpha and Intel Itanium platforms and is still a viable operating system for server type business applications yet is no longer an industry leader and continues to lose market share to more popular operating systems.

4.3.3 UNIX

UNIX is a multi-user, multitasking operating system that was developed by Bell Labs in the early 1970s. It was designed to be a small, flexible system to be used exclusively by programmers. UNIX was one of the first operating that could be installed on almost any computer since the main prerequisite was that a C compiler was available. The cost of UNIX was low when compared to other operating systems of the time due to anti-trust regulations limiting Bell Lab from marketing it. Due to the portability and low cost, the popularity of UNIX soared.

Bell Labs distributed the operating system in its source language form, so other companies have been able to create other versions that were tightly integrated with their own platform products. Due to its portability, flexibility, and power, UNIX has become a leading operating system for many different computer workstations. AT&T began to market UNIX in earnest after its breakup in the early 1980s. AT&T also began the long and difficult process of defining a

standard version of UNIX. Historically, it has been less popular in the personal computer market. Today, the Open Group owns the trademarked "UNIX" and the "Single UNIX Specification" interface. An operating system that is certified by The Open Group to use the UNIX trademark conforms to the Single UNIX Specification.

4.3.4 Solaris

Solaris is a UNIX variant developed by Sun Microsystems to run on Sun's popular workstations. The most recent versions are powerful natural 64-bit systems written to run on SPARC (Scalable Processor Architecture) processors developed by Sun. When released in December of 2004, Solaris 10 is expected to be available for SPARC, x86 and the AMD64 processors. In addition, Sun advertises that Solaris 10 will allow native Linux binaries to run since they are incorporating the Linux standard base into the Solaris 10 kernel. Solaris is presently proprietary software but Sun has allowed both binary and source versions to be freely downloadable at various times. Sun has recently confirmed their intention to make Solaris open source, but details have not been finalized (CNETAsia News).

4.3.5 HP-UX

HP-UX is Hewlett-Packard's proprietary implementation of UNIX. It presently runs on the HP PA-RISC processors and Intel's Itanium processor. HP-UX has been a leader in advancing the capabilities of UNIX such as being the first to use access

control lists for file access permissions and the first to include a built-in Logical Volume Manager designed to be more flexible than normal physical partitioning.

4.3.6 SCO UnixWare

SCO UnixWare is a UNIX operating system sold by the Santa Cruz Operation Corp. The SCO group is currently involved in a dispute with various Linux vendors and users, with the assertion that Linux violates some of SCO's intellectual properties surrounding UNIX. Although there are many skeptics of SCOs claims, the SCO initiated lawsuits, if upheld by the courts, may impact the future of both Linux and UNIX. The success or failure of the claims will also impact the financial future of the SCO Group. SCO has, to date, made little progress in this dispute.

4.3.7 Linux

Linux is an open source implementation of UNIX initiated by Linus Torvalds, which runs on many different hardware platforms including Intel, Sparc, PowerPC and Alpha processors. Open source refers to software source code that is available to the general public, free of charge, for use and/or modification. The open source approach sprouted from within the technical community as a response to proprietary software owned by corporations. Because the source to Linux is open source, it is easy to customize and to update quickly. This flexibility has allowed Linux to be ported to a wide variety of platforms ranging from embedded systems to clusters of hundreds of servers. Hundreds of application programs have been written for Linux, some of these by the GNU project.

4.3.8 GNU

GNU operating system is a UNIX-compatible software system. GNU is a recursive acronym for "GNU's Not UNIX. The GNU project was started by in 1983 Richard Stallman at the Massachusetts Institute of Technology with the initial goal of creating a complete and free operating system. The GNU project is now managed by the Free Software Foundation (FSF). The FSF states that the Linux operating system is actually a version of GNU using the Linux kernel, and should therefore be called GNU/Linux. The present philosophy of the GNU project is to produce software that is non-proprietary and free to use. Anyone can download, modify and redistribute GNU software with the only restriction being that further redistribution cannot be limited. Over 3,000 programs are presently available, free of charge, from their web site.

4.3.9 OS Comparisons

The operating systems studied provide excellent connectivity, stability and scalability. There is little doubt that the modern versions of these systems would be able to provide a stable platform for the NG-SOCS system. Differentiators would be how well each system supports the possible development environments and the overall cost impact. As argued by Fink (90-96), the use of Linux for mission critical applications has become more prevalent as the stability of the system grows and the business community becomes more comfortable with notion of open source. Table 4.2 shows a comparison between the TCO between Microsoft and Linux operating systems over a 3 year period for a

modeled company with 250 users. As would be expected, the total cost of any system is not due simply to one or two line items like the initial purchase price of an OS or the continued maintenance agreement, but on a series of well-documented and definable costs. Linux, being open source, has a clear advantage over a proprietary OS.

Table 4.1 Comparison of TCO Between Linux and Microsoft OS

	Microsoft Solution (TCO Over 3 Years)	**Linux/Open Source Solution (TCO Over 3 Years)**	**Savings Achieved by Using Linux (Over 3 Years)**	**Percent Saved (Over 3 Years)**
Existing Hardware & Infrastructure is used	$733,973	$482,580	**$251,393**	34.26%
New hardware & Infrastructure is purchased	$1,042,110	$790,717	**$251,393**	24.69%

Source: Cybersource. "Linux vs. Windows Total Cost of Ownership Comparison." 2002. Oct. 12, 2004. Retrieved Nov 12, 2004: http://www.cyber.com.au/cyber/about/linux_vs_windows_tco_comparison.pdf

4.4 Database Technologies

4.4.1 Oracle

Oracle Database is an industry leading relational database produced by the Oracle Corporation. Historically, Oracle has targeted high-end workstations and minicomputers as the server platforms to run its database systems. Its relational database was the first to support the SQL language, which has since become the industry standard. Oracle database is used for application support in large and small enterprises and has an excellent reputation for a highly reliable and stable

platform. The latest version, Oracle Database 10*g*, is the first relational database designed for Grid Computing. Oracle versions exist for all major operating systems, including Windows, UNIX and Linux.

4.4.2 MySQL

MySQL is an open source relational database management system that relies on SQL for processing the data. MySQL provides APIs for many languages including C, C++, Eiffel, Java, Perl, PHP and Python. MySQL can run on UNIX and Linux operating systems. In addition, Microsoft environment operation is possible by using OLE DB and ODBC providers. A MySQL .NET Native Provider is also available, which allows native MySQL to .NET access without the need for OLE DB. MySQL is often used for Web applications and for embedded applications and has become a popular alternative to proprietary database systems like Oracle because of its speed, reliability and open source approach. The database is available for free under the terms of the GNU General Public License (GPL) or for a fee to those who do not wish to be bound by the terms of the GPL.

4.4.3 DB2

IBM DB2 is a family of relational database products offered by IBM. DB2 provides an open database environment that runs on a wide variety of operating systems including OS/390, UNIX, Linux, HP-UX, Solaris, SCO UnixWare and Window 2000. DB2 includes a range of application development and management tools. DB2 databases can be accessed from any application

program by using SQL, Microsoft's Open Database Connectivity (ODBC) interface, the Java Database Connectivity (JDBC) interface, or a CORBA interface broker. According to IBM, DB2 is an industry leader in terms of database market share and performance. Although DB2 products are offered for UNIX and Windows based systems, DB2 trails Oracle's database products in UNIX-based systems and Microsoft's Access in Windows systems.

4.4.4 SQL Server

SQL Server is Microsoft's relational database system targeted for enterprise class operation. SQL Server provides core support for Extensible Markup Language (XML), SQL and Internet queries. It also is designed to support the rapid development of enterprise-class business applications. Drawbacks are that it runs only on Windows operating systems.

4.5 Web Servers and Web Technologies

4.5.1 Apache Web Server

Apache Web Server is a public-domain open source Web server developed and maintained by the Apache Group. The first version of Apache, based on the NCSA httpd Web server, was developed in 1995. Since it has become a successful open-source product, significant public libraries of Apache add-ons have been developed and are also free to the public. In many respects, Apache has similar open source lineage and history to that of Linux. Present versions of Apache run on UNIX, OS/2 and Windows. As part of the Apache Web Server,

Apache Tomcat is the servlet container that is used for support of Java Servlets and JavaServer pages.

4.5.2 Microsoft Internet Information Server

Microsoft Internet Information Server (IIS) is a set of internet based services for Windows machines and contains a HTTP web server. IIS is tightly integrated with the Microsoft operating systems, which gives it the advantage of being relatively easy to administer. A downfall is that IIS cannot be used on other operating system platforms like UNIX or Linux.

4.5.3 WebSphere

The WebSphere Application Server is the IBM solution for the integration and operation of applications across multiple computer platforms. The target user community is e-business applications with solutions for web servers, application servers and the like. Many portions of the Websphere product could be classified as middleware. All WebSphere products use open standards like the Java 2 Platform, Enterprise Edition (J2EE) and XML. Present versions run on UNIX, OS/2 and Windows.

4.5.4 Common Gateway Interface

A Common Gateway Interface (CGI) program is used to dynamically provide web server based information to the end-user. The program can be written in any programming language, including C, Perl, Java, or Visual Basic. CGI is a server-side technology for providing dynamic information compared to client side technologies

such as Java applets, Java scripts or Active X controls. A limitation of the CGI is that a new server process is started every time a CGI script is executed, adding substantially to server overhead and reducing response time.

4.5.5 Active Server Page

Active Server Page (ASP) is a Microsoft implementation for a dynamically created web page utilizes ActiveX scripting, usually from either VB Script or JScript. The mechanism for this technology is that the web server servers an ASP to a browser by generating a page with HTML code at the time of the request. For this reason, ASP is similar to CGI scripts, but they enable Visual Basic programmers to work with familiar tools.

4.5.6 ASP.NET

ASP.NET is the latest Microsoft server-side Web technology that improves on the speed of standard ASP. In ASP.NET, each page is compiled into an intermediate language by a .NET compiler. A JIT compiler then translates the intermediate code to native machine code which is run on the processor. The native code running on the processor equates to pages loading much faster than classic ASP pages where embedded VB Script or Jscript had to be continuously interpreted.

4.5.7 Java Servlets

Java servlet technology is a component-based, platform-independent method for building Web-based applications. Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. Servlets can

also access a library of HTTP-specific calls and receive all the benefits of the mature Java language, including portability, performance, reusability and crash protection. This technology has overcome the performance limitations of CGI programs. Third-party servlet containers are available for Apache Web Server, Microsoft IIS, IBM's WebSphere and others.

4.5.8 JavaServer Pages

JavaServer Pages (JSP) is a server-side extension of Java servlet technology that allows for the dynamic generation of HTML and XML web pages. It allows Java code and certain pre-defined actions to be embedded into static content. JSP allows for the creation of JSP tag libraries that act as extensions to the standard HTML or XML tags and provide a platform independent way of extending the capabilities of a web server. To accomplish this, JSPs are compiled into the servlets by the JSP compiler.

4.5.9 PHP Hypertext Preprocessor

PHP self-referentially stands for PHP Hypertext Preprocessor is an open source, embedded scripting language used for the creation of dynamic web pages. PHP code is embedded within an HTML document by the use of special tags. The syntax of PHP is similar to that of Perl and C and is therefore favored by UNIX developers. In a fashion similar to that of ASP, the programmer can jump between PHP and HTML code, which provides flexibility to use the most appropriate language to accomplish a task. PHP can perform any task that any CGI program can do, but its strength lays in its compatibility with many types of

databases. Also, PHP can talk across networks using IMAP, SNMP, NNTP, POP3 or HTTP.

4.6 Inter Process Communication Techniques

It was necessary to research the topics contained within this section to ensure that there was the proper foundation and understanding of the underlying available technologies.

The Component Object Model (COM) is a software architecture developed by Microsoft to assist in the building of component based applications. COM objects are discrete components that provide interfaces that allow other components and applications to access their features. COM objects can be built from multiple programming languages and can easily communicate between themselves and are therefore referred to as being 'language independent'. COM objects have built-in interprocess communications capability that previous technologies from Microsoft did not support. Distributed Component Object Model (DCOM) is an extension of COM that allows DCOM components to communicate across networks. DCOM uses the Remote Procedure Call (RPC) mechanism to transparently send and receive information between COM components (i.e. clients and servers) on the same network. RPC is a protocol that allows an application to execute a program on a different computer and have the results of that execution returned to the calling application.

An architecture developed by IBM, System Object Model (SOM), allows binary code to be shared by different applications. SOM is a complete

implementation of CORBA. DSOM is a distributed version of SOM that allows binary objects to be shared across networks. SOM and DSOM serve the same purpose as Microsoft's competing COM and DCOM standards.

Remote Method Invocation (RMI), is a set of protocols developed by Sun Microsystems that enables Java objects to communicate remotely with other Java objects. In comparison to DSOM and DCOM, RMI is a much simpler protocol, yet only works with objects written in Java.

4.7 Development Environments/Active Frameworks

4.7.1 Common Object Request Broker Architecture

Common Object Request Broker Architecture (CORBA) is an architecture that enables program objects to communicate with one another independent of the programming language used for the objects or the operating system that the objects run on. The Object Request Broker (ORB) is a component in CORBA that acts as the middleware between clients and servers. In the CORBA model, a client can request a service without any knowledge of the server(s) on the network. The various ORBs receive requests, forward the requests to the appropriate servers and return results back to the client. Interface Definition Language (IDL) is used to define an interface for each object. For example, a legacy application, such as a billing system, can be wrapped in code with CORBA interfaces and opened up to clients on the network. The IDL interface definition is independent of programming language due to the use of

standardized mappings from IDL to C, C++, Java, Ada, COBOL, Smalltalk, Lisp, Python, and IDLscript (see Figure 4.1).

