Design of a Satellite Data Distribution Center

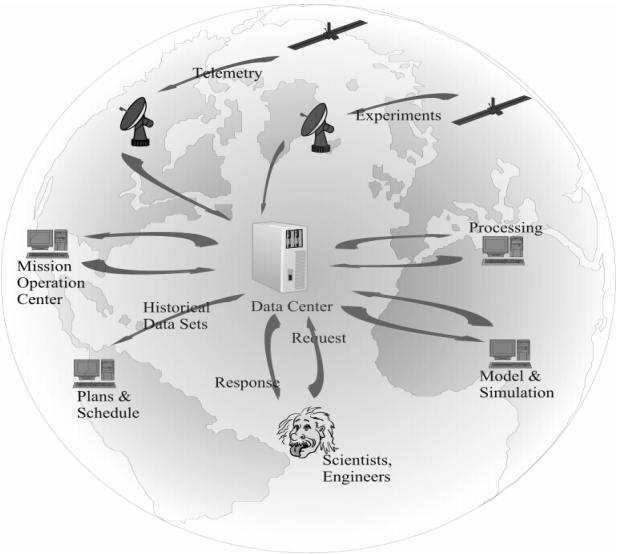
Lance Self
Air Force Research Laboratory
Space Vehicles Directorate
Kirtland AFB, NM
Lance.Self@kirtland.af.mil

Abstract

Designing a data archive and data access capability is an important development area for realizing satellite experiment objectives. The data center must be designed to provide a consistent and stable environment yet at the same time be able to adapt to the inevitable technological changes. This paper discusses such a framework being developed for an Air Force experimental satellite.

Introduction

The Air Force Research Laboratory (AFRL), Space Vehicles Directorate, and the Air Force Office of Scientific Research are co-sponsors of an experimental satellite program. The mission of this program is to demonstrate and validate micro satellite cluster system concepts and enabling technologies.



The principal experimental objectives are to demonstrate sparse aperture sensing from space, qualify ultra-light and low cost spacecraft technologies, formation flying technologies for clustered satellites, autonomous cluster and spacecraft operations, and validate models and simulations of cluster performance and operations.

The System Simulation & Control branch decided to fund the development of a data center to create the services and capabilities that will provide for data preservation, access, distribution, and exploitation. These services and capabilities will support this satellite program's efforts and support the customer base that exploits the experimental results of the program.

Design Guidelines

All data center customers (TestBed, MOC, scientists, engineers, technologists, acquisition community, etc.) share a common need for a stable, secure, and accessible environment from which to re-create and exploit data sets. To satisfy these diverse requirements, the data center is designed to be a central point to manage, secure, and distribute the irreplaceable data sets this program will provide by offering the following services:

- Manage data access for internal and external AFRL customers
- Provide a central archive site for experiments, satellite telemetry, M&S, and other assets
- Provide data access and retrieval services
- Insure data integrity

The data center is oriented toward providing state-of-the-art information services and not toward developing and demonstrating new technology. The functional elements such as the TestBed, MOC, and individual payload experiments are responding to technology objectives and requirements. The data center is responding to the data and information management

needs of these elements in addition to those of the satellite program office and external users interested in the technology that is captured in the data and information maintained by the data center.

Data Center Design

This section discusses the overall design of the Ground Element and specifically how the data center fits within that element. It discusses the design approach the data center is taking, the flow of information between ground element components, where and how the data center ingests data, and finally a deployment model.

The data center has three principal sources of data: TestBed, Mission Operations Center, and on-board satellite experiments. The broader range of data center customers, researchers, scientists, etc., will use data sets collected from those groups to perform further processing, analysis, and experiments. At times, however, these external customers may choose to store data to the data center on an ad-hoc basis.

For purpose of reference, the applications, processes, and the database management system (DBMS) lie within Layer 7 (Application layer) of the OSI model. Within that OSI layer the data center has adopted the model depicted below (Figure 1) to define the internals of where various processes lie and what messages are passed between these layers. This model equally applies toward an E-business as well as what the data center is developing, and is consistent with a modular object-oriented approach.

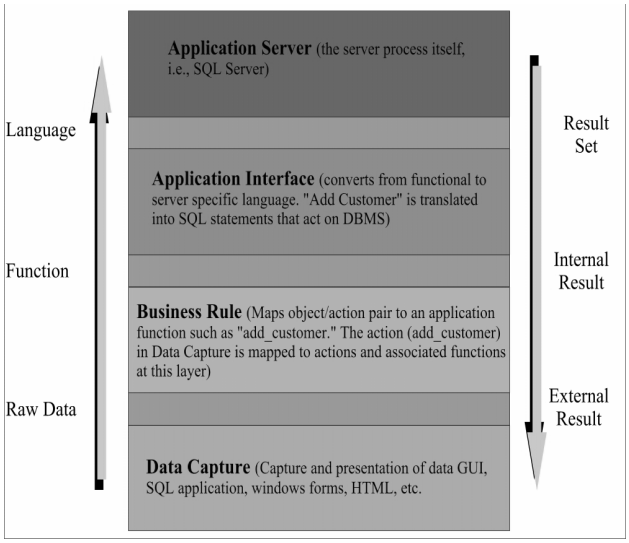


Figure 1: Data Center Design Model

At the bottom of the model is the Data Capture or "user's view" layer. This layer defines how and in what format data is presented to the user. What the user "sees," buttons the user clicks, etc., is invoked at this layer. Everything above this layer should be transparent to the user, that is to say, how a business rule is implemented or what internal process the database engine uses, takes place behind the scenes. Above that is the Business Rule layer that receives raw data and maps an action or event to a pre-defined rule. If a user clicked on the *add_customer* button for example that event is mapped to some function that is brought into play because of that action. Next up in the hierarchy is the Application Interface layer that converts the function, *add_customer* for example, into a language the Application Server layer can implement. The result (Result Set) of that implementation, i.e., the results of a query, are passed back down the layers and presented to the user in the correct format.

This model hides layers, such as a proxy layer, that more definitively describes the process of taking a request from a user and presenting the results of that request. However, for data center development, this model serves the intended purpose since those layers that are absent from this model are, for the most part, embedded in the tools that the data center will use to build the applications, methods, and processes for customer use.

Ground Element

Figure 2 shows how information is channeled between external customers and the ground element, and between ground element components and the data center. External customers that want access to AFRL assets will cross through the AFRL firewall to access servers that are located between the Kirtland and AFRL firewalls. First, before crossing AFRL firewall boundaries, users must have submitted the proper documentation and been assigned a login name and password by AFRL. Once inside the AFRL firewall users can access the two servers that have tools with which to conduct their business. The tools server, *Anasazi*, houses development tools such as RealTime Studio Pro and Perforce. The other server, *Marmot*, houses the Data Services Interface that allows users access to the data center server, *Hoahu*.

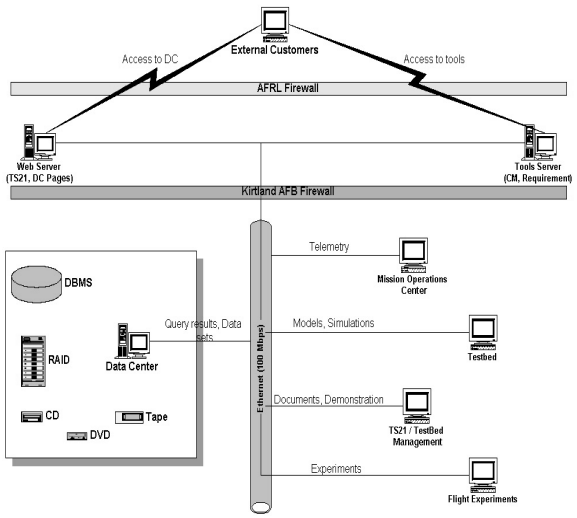


Figure 2: Information Channels

Beneath the Kirtland firewall are the Mission Operation Center and the TestBed, which are connected to the DASL LAN (Figure 3). This is an internal Ethernet sub-network, running at 100 Mbps, that creates an uninterrupted means for the TestBed and MOC to exchange data without disruptions (such as email) from the AFRL network. The DASL LAN links physical components located within the DASL room, where most of the equipment is located, and can be switched to the external AFRL network using the DASL network switch.

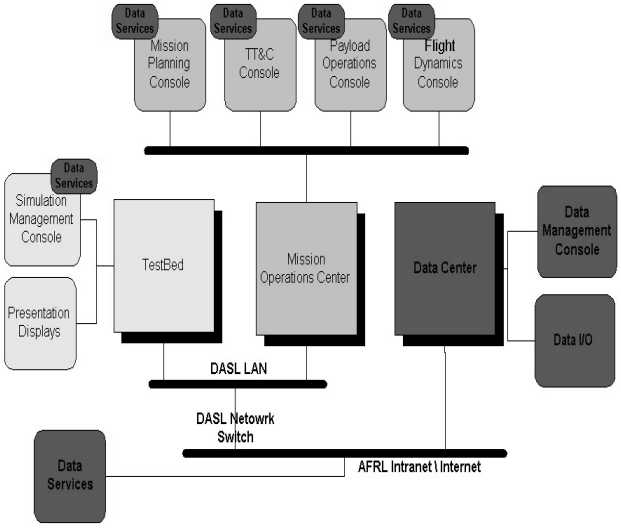


Figure 3: Ground Element Architecture

The data center hardware and software architecture is built around a client\server framework based on COTS and GOTS components. The data center primarily resides on the AFRL Intranet \ Internet network but can be switched, using the DASL network switch, to the DASL LAN. This configuration allows for the exchange of data from ground element components to the data center, and from the data center to external customers, using the AFRL Internet network.

The Data Services Interface is connected to the AFRL LAN. All data exchanged between the data center and customers use the Data Services Interface regardless of their physical location. If a ground element component needs to store satellite telemetry files to the data center, or if a customer in Boston wishes to retrieve experiment data, they would both use the Data Services Interface to conduct their business with the data center. One subtlety to note is if a MOC console (Mission Planning, Flight Dynamics, etc.) or TestBed console wishes to use the Data Services the DASL Network switch must be in the correct position.