Interoperability is supported by the standardized protocols GIOP and IIOP. It is the use of these well-defined protocols that allows for the interoperation of CORBA objects independent of the programming language or operating platform. A long list of idiosyncrasies for different servers has required the development of the Portable Object Adapter (POA). The POA allows or the creation of a very stable environment for the server side CORBA implementation.

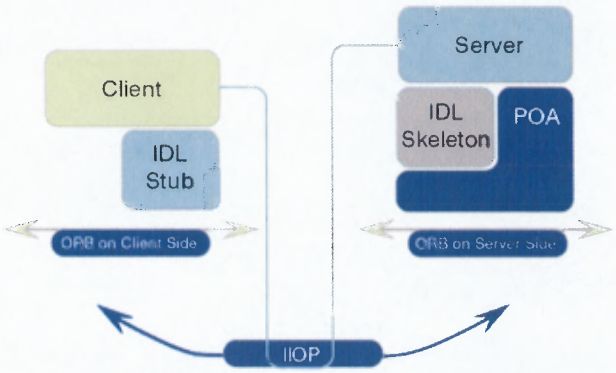

Figure 4.1 Participants in a CORBA request (IBM, 2000).

An industry consortium known as the Object Management Group (OMG) maintains the CORBA specification. There are many implementations of CORBA, with the IBM SOM and DSOM architecture a leading product. Two competing models are Microsoft's COM and DCOM and Sun Microsystems' RMI.

As reported by Wang, Schmidt and Levine, well-documented performance bottlenecks have existed in conventional CORBA implementations. While the industry continues to address and develop solutions to this issue, continued

research is needed in this area in order for this technology to be considered appropriate for high performance, real-time applications.

4.7.2 .NET

.NET is a wide-ranging family of products that provide for the aspect of developing distributed applications and managing the underlying servers and systems. .NET is provide by Microsoft and built on industry and Internet standards. From a developer's viewpoint, Visual.NET is a comprehensive series of products containing a suit of software technologies used for development of both small and enterprise class applications that can be integrated as needed. When defining .NET, Microsoft claims that it enables a high level of software integration through the use of Web services. Web services being defined as small discrete building blocks that connect to each other, as well as to other larger applications over the Internet.

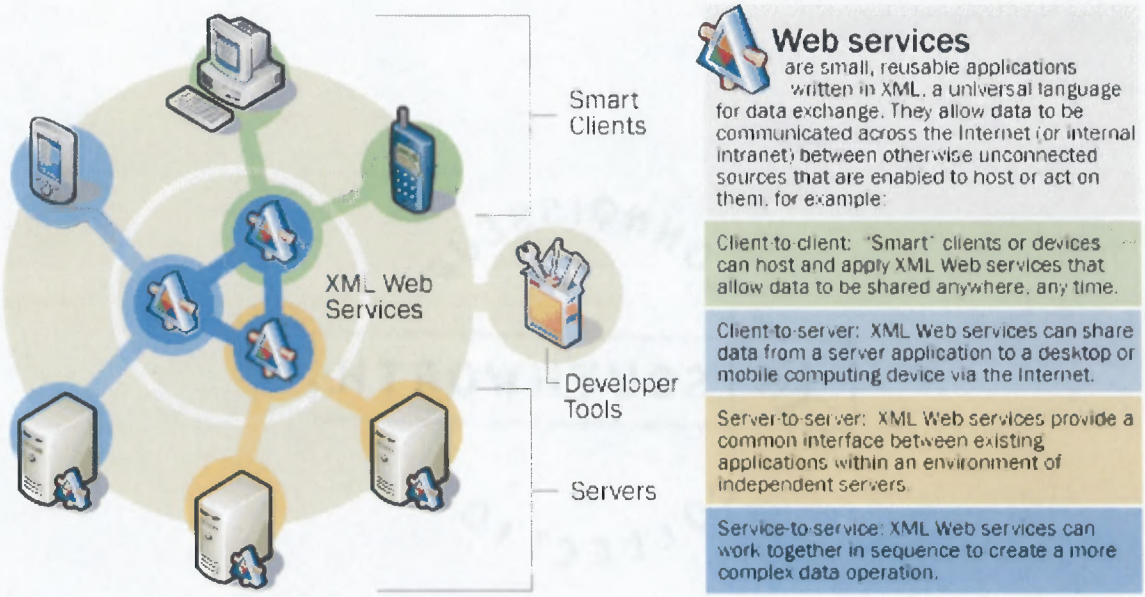

Figure 4.2 Components of Microsoft .NET- Software (Microsoft, 2003).

.NET is highly integrated with other Microsoft products that provide web services, general server applications and system security etc. Microsoft touts one main advantage of using .NTE technologies is that users will have access to their information on the Internet from any device, anytime, anywhere. This is obviously a key advantage that modern applications can provide, if not necessarily a key requirement.

In .NET, multiple programming languages can easily be used to within one application. Languages presently supported are C#, VB.NET, Jscript, ASP.NET, C++, FORTRAN, Perl, and Python. Source code is converted to an intermediate language code (IL) by the appropriate language compiler, which is then converted to native code at execution time by the .NET JIT compiler. Since all code gets converted to IL first, components written in different languages can

easily be used within a particular application. This is generally referred to as language independence.

.NET for software development is only compatible with Microsoft Windows operating systems although the developed applications are platform-independent. The latest versions contain many built-in features including Internet integration and features intended to enhance security.

Specifically for web applications, .NET allows for the creation and use of XML-based applications, processes, and websites as services that share and combine information and functionality with each other by design, on any platform or smart device.

4.7.3 Java 2 Platform Enterprise Edition

Java 2 Platform Enterprise Edition (J2EE) is a platform independent software environment from Sun Microsystems. It is used for developing, building and deploying enterprise applications based on web and other standard technologies. The J2EE platform contains of a set of services, APIs and protocols that provide the functionality for developing multitiered applications using web technologies. Database connectivity is achieved using Java Database Connectivity (JDBC), the Java equivalent to ODBC. Transparent services such as threading, concurrency, security and memory management are provided by Enterprise Java Beans (EJB). At the client, J2EE supports HTML as well as Java applets. It relies on Java Server Pages and Servlet technologies to create HTML or other formatted data for the client. JBOSS is an open source application server, written in Java, which

can host business components developed in Java. It uses the Enterprise Java Beans specification as the underlying technology.

J2EE is considered by many in the field to be the leading enterprise platform today, and is well documented to be the leading solution for web based application development (Sun Microsystems).

Clients may range in complexity from simple HTML forms to sophisticated Java applets. Servlets will frequently use some kind of persistent storage, such as files or a database (see Figure 4.3).

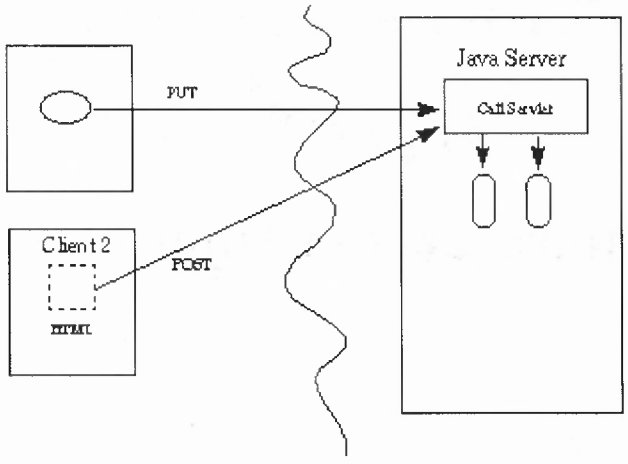

Figure 4.3 Clients and Java Servlets. (Sun Microsystems, White Paper).

Combined, the J2EE technologies allow developers to create enterprise applications that are platform independent and scalable, while providing for the integration of legacy technologies and systems if needed (see Figure 4.4). Another benefit of J2EE is that it is possible to get started with little or no cost. Sun allows free download of the J2EE implementation and many open source

tools are available from independent developers and groups that extend the platform or simplify development.

J2EE is compatible with most versions of UNIX, Linux and Windows.

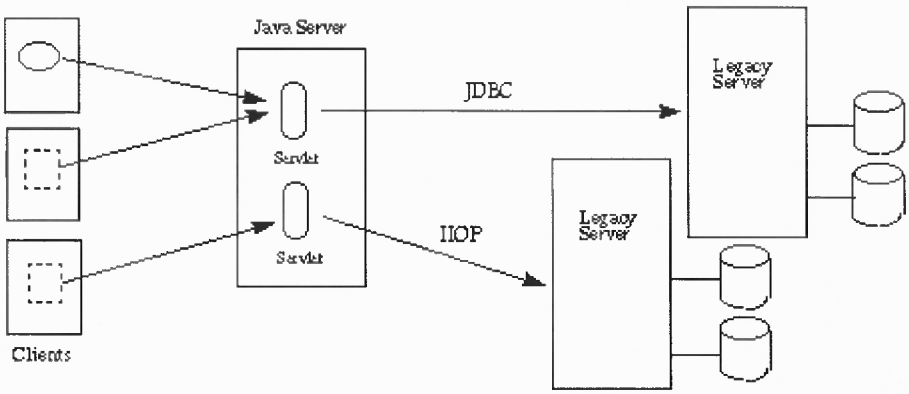

Figure 4.4 Servlets and application tiers (Sun Microsystems, White Paper).

CHAPTER 5

HETEROGENEOUS PLATFORMS AND DIFFERENT PROGRAMMING LANGUAGES

As with many modern applications, it is important the NG-SOCS solution address the heterogeneous platform issue. NG-SOCS must be able to run client applications on Windows operating systems since end-users use company provided laptop computers. In addition, the system must also interface with existing satellite control systems at Astra and Americom, which run of different platforms.

The NG-SOCS solution must also take into account the re-use issue of existing FORTRAN, C and Ada software to ensure that cost and schedule impacts due to re-design, re-testing and the creation of new documentation is kept to a minimum.

Given the existing technologies documented in the previous section, three viable approaches exist.

First, the use of CORBA technologies as middleware between the server and clients is a viable solution for the development of a robust system given the re-use desire of the existing legacy components. With respect to the desire for supporting heterogeneous platforms and multiple programming languages, research of existing CORBA capabilities supports the notion that these issues have been overcome by this technology.

Secondly, the use of .NET technologies as the umbrella environment for the development and maintenance of the new application would allow for connectivity to the client application running on laptop PCs using windows. Complete platform independence would not be achieved with this approach since the new server would have to be a Microsoft operating system for it to be able to act as the initial .NET development platform and then as the final application server. Also, .NET simplifies the use of source code from multiple languages via the language independence aspect of the environment. For this reason, use of using C and FORTRAN software would greatly reduce development time since porting of these modules to a common language would not be required. However, .NET support for Ada does not presently exist and therefore could be a potentially significant limitation that would require a significant code porting activity.

J2EE is the third software development environment robust enough for consideration. The J2EE platform consists of a set of services, APIs and protocols that provide the functionality for developing multitiered applications that can be integrated with legacy systems as required. Like the .NET solution, J2EE can provide applications that are platform independent. Yet unlike .NET, the J2EE development environment is also platform independent thereby providing additional flexibility to the programming team. Use of the existing Ada, FORTRAN and C software can be accomplished in this environment. Specifically, the Ada language has existing libraries that allow for the call of C routines from the Ada code. In this manner, the use of any VMS environment

variable or VMS specific system call could be replaced by C routines which provide the same functionality of the VMS environment routines in the target environment. In a similar manner, the same can be done for the existing FORTRAN code.

CHAPTER 6

CANDIDATE ARCHITECTURES

In this chapter, four approaches are presented and are reviewed to determine how well they would meet the developed requirements and how well they would address other concerns of the company. Only one solution can be deemed the most appropriate when weighted with real-world concerns such as the support for heterogeneous platforms, language independence, maximum reuse of legacy code, skill match with the existing development team and the total cost of ownership.

6.1 Tier'd Approach

In development of the proposed architectures, a tier'd approach to the division of tasks was reviewed and considered the most appropriate due to the logical division of the specified requirements and the given architecture of the existing FLD and PC-SOCS systems. Separation of responsibilities in this manner also supports an iterative development approach by allowing the incorporation of new functionality after the end-users have a chance to interact with initial capabilities. In addition, by using a tier'd approach, future modifications are more easily supported since changes would mostly likely be limited to one tier (i.e. change of the RDBMS from SQL to Oracle).

6.2 Candidate Architectures

As discussed in Sommerville (219–224), three standard architectural designs of software systems are the repository model, the client-server model and the abstract machine model. In some cases, the models are melded together to form a new model that fits the needs of an individual system. In the case of this NG-SOCS effort, a combination of the repository and client-server model is needed to address the requirements of stakeholders documented in Chapter 2 and the Appendix of this document. As reported by Wang, Schmidt and Levine and in Chapter 4 of this document, middleware technologies beside CORBA include products from Microsoft and Sun Microsystems, namely the .NET and J2EE product labels. The proposed NG-SOCS architectures use CORBA, .NET and J2EE technologies. In addition, the architecture proposed in the Bulut Thesis, from 2003, is reviewed.

6.2.1 Bulut Architecture

Bulut did previous work in this area as part of a CIS Thesis, Figure 6.1 depicts the architecture select as part of that effort. It is important to note that the requirements for the system designed by Bulut were significantly different than that for this NG-SOCS effort. Due to an enlarged stakeholder group, more complicated requirements were identified in this later, NG-SOCS, effort that Bulut did not have to address.

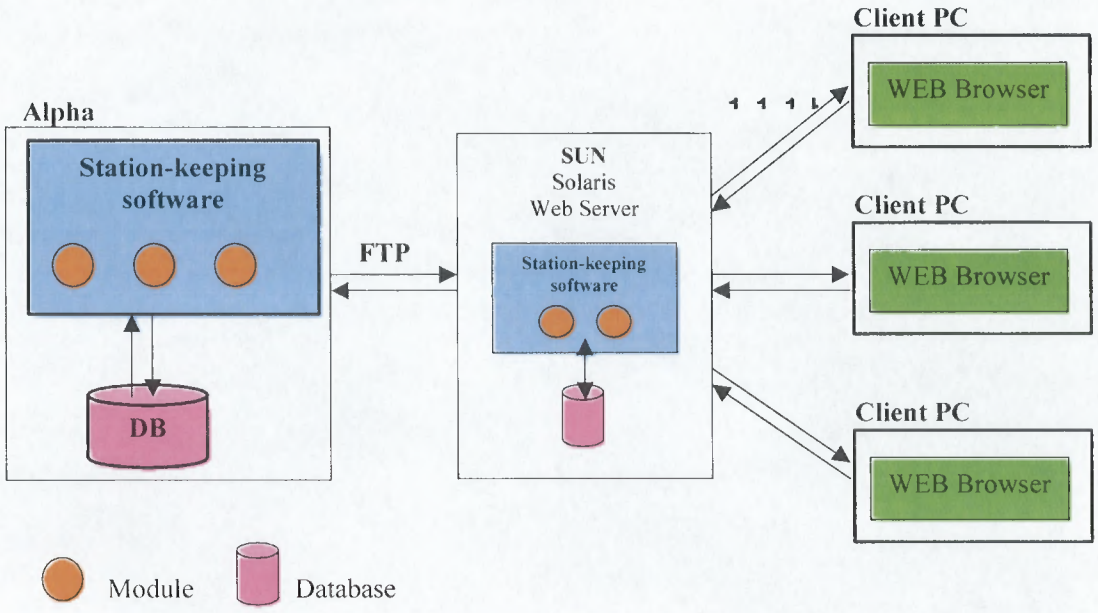

Figure 6.1 Architecture selected as part of previous work (Bulut, 2003).

The Bulut architecture correctly satisfies the requirements that were levied at the time for a new web browser interface for the front-end of the FLD system. It was well demonstrated that the approach selected was sound and met that short-term goal.

Since the Bulut work was limited to development of a new web interface only, it did not consider steps forward in the incorporation of graphical representation of the data, as is presently employed in PC-SOCS or 3rd party software solutions like those provided by AGI or ISI. In addition, this previous work was not tasked with development of a standalone capability to assist the business in leveraging laptop investments. For these reasons, it was not required for Bulut to consider fundamental changes to the orbital control system architecture, including the porting of modules to other languages.

Since this architecture cannot meet the requirements specified as part of the NG-SOCS effort, it is not considered as a potential solution.

6.2.2 J2EE & .NET Approach

The architecture depicted in Figure 6.2 can be supported by either a Microsoft or Sun Microsystems approach. Both vendors have a suit of technologies, under the .NET or J2EE banners, that would allow for successful fulfillment of the identified NG-SOCS requirements. Within each approach, some options on the selected product to meet a need are possible (i.e. Apache or WebSphere web server in the J2EE approach).

As discovered and discussed in Chapter 4, the Microsoft suit of technologies are geared to interaction with other Microsoft products while the Sun and IBM products are more supportive of industry standards, not just company proprietary ones. For this reason, if a Microsoft technology were selected to satisfy one development need, some program risk could be reduced if the rest of the solution were also satisfied using Microsoft products. Otherwise, a rapid-prototype approach would be recommended where the operation of intermixing of Microsoft with non-Microsoft products could be verified early in the development effort.

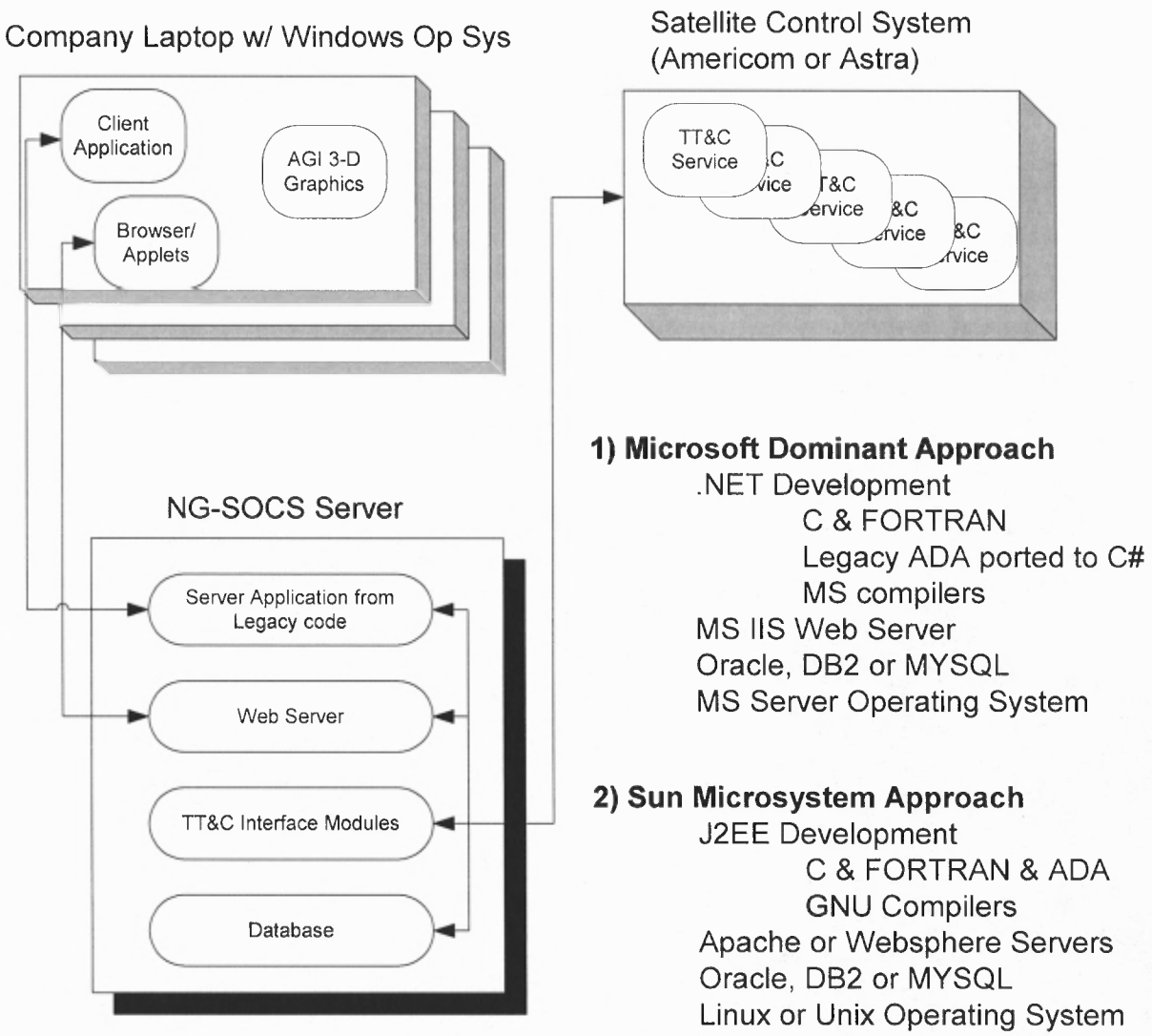

Figure 6.2 Architecture using either .NET or J2EE.

Performance measurements used to measure the speed of .NET and J2EE solutions were reviewed to determine if either product has a significant advantage. The data from both companies show that their particular product is faster and that their testing is more accurate than their competitors. For example, Microsoft states, " In short, the .NET results are actually more than two

to three times than Sun reported" (Microsoft Corporation, July 2004). In summary, for this application, performance differences of this size are not perceived to have a serious effect on the operation or response time of this application. In addition, final server platform selections will be based on performance tests performed on the development platform.

The UNIX, Linux, Windows and VMS operating systems have advantages and disadvantages and each can satisfy the needs of this effort. It is arguable that UNIX popularity stems from the fact that it can run on many available computer platforms. VMS and Window were limited in the platforms supported, at least originally. Linux stability and acceptance has grown to the point where it is a strong candidate for this system.

A review of the AGI and ISI software capabilities for graphical representation of the data in both 2D and 3D viewpoints shows that a client application with full access rights can interface with the appropriate modules. The AGI suite of modules have well defined APIs that would allow a client application to control and display graphically computationally intense data. Specifically, a viewpoint from behind a satellite in orbit, with the earth and other satellites also in the field of view, is possible. The depiction of the propagation of satellite orbit is also possible, either with AGI supplied propagation tools or with date supplied by the FLD/PC-SOCS modules.

6.2.3 CORBA Approach

The architecture depicted in Figure 6.3 shows how CORBA could be used to satisfy the requirements of this system. Since the client application will be a subset of the server code that runs the main application, insertion of CORBA between them does not seem to add much overall value. In addition, since the two other proposed architectures gracefully solve the web server to client browser aspect, the benefits of a CORBA approach is further diminished.

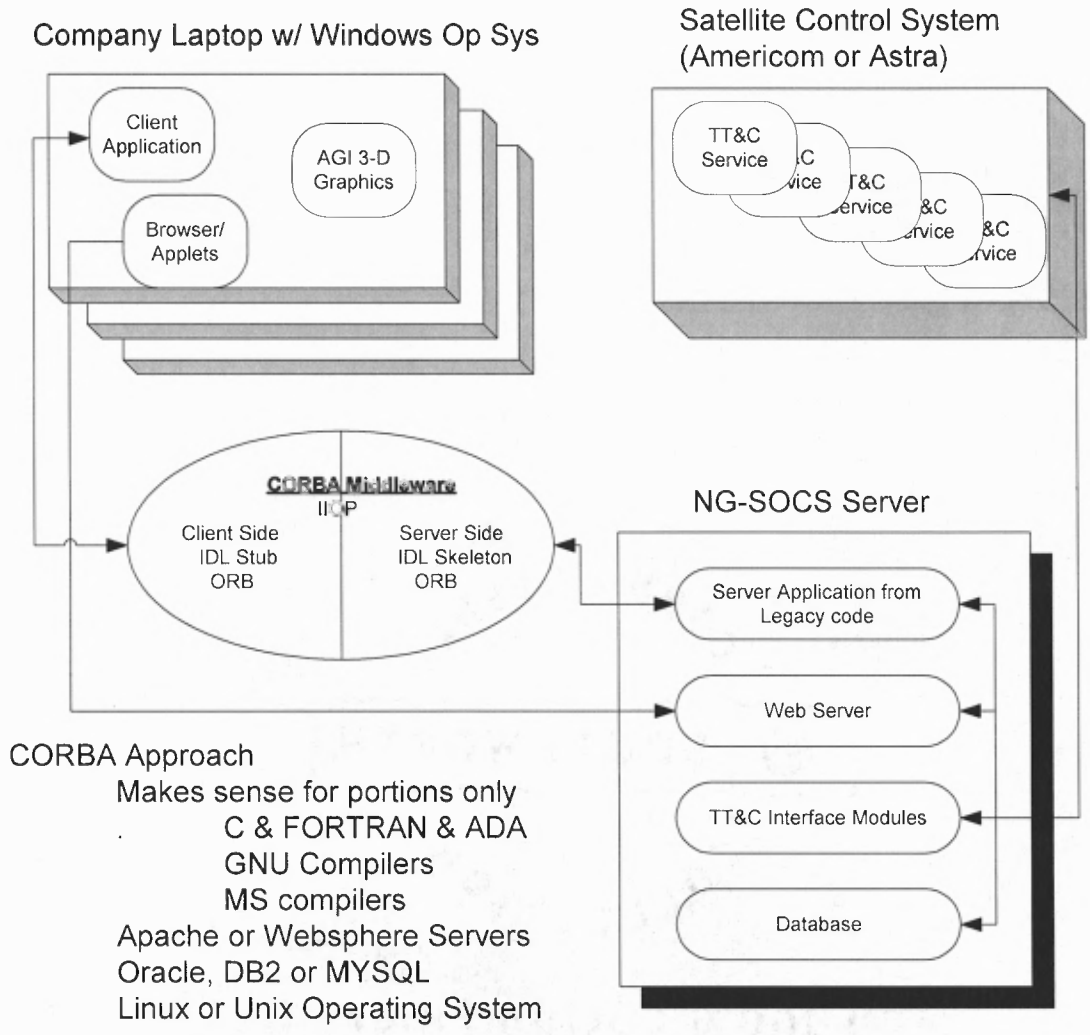

Figure 6.3 Architecture using CORBA approach.

6.2.4 Selected Approach

The criterion used in selecting the most appropriate solution is identified in Table 6.1. A total of fourteen criteria were used in the evaluation of the approaches. In addition to the specific technical concerns identified, additional criterion of staff competency and skill were considered due to fact that this is a corporate development effort. As stated by Roepke, Agarwal and Ferratt (8), development leaders must weight heavily their workforce competence and workforce

development prospects in the selection and design of new systems so that they are in position to support their systems and department strategic role within the business.

Given the identified criteria, the J2EE framework solution scored the highest compared to CORBA and .NET solutions. Additional analysis was considered where the relative weights of each criterion would be taken into account, however, that was not deemed to be needed since one solution scored so strongly.