Deployment

The Deployment diagram (Figure 4) shows how the hardware and software that make up the system are configured and distributed from a data center perspective. The large blocks are *nodes* that are used to depict a physical element, and within the nodes are components.

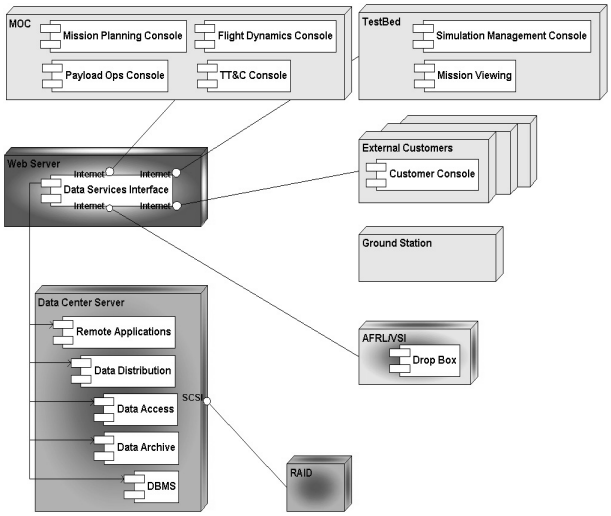


Figure 4: Deployment

Each of the gray nodes (MOC, TestBed, External Customers, Ground Station) is a major source and/or sink of data sets that physically exist. Each, with the exception of the Ground Station, has a direct Internet link to the DSI via the web server. The AFRL/VSI node houses the Drop Box component that is also linked to the DSI. Using the Web Server these various nodes and components gain access to the data center and the components housed within.

Enabling Technologies

Microsoft .NET (dot net) is Microsoft's latest push at a platform for XML web services that is their vision of connecting people and information. This platform enables the use of XML based applications and processes to work together to perform a task or service to the consumer while using the HTTP protocol as the vehicle to transport these applications and processes. Conceivably, these applications will run on any smart device such as phones, handheld computers, and laptops, and is the next generation of Internet based computing.

The .NET is being designed to let programmers write XML web services instead of focusing on writing stand-alone applications. These web services will be interacting with other web services to form a loosely coupled "constellation" of software programs dependent on XML as the communication venue. To achieve this Microsoft has developed tools that permit the programmer to develop these "constellations", primarily Visual Studio.NET.

.NET is comprised of a library of objects that the developer uses to create applications in a similar manner that the old Windows API did such as creating windows, dialog boxes (specialized window), threads, etc. The .NET also adds features of connecting to databases or the Internet using the objects contained in the library. The second area .NET provides is the environment to run these applications via the .NET Runtime (a.k.a. Common Language Runtime or CLR).

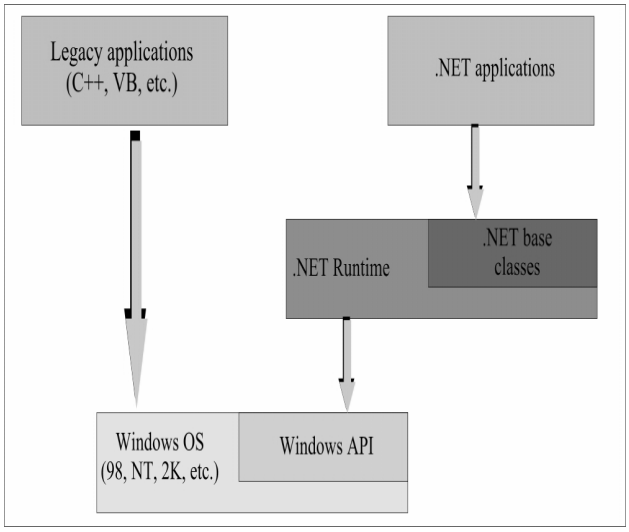


Figure 5: .NET Environment

Visual Studio.NET is Microsoft's latest IDE that focuses on creating XML based web services. Included are a number of tools for developers and modeling tools that support UML. Of particular interest to the data center are the languages supported (C#, Visual Basic.NET, Java), deployment of applications over the Internet, and the database modeling support.

Using any language to write deployable applications permits developers to use that which they are most familiar. Visual Basic programmers can write applications in that language while C# programmers can use their experience in C#. The need to learn a new language is eliminated which increases programmer productivity. The data center has a mix of developers with such varied programming backgrounds, and this is expected to be the future case.

Writing deployable applications will benefit this satellite program in that data center staff as well as any others can write programs. Deploying applications to customers over the web will let them use the experience of those developers that are closest to the problem domain. This should decrease the time for Verification & Validation tests and allow sharing of data (including test results) over the Internet. .NET is using XML so these applications will be able to be used on any system (UNIX, PC) over any browser.

Conclusion

A secure, stable operating environment is an important link when dealing with experimental satellites. Researchers, scientists, and engineers need assurance that data integrity is maintained throughout the satellite life-cycle. However, the environment should also be able to adapt to technological advances, which was a driving force behind this development effort.