Table 6.1 Trade Off of Proposed Architectures

Trade Off Matrix					
	Criteria	**CORBA**	**.NET**	**J2EE**	**Comments**
1	Mature Technology	3	3	3	Each technology has demonstrated the ability to create apps of this size
2	Able to Meet Functional Requirements	3	3	3	Review of each technology indicates that requirements can be met
3	Heterogeneous Platform Support	3	2	3	.NET development must be done on Windows Platform
4	Language Independence	3	3	1	Native J2EE is Java only but linking of Ada and FORTRAN modules possible
5	Legacy Code Support	3	1	3	No .NET ADA support. Either port to C# or use middleware for glue to ADA
6	Maintainability	3	2	3	Maintainability of ADA is excellent since it is a highly readable language
7	IT Support	1	3	3	Americom IT dept. has limited relationship or knowledge of IBM products
8	CSE Skilled in Technology	0	1	3	CSE dept. has experience with J2EE, not with CORBA or .NET
9	Staff Competence	2	3	3	Difficult to maintain CORBA expertise
10	Proven Throughput Solution	1	3	3	CORBA solutions have had known bandwidth problems in earlier versions
11	Security Concerns	3	2	3	Microsoft products with known security issues
12	3rd party graphical support (i.e. AGI)	3	3	3	APIs can support 3rd party satellite graphics
13	Initial Cost	0	0	3	Only J2EE is completely free
14	Total Cost of Ownership (TCO)	2	2	3	J2EE approach uses open source solutions
	Total Score	**30**	**31**	**40**	**Maximum possible is 42**

Notes: 1) Criteria scale is 0 to 3
2) Further weighting of each criterion was not deemed necessary since the J2EE architecture scored strongly.

CHAPTER 7

MERGER PLAN AND REQUIREMENTS COMPLIANCE

The next steps in this effort address normal iterative development activities with proof of concept milestones to ensure that risks to the successful completion of this effort are retired early. Additionally, the iterative development approach will be used to ensure that functionality is delivered and can be used as soon at it is available. This approach has the continued benefit that developers and stakeholders interact, review functionality and test the application more often than in more traditional development approaches like the waterfall model.

Requirements compliance will be monitored via the use of a compliance matrix that identifies, for each requirement, where in the code each specific requirement is satisfied. Additionally, acceptance tests will be defined to ensure that the specific functionality is fully operational and meets any new requirements that are uncovered during the iterative development approach.

During this effort, it became clear that since FLD and PC-SOCS had to solve the same problems, the underlying mathematical algorithms were very similar. For this reason, the selection of specific FLD and PC-modules was not critical at this stage of development. Selection will be explored and reviewed during the development phase to ensure that the most appropriate module is used. Since the FLD system has gone through more rigorous and well documented testing, the prevailing thought is to leverage that code whenever possible in order to reduce the number of coding errors introduced and lessen

the overall acceptance test time required. In addition, by using the maximum amount of FLD code in NG-SOCS, the effort to update existing documentation will be kept to a minimum.

APPENDIX

COMPILED REQUIREMENTS FOR THE NG-SOCS SYSTEM

This Appendix contains the requirements specification for the Next Generation Satellite Orbital Control System (NG-SOCS). The effort to develop this specification is detailed in Chapter 2 of this document.

Common Orbital Control System
Combined Requirements Specification

SES ASTRA

SES AMERICOM

An SES GLOBAL Company

September 29, 2004

SES AMERICOM

SES ASTRA

Table of Contents

SES AMERICOM

SES ASTRA

1 INTRODUCTION

1.1 PURPOSE

The purpose of this document is to clearly present the requirements for the creation of common orbital control software. This document is a result of meetings between stakeholders at SES-Americom and SES-Astra and represents an agreement on the functional and performance requirements of the system. The systems design and interface needs are also outlined, as well as a basis for the validation and verification of the completed system. The document will serve as a reference for all stakeholders; especially as confirmation that all requirements have been acknowledged and as a reference throughout the design and implementation stages to ensure that the requirements are being met.

1.2 SCOPE

This document describes the requirements for the creation of a common orbital control system. The system will also be referred to as NG-SOCS in this document. NG-SOCS will be packaged as a complete system and can be regarded as being composed of three distinct parts:

Near Real-time software system: This system will be designed to operate in a server environment and be capable of supporting end user graphics displays and intense mathematical computations. The system will be able to register transactions to satellite system databases for the interchange of information with other systems and will also exchange information via direct TCP/IP connections with other satellite operations and engineering systems.

Software documentation: This will be intended for those computer professionals responsible for maintaining or modifying the system, and will contain a complete and unambiguous description of the system and its operation.

Online user support: This will serve as a user's manual, covering general purpose users and administrative users. It will be available online, within the real-time software itself. There will be several parts to it: a frequently-asked questions page (FAQ), searchable index of common problems and how-to's and several tutorials for common tasks. The online user support will also contain documentation regarding security policies.

1.3 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

A list of terms, acronyms and abbreviations that are used in this document follows, and their meaning is included here for reference.

API	Application Program Interface - a set of routines, protocols, and tools for building software applications.
CDR	Critical Design Review
CFE	Customer Furnished Equipment
COTS	Commercial Off-the-Shelf
Delta V	Delta Velocity, the require velocity change in a satellites orbit (3 axis) to effect a desired orbit change.
DOP	Daily Operations Plan
FLD	Flight Dynamics System in use at SES-Astra
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IEEE	Institute for Electrical and Electronics Engineers
Intranet	Similar to the Internet, yet that portion of a private network that only internal (i.e. employees) can access.
Internet	An interconnected system of networks that connects computers around the world via the TCP/IP protocol
MB	Megabyte
NG-SOCS	Next Generation Satellite Orbital Control System. Also know as Common Orbital Control System.
PC-SOCS	PC Satellite Orbital Control System in use at SES-Americom
PDR	Preliminary Design Review
RAM	the most common computer memory which can be used by programs to perform necessary tasks while the computer is on; an integrated circuit memory chip allows information to be stored or accessed in any order and all storage locations are equally accessible
Secure Socket Layer	A protocol designed by Netscape Communications Corporation to provide encrypted communications on the Internet. SSL is layered beneath application protocols such as HTTP, SMTP, Telnet, FTP, Gopher, and NNTP and is layered above the connection protocol TCP/IP. It is used by the HTTPS access method.
SOCS	Satellite Orbital Control System
SRS	Software Requirement Specification
SSL	Secure Socket Layer -
TCP/IP	A protocol for communication between computers, used as a standard for transmitting data over networks and as the basis for standard Internet protocols.
TT&C	Telemetry Tracking and Control
Wi-FI	Short for wireless fidelity and is meant to be used generically when referring of any type of 802.11 network, whether 802.11b, 802.11a, dual-band, etc. The term is promulgated by the Wi-Fi Alliance.
Web	World Wide Web

SES AMERICOM
An SES GLOBAL Company

SES ASTRA

Web Portal	A Web site that offers numerous services and resources, often directing users to different stores [Webopedia]
World Wide Web	Computer network consisting of a collection of Internet sites that offer text and graphics and sound and animation resources through the hypertext transfer protocol
WWW	World Wide Web

1.4 REFERENCES

ASTM E1340-96, Standard Guide for Rapid Prototyping of Computerized Systems.[1]

IEEE Std. 830-1984 (1993, 1998) IEEE Recommended Practice for Software Requirements Specifications.[2]

IEEE Std. 730-1989, IEEE Standard for Software Quality Assurance Plans.

IEEE Std. 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.

IEEE Std. 828-1998, IEEE Standard for Software Configuration Management Plans.

IEEE Std. 1012-1988, IEEE Standard for Software Verification and Validation.

IEEE std. 1028-1997, IEEE Standard for Software Reviews

Webopedia Dictionary. http://www.webopedia.com/, online dictionary for computer and Internet technology definitions

SES-Americom, Daily Operations Plan (DOP) Database, API Specification, Version 1.2

SES-Americom, TT&C Ground Control System, API Specification, Version 3.5

SES-Astra, Operations Planner, API Specification, Doc Number 573345, Version 4.05

SES-Astra, Raytheon Ground Control System, API Specification, Doc Number 573921, Version 2.02.1

SES-Astra, FLD Tracking Data Interface Specification, Doc number [001 – Rev 2]

[1] ASTM publications are available from the American Society for Testing and Materials, 100 Barr Harbor Drive, West Conshohocken, PA 19428-2959, USA.

[2] 1EEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, PO. Box 1331, Piscataway, NJ 08855-1331, USA.

1.5 OVERVIEW

This document is divided into three major sections, and includes one appendix. It is the intent of this requirements specification to address the services, performance, attributes, user interfaces and constraints of the system to be developed. In addition, all external interfaces will be identified within this document. All appendices of this document are to be considered, in whole, as part of the overall system requirements. Below is an outline of the major headings and a summary of what each covers.

Section 1 (Introduction): this includes an introduction to this requirements document.

Section 2 (Overall Description): this provides an introduction to system that will be developed. It covers the following topics:

- Product perspective
- Product functions
- User characteristics
- Constraints
- Assumptions and dependencies

Section 3 (Specific Requirements): this is the most extensive section, and includes a detailed discussion of the requirements.

- Functional requirements
- External interface requirements
- Performance requirements
- Design constraints
- Attributes
- Other requirements

Appendix A (User Interface Recommendations and Prototype): this section lists the recommendations regarding the user interface, and provides a prototype of the various screens of the system.

SES AMERICOM

SES ASTRA

2 OVERALL DESCRIPTION

This section describes the system's characteristics and the limitations that impact the product and which will need to be considered in the requirements.

2.1 PRODUCT PERSPECTIVE

This system is a totally self-contained, allowing end-user use of all functionality independent of the status of other systems. Other systems in use by Spacecraft Operations and Engineering can provide data to this system and also receive data from this system as specified herein.

2.1.1 SYSTEM INTERFACES

2.1.1.1 SES-Americom Daily Operations Plan (DOP) Database, API Specification

The DOP Database will be updated, as new satellite orbital information is available from the NG-SOCS system, so that the satellite event and action levels are kept current at all satellite controller locations within the SES-Americom ground control network.

2.1.1.2 SES-Americom, TT&C Ground Control System, API Specification

Certain NG-SOCS data will be provided to all SES-Americom TT&C ground systems. This satellite data is critical to the Satellite Operations Department and will be given priority over the Americom/Astra computer network. At a minimum, the following data will be exchanged:

1. Range Files
2. Thruster Data Files
3. Orbital Elements for each satellite
4. Daily Operational Plan information
5. Moon and Sun hits predictions
6. Fuel calculations and associated information

2.1.1.3 SES-Astra, Operations Planner, API Specification

The Operations Planner will be updated as new satellite orbital information is available from the NG-SOCS system, so that the satellite event and action levels are kept current at all satellite controller locations within the SES-Astra ground control network.

2.1.1.4 SES-Astra, Raytheon Ground Control System, API Specification

Certain NG-SOCS data will be provided to all SES-Astra TT&C ground systems. This satellite data is critical to the Satellite Operations Department and will be given priority over the Americom/Astra computer network.

2.1.1.5 SES-Astra, FLD Tracking Data Interface Specification

Astra's standalone ranging system provides highly accurate range data on the satellites of interest and provides that data in accordance with this specification. NG-SOCS shall be able to receive this data for orbit determination purposes.

2.1.2 USER INTERFACES

A Graphical User Interface (GUI) will make all end-user functionality available. There will be three main interfaces with the system as identified in the following sections.

2.1.2.1 General User Interface

This interface shall be available to Satellite Controllers, Orbital Analysts, Earth Station Managers and System Administrators. This shall be the default view for Satellite Controllers and Earth Station Managers.

2.1.2.2 System Administrator Interface

This interface shall be available to NG-SOCS Administrators only. This shall be the default view for System Administrators and give the administrator access to system level control functions.

2.1.2.3 Orbital Analyst Window Interface

This interface shall be available to Orbital Analysts, Earth Station Managers and Administrators. This interface provides access to all the low level functions required to safely plan the satellite missions. This shall be the default view for Orbital Analysts.

2.1.3 HARDWARE INTERFACES

This system shall support standard Commercial Off-the-Shelf (COTS) external devices such as printers, monitors, keyboards and mice as needed to allow full use of system by any end user. If the system design is such that support of such external devices is the responsibility of the native operating system, the operation of such will be verified during system verification testing. This system has no interface to custom hardware.

2.1.4 SOFTWARE INTERFACES

It is expected that the NG-SOCS system will contain some internal COTS software interfaces in order to meet the requirements in this specification (i.e. relational database system, an operating system, or a

mathematical package). These COTS software applications must be explicitly identified at PDR and written approval given from the customer immediately thereafter.

2.1.5 COMMUNICATIONS INTERFACES

Networking and inter-process communications shall use TCP/IP communication protocols whenever possible. Use of other communications methods must be explicitly identified at PDR and written approval from the customer immediately thereafter.

Client-Server communications shall use pre-defined port numbers.

Use of Registered ports or Private Ports shall be limited to those services or communications requiring special functions that cannot otherwise be made available. Intended use of such ports requires specific disclosure during PDR and written approval from the customer immediately thereafter.

2.1.6 MEMORY CONSTRAINTS

The maximum memory usage for the system computers is 50% of the total memory available during anticipated peak usage periods of the system. The peak usage period shall be determined using NG-SOCS software functionality, operating system usage and any other application(s) executing that are intended to be resident on the NG-SOCS computer(s) (i.e. thin clients for external software applications).

For client computes delivered as part of this system, the maximum memory usage for operating system and NG-SOCS functionality shall not exceed 50% of the total memory available. A minimum of 256MB of RAM is required on client computers.

2.2 PRODUCT FUNCTIONS

This subsection of the SRS provides a summary of the major functions that this software will perform.

2.2.1 ORBIT DETERMINATION FUNCTION

The Orbit Determination (OD) capability will determine a particular satellite's orbit using the range and pointing data generated by other systems such as the real-time TT&C system in use at SES Americom or the SES Astra Ranging system. This data will be processed by this function and a set of orbital elements describing the satellite orbit will be generated.

2.2.2 EPHEMERIS PROPAGATION FUNCTION

This function will take a set of orbital elements that describe the motion of a satellite, apply various perturbing forces, and propagate the motion of the satellite to a designated time in the future. Based upon the location of the satellite and the time, this function will also predict the occurrences of predefined events (i.e. moon hits), which may effect the operation of the satellite.

2.2.3 MANEUVER PLANNING AND EVALUATION FUNCTIONS

Planning and evaluation capability for East-West (EW), North-South (NS), relocation and de-orbit maneuvers will be accomplished by this function. This function will also track the amount of fuel used for all executed maneuvers.

2.2.4 PLOTTING FUNCTION

This function will take selected data provided by other functions within the application and provide an on-screen or hardcopy graphical presentation of the data. Use of a third party graphics package is permissible.

2.2.5 ANTENNA FUNCTIONS

This is a collection of utilities that facilitate the computation of various antenna related information. Capabilities provided are antenna predicts, look angle computation and conjunctivity predictions.

2.2.6 DATABASE FUNCTION

A database will be maintained and used by all functions of this system. The database will be accessible from any terminal/workstation/computer, which is running this application, from any location that has connectivity to the company network and appropriate access rights.

2.2.7 DISPLAY FUNCTION

This function shall act as a file management capability within the application. It allows the user to rename, copy, move, edit and delete files without having to exit the program.

2.3 USER CHARACTERISTICS

Spacecraft Analysts will be the primary end-users of this system and are expected to have good experience with computers and networks, but not necessarily be advanced computer users. The Spacecraft Analysts will receive complete rigorous training in the use of this system.

Spacecraft Controllers will be casual users of the system and are expected to have good experience with computers and networks, but not necessarily be advanced computer users and will receive basic training in this system. They will be able to generate reports from the system.

Administrators are the application developers, and will be responsible for ensuring the day-to-day operation of the system. They are expected to have detailed knowledge of the system and have full access to reconfigure any part of the system to ensure its operation.

Earth Station Managers - These are a special case of Spacecraft Analysts, with additional access privileges. Managers of the system are expected to be competent computer users, have some familiarity and experience with the tools, and will be able to generate reports from the system.

SES AMERICOM

SES ASTRA

2.4 CONSTRAINTS

This subsection provides a general description of all items that will limit the developer's options.

1. This system shall allow all end users to run and access all functionality, as identified herein, from a PC running standard XP Windows operating systems. System server platforms, if used, may use other operating systems (i.e. UNIX) to run any portion of the system, so long as the above connectivity is possible.

2. The end user shall have the ability to run all functionality from any LAN/WAN connection at any SES Americom or SES Astra location. In addition, all functionality will be available from any remote location via a phone line using existing authentication systems (i.e. secure ID). Access to all information (range files etc) shall be capable when connected remotely.

3. The system shall support Spacecraft Operations on a 24x7 schedule.

4. The system shall be implemented as a secure application with usernames, passwords and timeout periods.

5. This system will consist of Commercial Off-the-Shelf (COTS) hardware and operating system(s). Custom hardware or operating system components shall not be used in any part of this system without prior written consent of the customer.

6. In order to keep the SES Americom and SES Astra systems aligned after deployment, a list of common source code modules and other items as needed, shall be defined. This common list shall include as a minimum any of the generic modules of Figure 1 that remain in the new system.

7. A configuration control (e.g. s/w change request, assignment of application managers) and change approval process shall be put in place in consequence.

8. Design of the system shall allow as much portability as possible to different platforms.

9. Since the modules in the FLD system (Figure 1) have been fully validated and used successfully for years, any change to the modules shall require full acceptance testing.

10. Due to the limited documentation on past PC-SOCS validation and verification, use of any PS-SOCS module or rewrite, requires full acceptance testing.

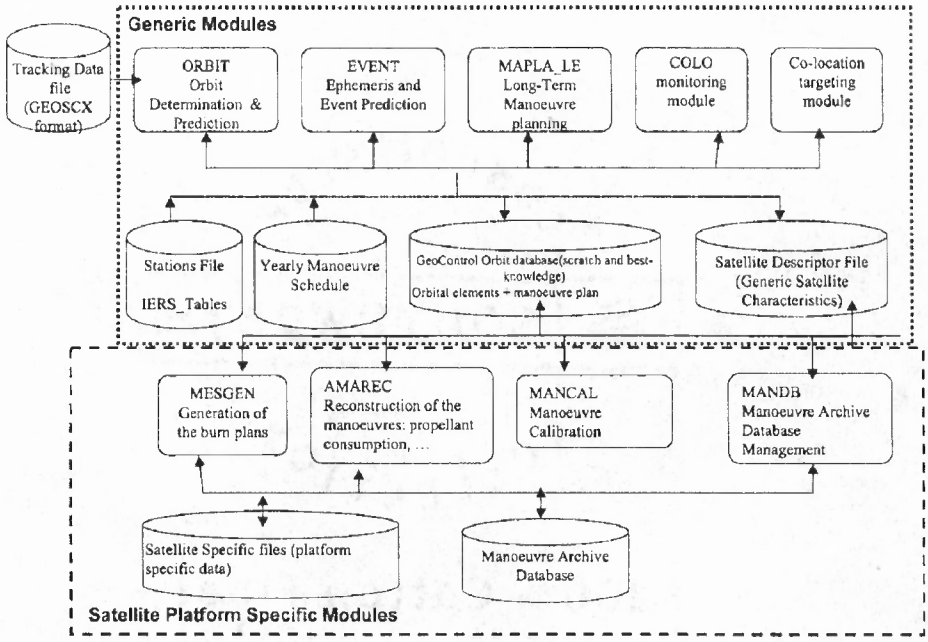

Figure 1: Current FLD software architecture

2.5 ASSUMPTIONS AND DEPENDENCIES

These requirements are based on the approach that the existing FLD and PCSOCS software modules can be used in the NG-SOCS system. If this were not possible, the requirements contained herein would have to be revisited.

3 SPECIFIC REQUIREMENTS

It is intended that this section of the SRS contain all of the software requirements to a detailed level that enables software engineers to design the system such that the requirements are met, and for testers to verify the operation of the system.

3.1 EXTERNAL INTERFACES

3.1.1 SYSTEM INTERFACES

3.1.1.1 SES-Americom Daily Operations Plan (DOP) Database, API Specification

The DOP Database will be updated as new satellite orbital information is available from the NG-SOCS system, so that the satellite action levels are kept current at all satellite controller locations within the SES-Americom ground control network.

3.1.1.2 SES-Americom, TT&C Ground Control System, API Specification

Certain NG-SOCS data will be provided to all SES-Americom TT&C ground systems. This satellite data is critical to the Satellite Operations Department and will be given priority over the Americom/Astra computer network. Means to verify data synchronization will be provided.

3.1.1.3 SES-Astra, Operations Planner, API Specification

The Operations Planner will be updated as new satellite orbital information is available from the NG-SOCS system, so that the satellite action levels are kept current at all satellite controller locations within the SES-Astra ground control network.

3.1.1.4 SES-Astra, Raytheon Ground Control System, API Specification

Certain NG-SOCS data will be provided to all SES-Astra TT&C ground systems. This satellite data is critical to the Satellite Operations Department and will be given priority over the Americom/Astra computer network. Means to verify data synchronization will be provided.

3.1.1.5 SES-Astra, FLD Tracking Data Interface Specification

Astra's standalone ranging system provides highly accurate range data on the satellites of interest and provides that data in accordance with this specification. NG-SOCS shall be able to receive this data for orbit determination purposes.

3.1.2 USER INTERFACES

A Graphical User Interface (GUI) will make all end-user functionality available. It shall have the look and feel of Microsoft Windows products. There will be three main interfaces with the system as identified in the following sections.

In general, the GUI shall

1. Provide tips to the user (information box appears when mouse is over key data)
2. Provide online hypertext help/documentation.
3. Create error and warning dialog boxes

3.1.2.1 General User Interface

This interface shall be available to Satellite Controllers, Orbital Analysts, Managers and System Administrators. This shall be the default view for Satellite Controllers and Managers.

3.1.2.2 System Administrator Interface

This interface shall be available to NG-SOCS Administrators only. This shall be the default view for System Administrators.

3.1.2.3 Analyst Window Interface

This interface shall be available to Orbital Analysts and Administrators. This shall be the default view for Orbital Analysts.

3.2 FUNCTIONAL REQUIREMENTS

This section, organized by functions, contains the necessary information for the software engineer to design the detailed specifications for the overall NG-SOCS system.

It is recognized that this section should include, at a minimum, a description of every input into this system, every output from the system and every function performed by the system in response to an input or in support of an output.

It is also recognized that this SRS will be complete, if and only if, definitions exist for the software responses to all realizable classes of input data and in all realizable classes of situations. This includes specifying the system response to both valid and invalid input values.

SES AMERICOM

SES ASTRA

3.2.1 ORBIT DETERMINATION AND ESTIMATION FUNCTION

3.2.1.1 General

The Orbit Determination (OD) capability will use the range and pointing data generated by the Americom real-time TT&C system or the Astra standalone ranging system to determine a set of orbital elements that describe a particular satellite's orbit.

This function will also support orbit estimation using continuous thruster burns, with the possibility to input time-varying thrust profile for each burn.

In addition, this function will support simultaneous estimation of the maneuvers Delta-V components with the initial orbital elements using tracking data before and after the maneuvers (so called orbit determination through maneuvers)

3.2.1.2 User Interface Requirements

1. The user will select the satellite for which the OD is to be run.
2. All ranges available for the selected satellite in the range storage area will be copied to the OD work area.
3. As ranges are copied, they are checked for reasonableness using existing SOCS algorithms.
4. The ranges are processed to put them in the appropriate format for processing by the OD function, and biases from the database are applied.
5. A plot of the range residuals is created for user evaluation/editing of the range files.
6. The user will have the ability to edit the range files used by the OD function. This will include the ability to delete files; delete range, azimuth, or elevation data from the files; and edit the phase calibration data within the files.
7. The OD function will use either the time of the first range (default), or a time specified by the user as an epoch for the OD.
8. The OD function will use as a starting point, orbital elements corresponding to the epoch. The elements will be either derived from the active ephemeris, OD file, maneuver file, or other user input.
9. The OD function should process the ranges to determine a new set of orbital elements using either a batch method, or Kalman filter method, as indicated by the user.
10. The OD function will determine range residuals and display as a text file and plot. These residuals will indicate how well the range data agrees with the orbit. The plot, or file, will also be used to remove/insert ranges for subsequent runs if necessary. The header of the plot shall contain RMS, mean, and full-scale values.

11. The OD will output a set of orbital elements for the input epoch. The element type shall be selectable by the user, and saved for subsequent runs. It shall be possible to convert to different element types conveniently. The element types shall be selectable from the following list:
    a. Kepler
    b. F/G
    c. Geodetic
    d. Rectangular ECF
    e. Equinoctial
    f. Rectangular ECI
    g. Polar ECI
    h. Intelsat 11- element set
    i. NORAD 2-line
12. Standard deviations shall be calculated for the elements and provided as part of the output.
13. Range, azimuth, and elevation biases shall be calculated as required by the user. In addition, it should be possible to calculate range biases to fit a given element set.
14. The user shall have the means to have the OD function iterate, using the elements, range, azimuth, and elevation biases that were output from the present run as input to the subsequent run.
15. The user shall have the ability to have the OD function solve for solar pressure to improve the relationship between the orbital elements and the ranges.
16. A means shall be provided to have the output of the OD function update the database.
17. The orbital elements generated shall be directly usable as input for ephemeris generation.

3.2.1.3 Input Requirements

1. Range data with or without azimuth and elevation.
2. Spacecraft specific data such as mass and area.
3. Station specific data such as location and antenna biases.
4. Global parameters such as solar pressure, gravity model, sun and moon gravity effects.
5. Epoch and initial elements.
6. Choice of batch, or sequential Kalman filter processing.

3.2.1.4 Output Requirements

1. Ability to calculate station biases, solar pressure and orbital elements.
2. Ability to update database with output.
3. Ability to use output directly as input to the next OD, or ephemeris run.
4. Orbital elements will be of the type specified by the user.
5. Ability to convert orbital elements to different types.
6. Standard deviations for output.
7. Range residuals as a text file and plot.

8. Ability to plot any orbit specific parameters, such as elements, biases, ranges.
9. Covariance matrix, or other indicator of the quality of the solution.
10. An Orbit determination report shall be available with screen, file, or printer output.

3.2.2 EPHEMERIS PROPAGATION FUNCTION

3.2.2.1 General

This function will take a set of orbital elements that describe the motion of a satellite, apply various perturbing forces, and propagate the motion of the satellite to a designated time in the future.

Based upon the location of the satellite and the time, this function will also predict the occurrences of events (i.e. moon hits), which may effect the operation of the satellite.

Upon user selection, a center-of-box based event prediction shall be possible. Center of Box defined as the area in the middle of the station-keeping control window.

3.2.2.2 User Interface Requirements

The user interface will allow specification of the source(s) of input data, propagation parameters, output parameters, output device/media, output file name and events templates to be created. The ability to select the perturbing force(s) to be applied during propagation shall also be provided.

3.2.2.3 Input Requirements

1. Orbital elements supplied by
   a. Orbit determination function and stored in database
   b. Element set contained within a specified file.
   c. Manual input (along with modifiable defaults for spacecraft mass and surface area).
2. Start and stop times for propagation or duration from start time (epoch).
3. Element set desired.
4. Output interval.
5. Output file name.
6. Event templates desired
   a. Start and stop times (subset of the ephemeris)
   b. Event template to be fulfilled.
7. Perturbing forces to be applied during propagation
   a. Solar pressure
   b. Earth gravity
   c. Lunar gravity
   d. Maneuver force vectors (single or multiple)

3.2.2.4 Output Requirements

The normal output from the ephemeris propagation function would be time-tagged, time-ordered table of orbital elements written to a user-named disc file and stored in the database. The file name will be user-selectable, but a default name would be provided.

This function shall generate the necessary information to operate the satellite as well as to produce the following input ASCII data files required by the real time systems at Astra and Americom for properly configuring the satellite:

1. On-board orbit propagator initialization or update
2. Sensor inhibit information
3. Orbital elements

This function will be capable of displaying this table on a split screen with all events and also capable of merging the two on request.

This function will provide the data for review and save the data to the database as directed. The ephemeris data will be available to the display function and therefore available for:

1. Screen plots
2. Hardcopy to designated printers
3. Off-line processing (ASCII format)
4. Saving to disc

A capability is required whereby multiple generated ephemerides can be compared. The comparison will provide spatial separation over a given period of time and also specify the time and distance of closest approach.

The ephemeris propagator will complete multiple event templates previously defined. Each template will be a time-tagged, time-ordered series of events selected from the following list

1. Ascending node
2. Descending node
3. Apogee
4. Perigee
5. Maximum longitude
6. Minimum longitude
7. Maximum latitude
8. Minimum latitude
9. Sun enter/exit primary ESA FOV
10. Sun enter/exit secondary ESA FOV
11. Moon enter/exit primary ESA FOV
12. Moon enter/exit secondary ESA FOV
13. Enter/exit earth eclipse of the sun
14. Partial earth eclipse of the sun
15. Enter/exit total earth eclipse of the sun
16. Enter/exit lunar eclipse of the sun
17. Partial lunar eclipse of the sun
18. Enter/exit total lunar eclipse of the sun
19. Enter/exit eclipse at specified longitude (assume box center)
20. Enter/exit sun interference

21. Enter/exit station keeping box
22. Enter/exit station keeping box center
23. Time of closest passage
24. Enter/exit "close pass window"
25. Pre/post maneuver related information

Each template could be configured to meet requirements of a different customer. The user named output file will be delivered to the display function for standard handling (available for hardcopy, plotting, manipulation, or electronic delivery).

It shall be possible to convert the units being displayed to another element set. This will be available in two steps. A global change will convert all numbers displayed. The second option will be a more specific, but temporary conversion where the user will be able to highlight a select group of parameters and convert only those specific parameters. Whenever the function is exited, or even if a new screen is selected, the parameters will revert to their original system of units.

Antenna predicts and pointing data are a direct output of ephemeris generation. Once the ephemeris file is available to the display function, antenna predicts and pointing data can be computed and output as desired.

The processing of this function should be such that once an ephemeris set is generated the user can, at a later time create a new events template without having to setup and rerun the entire ephemeris calculations and redo all of the existing templates associated with that ephemeris.

3.2.3 MANEUVER PLANNING & EVALUATION FUNCTION

3.2.3.1 General

Planning and evaluation capability for East-West (EW), North-South (NS), station keeping, relocation and de-orbit maneuvers will be accomplished by this function. This function will also accomplish all fuel tracking requirements.

3.2.3.1.1 Astra Function 4 Requirements

1. Computation of the velocity increments and times of impulsive and continuous maneuvers for more than one full station-keeping cycle (period between two longitude/eccentricity control maneuvers) in a single run. This implies that the longitude/eccentricity control maneuvers shall be computed considering the cross-coupling components of the inclination control maneuvers as well as the orbit perturbations generated by other maneuvers like spin control maneuvers for spinners.

2. Support different cycles for inclination control (e.g. 4 weeks) and longitude/eccentricity control (e.g. 2 or 3 weeks).

3. Support the following inclination control strategy: determination of the inclination control direction minimizing the propellant consumption.

4. Support the following eccentricity control strategies: (1) Sun-pointing perigee strategy with possibility to specify the control circle center and radius and to bias the eccentricity targets, (2) user's input of end of cycle eccentricity targets.

5. Computation shall ensure that the following constraints are satisfied: maximum burn duration, maneuver time constraints.

6. Computation of re-location to other orbital positions based on some or all of the following inputs: re-location start date, duration, new orbital position and maximum mean eccentricity during the drift.

7. The output plots shall show the propagated longitude and eccentricity elements supposing nominal maneuver performances as well as the predicted 2-sigma maneuver performance uncertainties (along the three Delta-V components).

3.2.3.1.2 Astra Function 5 Requirements

Compute station-keeping maneuvers for all operational modes supported by the satellite design. In particular, following requirements shall be satisfied:

1. Compute all thruster actuations required to achieve a specified velocity increment at a specified time and according to a given maneuver mode, and to generate a corresponding maneuver message ASCII file containing all information necessary to generate the maneuver commands list.

2. In order to minimize the number of manual inputs, the application shall retrieve information from the historical maneuver database (e.g. get initial masses), from the real-time systems to get the required telemetry inputs (e.g. pressure, temperature) and from satellite platform specific databases (e.g. calibration factors). The computation results shall be saved in the historical maneuver database.

3. Perform all checks required to ensure compliance with maneuver constraints (maneuver duration and time, maximum thruster on-time (number of pulses at given pulse-length), maximum Delta-V for chose mode), with attitude control constraints (e.g. sensor interferences, sensor transitions, angle to Sun for yaw computation, spin speed limits (if applicable)) or other power/thermal constraints (e.g. eclipses), or day-of-the week limitations.

3.2.3.1.3 Astra Function 6 Requirements

Maneuver calibration supporting following requirements:

1. Calibration of the Delta-V thrusters based on computed Delta-V and assessed Delta-V resulting from orbit estimation of past maneuvers. The application shall support data modeling based on Fourier fit versus parameters to be selected by the user (e.g. burn center time, solar array angle, date...), as well as the generation of plots and, after user confirmation, the update of the calibration table

2. The function shall support all the maneuver modes (primary and backup modes) and determine dedicated calibration factors for each mode.

3. Model the cross-coupling components of the station-keeping maneuvers and update the cross-coupling table in consequence.

3.2.3.1.4 Astra Function 7 Requirements

Maneuver reconstruction (based on the thrusters telemetry) supporting following requirements:

1. Based on the maneuver telemetry data file generated by the real-time system the application shall estimate the Delta-V components, the fuel used, the CG position, the Isp and thrust per thruster for all maneuvers and thruster firing events, as well as the momentum integrated over the maneuver.

2. Be able to reconstruct all maneuvers since satellite hand-over with minimum input from the users.

3.2.3.1.5 Astra Function 8 Requirements

Historical maneuver database supporting following requirements:

1. Store all data resulting from maneuver computation and reconstruction.

2. Support following functions: maneuver creation and update, retrieve of maneuvers using parameters (e.g. maneuver mode, thruster used...) or time interval, print maneuver report.

3. Export of data into spreadsheets to allow further processing (e.g. computation of accumulated thruster firing), data modeling and plotting of key parameters.

3.2.3.1.6 Astra Function 9 and 10 Requirements

Propellant lifetime prediction satisfying following requirements:

1. Estimate the nominal as well as 3-sigma low satellite propellant lifetime.

2. The error propagation shall include errors on following components: propellant loading, transfer orbit consumption, mixture ratio, thruster Isp, Delta-V, attitude control.

3. Support all the maneuver modes, as well as following events: de-orbit maneuvers, re-location, re-pressurization events, tank switches as well as attitude control related maneuvers.

4. The thruster thrust and Isp shall be calibrated based on past maneuvers stored in the historical maneuver database.

5. Estimate the remaining propellant in the tanks, based on propulsion subsystem telemetry information.

3.2.3.1.7 Astra Function 11 Requirements

Satellites co-location supporting following requirements:

1. Support following co-location strategies: longitude separation, eccentricity and inclination separation.

2. Support computation of following information/predictions: inter-satellite distance and separation along the radial, tangential and normal components, inter-satellite sensor interference and RF signal shadowing. For each prediction, the application shall generate both tables and plots.

3.2.3.1.8 Astra Function 12 Requirements

Integrated Monitoring and Alarm system supporting following functions:

1. Provide near-real time orbit determination functionality to continuously perform orbit estimations, evaluate maneuver performance (autonomous selection of the tracking interval) and evaluate the OD quality.

2. Provide near-real time co-location monitoring system to check the station-keeping and co-location parameters for the next 48 or 96 hours.

3. The Alarm system issues alarms (email, message to portable phone)

4. Combined with the event predict tool, the system provides also continuously orbital control related data to other applications or to the Intranet and Internet.

3.2.3.1.9 Astra Function 13 Requirements

Satellite momentum propagation and control utility satisfying following requirements:

1. Estimation of the satellite external torque based on satellite telemetry values.

2. Propagation of the momentum through maneuvers.

3. Computation of the momentum targets for dedicated momentum control maneuvers or as part of station-keeping maneuvers. Display on a plot of the results with possibility to bias the targets.

3.2.3.2 User Interface Requirements

1. Upon entry to the maneuver planning function, the user will enter the type of maneuver to be planned (NS, EW, drift and eccentricity, station change or fuel tracking).
2. For each maneuver type, this function will offer for selection the various thruster combinations appropriate to that maneuver. If only one set is appropriate, that set will be selected as default, and the next step in the process initiated.

3. Thruster combinations are to be maintained in a database with the capability of being edited, either manually, or automatically, and additional thruster combinations added, or deleted as necessary.
4. The user will have the option of entering the day of the maneuver either manually, or by selection from a plot of appropriate parameters (longitude vs. time for east west and drift and eccentricity maneuvers, inclination vs. right ascension of ascending node for north south maneuvers).
5. It shall be possible to change the start date of a maneuver.
6. This function will use the ephemeris indicated by the user to determine satellite position and drift rate at the epoch chosen for the maneuver.
7. The user will have the ability to indicate, either manually, or graphically, the target of the maneuver. Doing this will determine the burn duration, and, for NS maneuvers, the time of the burn. For manual entry, entering a target position will yield a burn duration, or entering a burn duration will yield a position.
8. For NS maneuver, the time of day for the maneuver will default to the time that corresponds to the Mean Annual Drive (MAD). The time shall be adjustable, either by manual entry, or graphically.
9. For EW maneuvers, the time of the burn is determined by manual, or graphical entry. For graphical determination of the maneuver time a polar plot of eccentricity vs. argument of perigee is required. This plot should be "filtered" to provide a clear representation of the data.
10. Drift and eccentricity maneuvers are a special case of EW maneuver planning requiring multiple burns spaced several hours apart. In almost all circumstances, these will be two burns, twelve hours apart, however the timing must be selectable. The goal of the drift and eccentricity maneuver is to perform a large correction to eccentricity and time of perigee, with a small change in drift rate, without exiting the boundaries of the station-keeping box. The total burn duration determines the change in the eccentricity "vector", while the difference between the burn durations will determine the drift rate.
11. The targets for the drift and eccentricity should have manual and graphical entry capability.
12. Station change maneuvers are a special case of drift and eccentricity maneuver. In this case, a new station location is the goal, while maintaining eccentricity control.
13. The user will input the target longitude either manually or graphically, and the lapsed time, or arrival date. The time of the burns will either be entered manually or graphically as is done for the drift and eccentricity. This function will output burn durations, and fuel estimates. Stopping the satellite at its new longitude is considered part of the station change maneuver planning.
14. For all maneuver plans, this function will output a maneuver date and time, burn duration and orbital elements that reflect the post maneuver orbit. These elements shall be directly usable to run an ephemeris to validate the maneuver plan.
15. It shall be possible to plan multiple maneuvers with the input of a single ephemeris. This can be either different burn durations, or maneuver time for modeling purposes. Or the maneuvers can be of different types, such as a NS followed by and EW.
16. An estimate of fuel consumption shall be a part of all maneuver planning.
17. Fuel usage shall be estimated based on relevant data in the database, and burn durations from the maneuver planner output.

18. Fuel accounting shall be part of this overall function.
19. This function shall use as input a thruster data file indicated by the user, or manual entry of burn durations of individual thrusters. The fuel calculations shall take into account burn durations and pulse widths that affect ISP and thrust.
20. The fuel accounting data shall be kept in a file for archiving. This file shall be accessible by other programs for maneuver evaluation, and updating of the spacecraft database.
21. Entries to this file shall be on a per maneuver basis, with fuel consumption for individual thrusters, consumption for this maneuver, and fuel remaining.
22. The user will indicate an ephemeris file for use in evaluating the current maneuver plan.
23. Thruster data information, including thruster pulse widths and burn time shall be used as part of the input to the maneuver evaluator.
24. With the input of the maneuver plan, thruster data file and post orbit determination ephemeris, the evaluator will calculate, the change in drift rate, vector change in eccentricity, and vector change in inclination. With these values the evaluator will determine fuel consumption, thrust, "thrust values" (change in drift rate per second of burn, change in inclination per minute of burn, change in eccentricity per second of burn).
25. It shall be possible to use the same ephemeris file for both maneuver planning, and maneuver evaluation, i.e. "ephemeris file A" which followed "maneuver A" can be use to evaluate the results of "maneuver A", and to plan the next maneuver, "maneuver B".
26. The evaluator shall have the capability of determining corrections to the thruster parameters used by the maneuver planner, and updating these values in the database at user request.
27. The maneuver evaluator shall generate a report that includes maneuver type, date and time, thruster burn durations, pre maneuver elements and actual post maneuver elements, change in drift rate, eccentricity, and inclination, thrust achieved, fuel consumption, and any database parameters updated.

3.2.3.3 Input Requirements

1. An ephemeris to determine satellite position and velocity at maneuver time.
2. Maneuver type.
3. Spacecraft parameters such as mass, fuel pressure and mass, thruster performance parameters.
4. Thruster selection.
5. Maneuver epoch (date and time, or target).
6. Maneuver burn duration, or target.
7. Plots of longitude vs. time, eccentricity vs. time of perigee, and inclination vs. right ascension of ascending node.
8. Calculation of Mean Annual Drive.
9. Thruster data file containing burn durations of individual thrusters.
10. For station change maneuvers the time allowed to achieve station.

3.2.3.4 Output Requirements

1. A post maneuver ephemeris is input for maneuver evaluation.

2. A maneuver plan providing date and time of maneuver, thrusters to be used, burn duration, orbital elements at the epoch of the maneuver for the pre and post maneuver conditions, expected fuel consumption. In addition, for Spacebus 2000 maneuver planning the appropriate sun sensor to be used during the maneuver will be part of the output.
3. An ephemeris for validation of the maneuver plan and as input for the orbit determination function.
4. Fuel consumption file to document fuel consumption by maneuver, for each thruster and a total remaining.
5. Updates to the database for mass and fuel remaining.
6. A maneuver evaluation report that includes maneuver type, date and time, thruster burn durations, pre maneuver elements and actual post maneuver elements, change in drift rate, eccentricity, and inclination, thrust achieved, fuel consumption, and any data base parameters updated.
7. The ability to update the relevant thruster parameters (i.e. flow rate, thrust, Isp, etc.) in the database with the values determined by the evaluator.
8. The Display function shall be capable of plotting those outputs from the maneuver planner/evaluator that are appropriate to plotting, such as orbital elements, fuel remaining, thrust, etc.

3.2.4 PLOTTING FUNCTION

3.2.4.1 General

This option will take selected data provided by other functions within this system and provide an on-screen or hardcopy graphical presentation of the data, or interface with a third-party graphics package to provide similar functionality.

3.2.4.2 User Interface Requirements

A setup screen will enable the user to select the data to be plotted, define the plotting parameters and select the output device. A standard Windows user interface, including left/right mouse button operation, shall be implemented.

3.2.4.3 Input Requirements

1. Database information or file names containing the data to be plotted
2. Selection of plotting coordinates
3. Maximum and minimum values of the parameters along both axes (for Cartesian plots)
4. Maximum radial component for polar plots
5. Type of filtering of data
    a. averaging
    b. sampling (specify 1/n)
    c. bounded
6. Bounds (limits) within the set extremes
7. Orientation of plots
8. Automatic scaling of output

SES AMERICOM

SES ASTRA

3.2.4.4 Output Requirements

1. Plot single traces of parameters
2. Plot multiple traces of parameters with like units (but not necessarily of identical parameters on the same plot.
3. Plots will be available in either Cartesian or polar coordinates
4. Ability to modify maximum/minimum values on plot, or select autoscale
5. The user will select if a hardcopy plot is desired (Screen plot will always display)
6. For longitude plots, the user will have the capability to insert on demand (or remove on demand) the ascending and descending nodes.
7. Allow different scaling of unsplined longitude plots. The user will have the option of selecting angle data "wrapped" to either –180 to 180 or 0 to 360 degrees.
8. The following parameters will be available for plotting
    1. universal time
    2. ephemeris time
    3. semi-major axis
    4. eccentricity
    5. inclination
    6. right ascension of the ascending node
    7. argument of perigee
    8. mean anomaly
    9. altitude
    10. latitude
    11. longitude
    12. speed
    13. flight path angle
    14. heading
    15. magnitude of geodetic acceleration vector
    16. elevation of the geodetic acceleration
    17. azimuth of the geodetic acceleration
    18. ephemeris time of perigee
    19. ephemeris time of the ascending node
    20. eccentricity at perigee
    21. inclination at the ascending node
    22. equinoctial longitude of perigee
    23. local solar time of perigee
    24. filtered longitude (daily oscillations removed)
    25. filtered eccentricity (daily oscillations removed)
    26. filtered inclination (daily oscillations removed)
    27. filtered equinoctial longitude of perigee (daily oscillations removed)
    28. filtered local solar time of perigee (daily oscillations removed)
    29. K (esin(arg of perigee + raan)
    30. H (ecos(arg of perigee + raan)
    31. Q (tan(inclination/2)*sin(raan)
    32. P (tan(inclination/2)*cos(raan)
    33. mean longitude
    34. orbital period
    35. x position ECI
    36. y position ECI
    37. z position ECI

SES AMERICOM

SES ASTRA

38. x velocity ECI
39. y velocity ECI
40. z velocity ECI
41. x acceleration ECI
42. y acceleration ECI
43. z acceleration ECI
44. x position ECF
45. y position ECF
46. z position ECF
47. x velocity ECF
48. y velocity ECF
49. z velocity ECF
50. x acceleration ECF
51. y acceleration ECF
52. z acceleration ECF
53. east west drift
54. north south drift
55. east west acceleration
56. north south acceleration
57. maximum longitude
58. maximum latitude
59. minimum longitude
60. minimum latitude
61. magnitude of position vector
62. spacecraft right ascension
63. spacecraft declination
64. magnitude of the velocity vector
65. angle down of velocity vector from radius vector
66. azimuth of the velocity vector
67. magnitude of the acceleration vector
68. right ascension of the acceleration
69. declination of the acceleration
70. F (esin(arg of longitude)
71. G (ecos(arg of longitude)
72. mean motion (angular rate of the mean anomaly)
73. argument of longitude (raan + arg of perigee)
74. chi (tan(inc/(1+cos(inc))*sin(raan))
75. psi (tan(inc/(1+cos(inc))*cos(raan))
76. acceleration magnitude due to solar pressure
77. acceleration magnitude due to solar gravitation
78. acceleration magnitude due to lunar gravitation
79. acceleration magnitude due to geoid gravitation
80. right ascension of the sun
81. declination of the sun
82. slant range to the sun
83. right ascension of the spacecraft to sun vector
84. declination of the spacecraft to sun vector
85. range from spacecraft to the sun
86. right ascension of the moon
87. declination of the moon

88. slant range to the moon
89. right ascension of the spacecraft to moon vector
90. declination of the spacecraft to moon vector
91. distance from spacecraft to the moon
92. local solar time of spacecraft
93. potential energy
94. kinetic energy
95. total energy
96. spacecraft mass
97. spacecraft slant range versus time from a given earth station
98. azimuth to spacecraft versus time from a given earth station
99. elevation to spacecraft versus time from a given earth station
100. MAD angle

3.2.5 ANTENNA FUNCTIONS

3.2.5.1 General

This is a collection of utilities that facilitate the computation of various antenna related information. Included are:

1. Antenna predictions
2. Look angles
3. Conjunctivity predictions

3.2.5.2 User Interface Requirements

A set-up screen will enable the user to specify the earth station antenna to be used in all computations, as well as the celestial coordinates of the target(s).

3.2.5.3 Input Requirements

1. Earth station antenna coordinates – either by manual input of station name, longitude, latitude, and altitude or by entering the earth station database and selecting a set of existing stored coordinates. If manual input is selected, the program will ask if the new coordinates are to be added to the database.

2. Location of celestial body – either manually input, from an existing ephemeris file or from the public domain accessible database of geostationary satellites. A satellite can be specified using only a longitude, in which case inclination and drift are assumed to be zero.

3. Start and stop times for the data generation.

4. Specification of the interval between successive coordinates.

5. If the sun or moon is specified as the target, their locations for the specified time period are to be calculated internally.

6. For conjuctivity computations, the user must also specify the window (+/- degrees) within which commanding is prohibited.

3.2.5.4 Output Requirements

For antenna predicts, a digital antenna drive file will be generated that can be read by the antenna control units at the TT&C sites.

The look angle function will generate an ASCII output of azimuth and elevation versus time, given the coordinates of an earthbound antenna and a celestial body. If the coordinates of the space-based object are input manually, a single set of pointing angles will be generated. If an ephemeris file is used to define the location of the space-based object, a series of pointing angles will be generated for the specified period of time and at the specified interval.

The conjunctivity option will output the date/time of the conjunctivity window along with the size of the window and an identifier of the other satellite.

3.2.6 DATABASE FUNCTION

3.2.6.1 General

The database serves as a repository for pertinent information, which is used on a recurring basis by the system. A single active database will be maintained but shall be divided into logical areas using standard schema techniques. All system users will access the same active database.

The database can be thought of as being divided into segments according to the type of information contained within the segment. Some of those segments are satellite specific, TT&C site specific, global parameters, earth station coordinates, general satellite locations, sun & moon orbits etc.

The concept of scratch areas and best-knowledge database shall be used for all database parameters. Data is saved or updated in the best-knowledge area only when explicitly selected by the user. This area shall reflect the best-known present situation of the satellite fleets. Scratch areas are used to plan and adjust operational situations until such time they are deemed to be elevated to best-knowledge status.

The database will be accessible from any terminal/workstation/computer, which is running the satellite orbit management software. When first started the system should interactively ascertain which satellite the user wants to work with and load the appropriate database information.

The database shall have the ability of handling a minimum of 16 earth stations and 50 satellites.

The system shall prevent more that one user updating the best-knowledge or identical scratch areas during the same sessions. Error messages shall be provide to the appropriate users alerting them of the potential problems when such situations arise.

3.2.6.2 Input Requirements

The database will contain items that are updated as a normal course of operations by other functions of this system, such as range biases, fuel remaining, orbital elements etc.

A method will be available to examine the contents of the database and to make manual updates. The user shall have the ability to specify the default units in which various parameters will appear. These units will be changeable in real-time, but will default to the database units each time the program is exited.

3.2.6.3 Output Requirements

Display of the database values to the screen and/or print segments of the database.

3.2.7 DISPLAY FUNCTION

3.2.7.1 General

This function shall provide a general purpose file management capability within the application. It allows the user to rename, copy, move, edit and delete satellite data files without having to exit the program.

In particular this would be used to rename generated output files to give information about the contents of the files.

Ephemeris, antenna predicts and other report files could be shortened or merged without having to rerun them.

A method to construct event templates shall be provided. These templates allow for pre-canned report formats to be identified that simply need to be 'run' by the system so as to fill in the most recent variable values contained in the template.

3.2.7.2 User Interface Requirements

Within a windows-like environment, this function will provide a user interface similar to Windows Explorer.

3.2.7.3 Input Requirements

All application related data files on local and server hard drives.

Template files are inputs that depict the report format and variables.

3.2.7.4 Output Requirements

All event templates will be available for transmittal and/or manipulation.

Reports may be produced as a single template or constructed out of several existing templates to provide the needed data.

3.3 PERFORMANCE REQUIREMENTS

The system shall be capable of handling up to 25 simultaneous users without noticeable system performance degradation given expected user workload requests.

The system shall be capable of providing all functions for a maximum of 50 active satellites.

3.4 LOGICAL DATABASE REQUIREMENTS

The information maintained within the database shall be all satellite parameters needed to provide the functionality identified in other sections of this document.

Every satellite shall have its own, complete set of data associated with that satellite in the database.

The database system shall be designed to allow for daily delta backups and complete weekly backups that minimize system down time.

The database shall use RSA Key Pair Authentication or equivalent security techniques.

Minimum database storage shall be 50% larger than the worst-case calculated maximum database size.

Users shall be able to easily access the information in the orbital databases: both scratch and best-knowledge areas.

User shall be able to easily access and edit the information pertaining to collocation configurations.

3.5 DESIGN CONSTRAINTS

System functionality shall be accessible from any PC that is connected to the corporate network. This includes use of general Internet access points (i.e. Wi-Fi) using existing corporate LAN security access methods.

3.6 SOFTWARE SYSTEM ATTRIBUTES

3.6.1 RELIABILITY

A full Acceptance Test will be completed prior to delivery and installation of the system. This test will establish that the delivered system meets all requirements contained within this specification.

3.6.2 AVAILABILITY

This system shall exceed 99.999% ('five nines') availability per year. Scheduled upgrades or maintenance does not count against this availability requirement.

3.6.3 SECURITY

This system will be located behind corporate firewalls and operate on the already isolated and secure TT&C network. For these reasons, the following security measures are adequate for this system:

Root passwords

Admin passwords
User passwords

To ensure that all passwords used to control access to this system and not easily guessed, the following shall be implemented:

1. Passwords shall expire every 30 days, requiring users to create new passwords.
2. The last six passwords cannot be re-used.
3. Passwords must be a minimum of eight characters in length and incorporate three of the following characteristics:
   - Any lower case letters (a-z)
   - Any upper case letters (A-Z)
   - Any numbers (0-9)
   - Any punctuation or non-alphanumeric characters found on a standard ASCII keyboard (! @ # $ % ^ & * ( ) _ - + = { } [ ] : ; " ' | \ / ? < > , . ~ `)

3.7 OTHER REQUIREMENTS

3.7.1 GUI INTERFACE

Most GUI panels should be contain two levels of configuration options:

1) Parameters frequently changed by the user

2) "Advanced" parameters.

3.7.2 TRANSACTION INFORMATION

The history of user initiated actions will be maintained for a period of the most recent 4 weeks.

This data will include:

User entered information
The user's IP address
A time stamp of the action
Login and logoff time

4 APPLICATION GROWTH

This section describes functionality that may be required in the future, but has no absolute requirement at this time. A systems design that allows for such expansion in the future would be considered the better approach.

4.1 HIGHLY INCLINED ORBIT

The ability to use the OD, ephemeris propagation and maneuver planning functions defined in earlier sections for satellites in a highly inclined orbit. A highly inclined orbit is defined as an inclination angle greater that 10.0 degrees.

4.2 LOW EARTH ORBIT

The ability to use the OD, ephemeris propagation and maneuver planning functions specified for satellites in a low Earth Orbit (LEO).

4.3 SPINNER FUNCTIONALITY

(Similar to Astra Function 14 requirement)

For a 2 axis stabilized satellite (i.e. spinner), the following functions shall be supported:

1. Attitude data processing
2. Attitude estimation and propagation, including the estimation of sensor related biases and precession
3. Attitude and Spin control maneuvers planning. In particular, the application shall support the planning of all the attitude control maneuvers considering the orbital maneuvers schedule. A spin maneuver planner based on spin speed propagation through maneuvers shall be provided also.

5 APPENDIX A: USER INTERFACE PROTOTYPE

This section outlines potential approaches to the Graphical User Interface (GUI) for the NG-SOCS system. Satellite orbit control parameters need to be displayed to the user in a special reference frame so that a quicker and better understanding of the satellite motion can be obtained when reviewing the data.

All GUI windows will be designed by the system developers and presented at PDR for stakeholder review and acceptance. This appendix is intended to give guidance to the system developers so as to foster a basic understanding of the anticipated user interface.

Examples in this section are from FLD, PC-SOCS or STK screen shots.

The system GUI shall have a Windows look and feel, with standard and custom controls as needed. The GUI should be user-friendly, which means it should be simple and familiar to all of the users. It has consequently been broken into several screens, each of which is dedicated to a single task to keep the risk of information overload down. Standard Windows-style interaction is maintained with textboxes, combo boxes and buttons. The color scheme will avoid eye-straining combinations such as red and green next to each other. There will be no more than two fonts on any given screen. Both the color scheme and the fonts will be consistent among all screens.

5.1 MAIN PAGE

An example of the main window for Satellite Analysts in shown in figure A-1. Three general panes are employed in this approach, namely the worldview, the application tool bars and the scenario pane. The user can select graphical items or choose from menu items in order to drill down into the underlying data.

SES AMERICOM

SES ASTRA

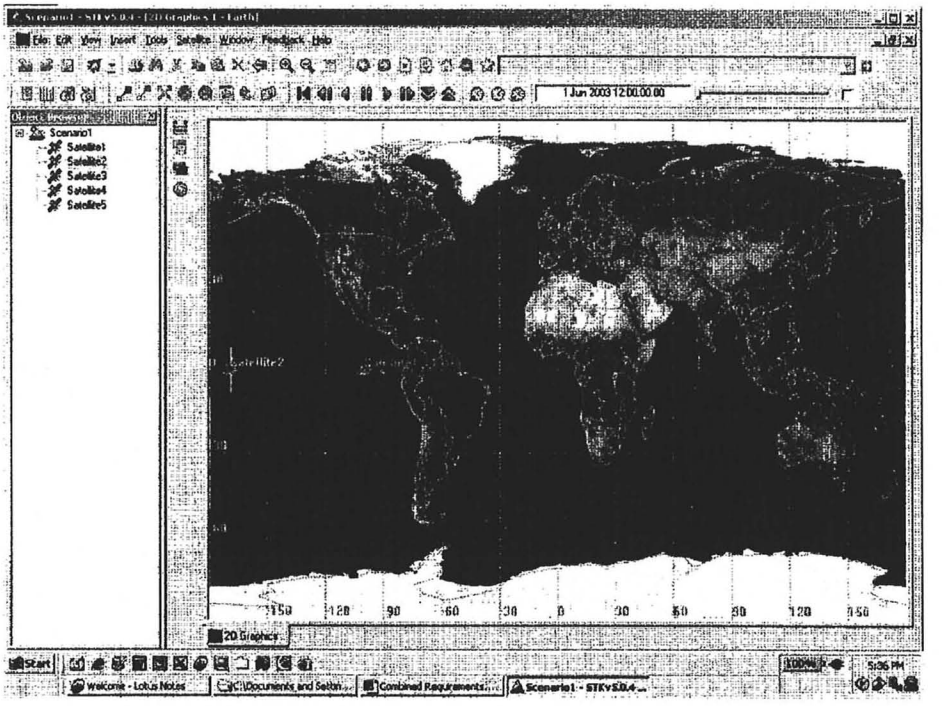

Figure A-1, Example of a Main Satellite Analyst Window

5.2 DATA DISPLAYS

An example of a data display is shown in figure A-2. Combo boxes are used to allow the user to save changes to the default satellite ephemeris data. The window displayed is for 'Satellite 1' (highlighted in the scenario pane) and would be populated with the most recent data from the database.

SES AMERICOM
An SES GLOBAL Company

SES ASTRA

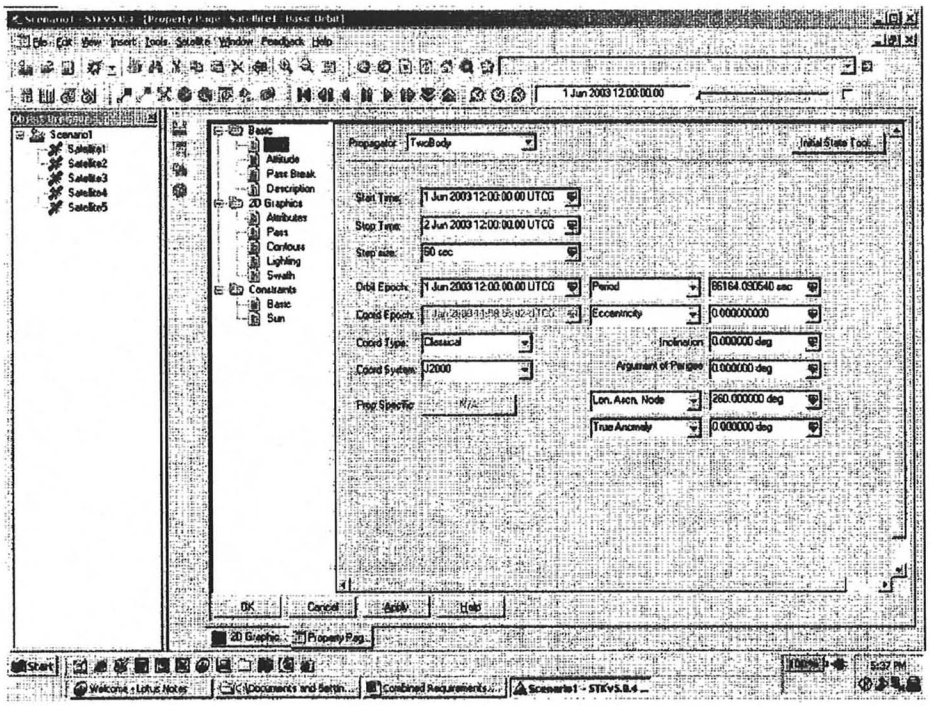

Figure A-2, Example of a Data Display – Satellite Elements

5.3 ORBIT DISPLAY

An example of an Orbit display is shown in figure A-3. Once the satellite(s) orbit is known, the user has the ability to display the data on the map. Multiple satellites can be depicted on the map. Figure A-4 depicts the ability to zoom onto any area of the map.

SES AMERICOM
An SES GLOBAL Company

SES ASTRA

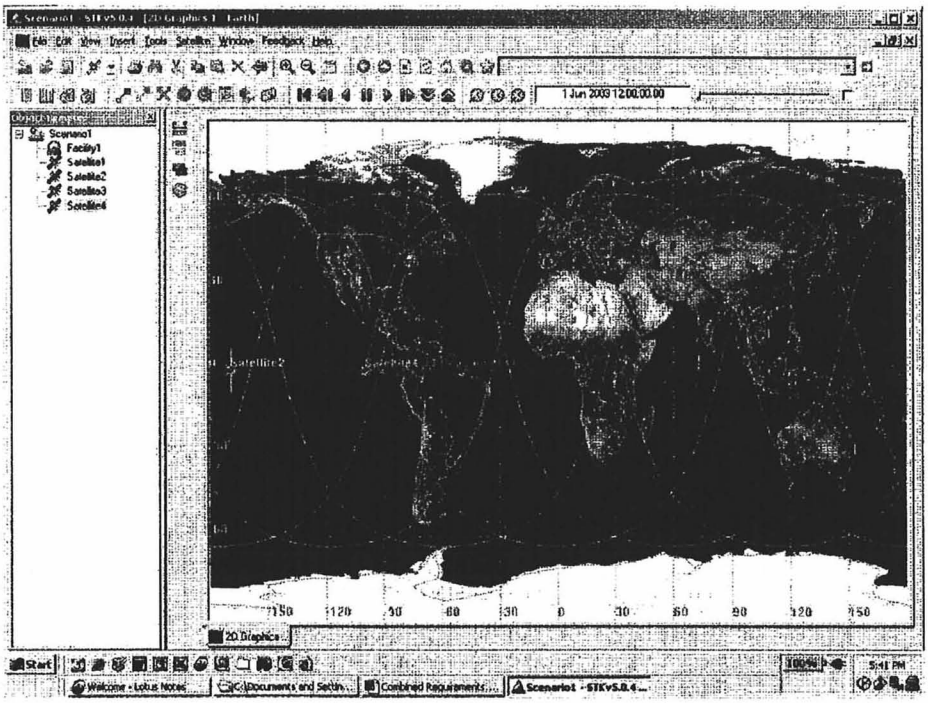

Figure A-3, Example of an Orbit Display

WORKS CITED

1. Bulut, Evren. "Memoire de Travail de Fin d'Etudes" MS Thesis Centrale dLyon, France, 2002.

2. Carr, Nicholas. "IT Doesn't Matter." Harvard Business Review (May 2003): 41-50.

3. Chiang, Chia-Chu. "A Distributed Object Computing Architecture for Leveraging Software Reengineering Systems." Communications of the ACM Volume 22 (May, 2001): 653-657.

4. Elmasri, Ramez, and Shamkant Navathe. Fundamentals of Database Systems. Massachusetts: Addison-Wesley, 2000.

5. Fink, Martin. The Business and Economics of Linux and Open Source. New York: Prentice Hall, 2002.

6. Garkal, David. "Software Architecture: a Roadmap." Communications of the ACM Volume 26 (June 2000): 93-101.

7. Institute of Electrical and Electronics Engineers. "IEEE Recommended Practice for Software Requirements Specifications". IEEE Std. 830-1984 (1998).

8. Logsdon, Thomas. Orbital Mechanics Theory and Applications. New York: John Wiley & Sons, 1998.

9. Markus, M. Lynne. "Power, Politics, and MIS Implementation." Communications of the ACM Volume 26 (June 1983): 430-444.

10. Public Agenda. "Defining the Basic Elements of .NET." Microsoft. January 24, 2003. Retrieved 1 Oct. 2004 from the World Wide Web: http://www.microsoft.com/net/basics/whatis.asp

11. Public Agenda. "Exploring the range of CORBA technology." IBM. 6 July 2000. Retrieved 10 Oct. 2004 from the World Wide Web: http://www-128.ibm.com/developerworks/webservices/library/ws-exploring-corba/index.html.

12. Public Agenda "The J2EE 1.4 Kickoff, Part One." Sun Microsystems. 18 May 2004. Retrieved 2 Nov. 2004 from the World Wide Web: http://java.sun.com/developer/technicalArticles/releases/j2ee14kickoff/.

13. Public Agenda "The Java Servlet API White Paper" Sun Microsystems. Retrieved 2 Nov. 2004 from the World Wide Web: http://java.sun.com/products/servlet/whitepaper.html.

14. Public Agenda. "Sun confirms plans to open source Solaris." CNETAsia News. 2 June 2004. Retrieved 1 July 2004 from the World Wide Web: http://asia.cnet.com/news/software/0,39037051,39181540,00.htm.

15. Public Agenda. "Web service Performance: Comparing Java 2 Enterprise Edition (J2EE platform) and the Microsoft .NET Framework." Microsoft Corporation. 12 July 2004. Retrieved 15 Nov. 2004 from the World Wide Web: http://msdn.microsoft.com/vstudio/java/compare/default.aspx.

16. Roepke, Robert, Ritu Agarwal, and Thomas W. Ferratt. "Aligning the IT Human Resources with Business Vision: The Leadership Initiative at 3M." MIS Quarterly Volume 24 No. 2 (June 2000): 327-353.

17. Sommerville, Ian. Software Engineering. Massachusetts: Addison-Wesley, 2001.

18. Wang, Narbor, Douglas Schmidt, and David Levine. "Optimizing the CORBA Component Model for High-performance and Real-time Applications." Dept. of Computer Science, Washington University, Boeing, NSF Grant NCR-9628218, DARPA contract 9701516, Siemens. 15 Oct. 2004 http://www.research.ibm.com/Middleware2000/WiP_Papers/wang.pdf.

19. Wohlstadter, Eric, and Brian Toone. "A Framework for Flexible Evolution in Distributed Heterogeneous Systems." Communications of the ACM Volume 38 (Sept. 2002): 39-42.

20. Wohlstadter, Eric, and Stoney Jackson. "DADO: Enhancing Middleware to support Crosscutting Features in Distributed, Heterogeneous Systems." Institute of Electrical and Electronics Engineers (March 2003): 174-186.

21. Zwass, Vladimir. Information Systems. New York: McGraw-Hill, 1998.