Paper 237-25

Enterprise Application Integration: Integration and Utilization of SAS Products

Matthew K. Hettinger, Mathet Consulting, Schaumburg, IL

ABSTRACT

The enterprise today requires many applications (from multiple vendors) concurrently running in parallel on multiple platforms. Managing these applications and the resources these applications require sharing data and meta-data across the enterprise, and integrating these applications for cross functionality has been until recently intractable. This is due to reasons such as inherent differences in functionality, differences in program languages, lack of standard communication methods. With the advent of CORBA, JAVA and Enterprise Application Integration frameworks, these issues are beginning to be addressed with some success. This paper presents an Enterprise Application Integration framework based on an N-tiered, distributed, O-O architecture and allows for the development, testing and deployment of custom applications as well as for application integration of third party products. Five unique architectures are introduced within this framework. When operating together these architectures provide a complete mechanism for application integration that is scalable, extensible, and robust. In addition, it is shown how various SAS products such as SAS Warehouse Administrator, Data Miner, Enterprise Reporter, STAT, Graph, BASE, CONNECT, SHARE, ACCESS, EIS, and SAS AF can be incorporated into different components of the framework. This paper is intended for advanced audience.

INTRODUCTION

With the advent of CORBA, the vision that "the network is the computer" is a step closer. CORBA allows objects created in different languages to communicate. Having this communication channel is a necessary structural requirement for processes to share information. In addition, CORBA makes the network transparent. That is, the network is abstracted so that the working unit becomes objects inside processes and not network domains, machine on the network, or domain applications. Conceptually, three levels of network communications can be identified. First is the physical network (inter- and intra-networks). The nodes of the network are computers. The second level (sub-net) is concurrent processes running on particular machines. The nodes on a machine are the processes. The third level (sub-net) is objects networked inside processes. The nodes inside an O-O process are the objects. CORBA allows object communications beyond the scope of a process or machine. Thus, the object nodes become nodes at all network levels. However, CORBA is not sufficient for an enterprise to treat all of its resources in an integrated manner. Enterprise Application Integration Frameworks address this issue. When CORBA is part of the structure of these frameworks, it is possible map the resources and applications of the enterprise onto the underlying network. Thus the behavior of the enterprise is reflected in the behavior of the network. It is the purpose of the present paper to present such a framework by introducing five unique architectures. In addition, it is shown how SAS products can be integrated into the framework and thus explicitly integrated into the enterprise.

ARCHITECTURES

The overall architecture presented here is one that is process-centric as opposed to thread centric. A process centric architecture is one which utilizes processes as its core processing unit, whereas a thread-centric architecture considers threads (in a single process) to be its basic unit of work. This is not to imply that threads are not used. Each process can in fact be multi-threaded. One advantage of process-centric architectures is that it is more global than thread-centric architectures in that threading can be incorporated into any implementation; however, it is difficult for implementations of thread-centric architectures to expand to use processes. In addition, the overall architecture is one that is N-tiered, distributed, and object-oriented. Figure 1 represents a rendering of the architectures described below.

Virtual Server Architecture (SVA)

The framework proposed here has a number of unique architectures, one of which is the Virtual Server Architecture. This architecture proposes that every GUI client process can be associated with its own GUI server (e.g. a Java servlet) process. This creates a Virtual GUI Server Layer. Likewise, every database "transaction" can have its own database server process. This creates a Virtual Database Server Layer. Two other virtual layers are found with this architecture: the Virtual Compute Layer, and the External Source File Server layer. The Virtual Compute Layer contains processes that analyze, model, and simulate. The External Source File server layer contains server processes that process incoming source files. All virtual layers are contained in N-tiered, distributed environments. Any number of GUI, database, compute and XS server processes can be interconnected, as proposed by the Multi-process Interconnect Architecture (see below). These architectures explicitly incorporate the model-viewer-controller paradigm. Briefly, viewers are detached from their models. Thus viewers and models may reside on different machines. Viewers are placed with GUI processes and models are placed with server processes. All server processes are created, monitored, tracked by a central "DIRECTOR" process. Given the fact that these are processes and not threads, this architecture provides for inherent parallel processing. Each process executes in parallel with all other processes. Multi-tasking can occur within a process via threading. Functionality of these processes is extensible in that APIs exist for all servers. That is customized code can be added to server process processing in all virtual layers. On the GUI side, since a GUI client can be associated with its own server, the server process can dedicate all of its resources to servicing the GUI client. Not only can servers process customized code; any server may be an application from any vendor. In this case, the "DIRECTOR" process invokes an instance of the application.

Inherently Distributed Objects Architecture (IDOA)

The Inherently Distributed Objects Architecture is one which proposes that any server process in any virtual layer can contain any object from any defined class (scoped by language) including classes defining business objects and classes responsible for defining meta-data. In addition, these objects are not for-seen to be limited to server processes; GUI processes may have instances of these objects also. One feature of the architecture is that these objects can be instantiated dynamically as needed. Any object in any process may address any of the following architectural perspectives: Enterprise, Technology, Engineering, Computation, and Information.

Distributed Projects Architecture (DPA)

The Distributed Projects Architecture is based on the "project" as being its basic unit of work. The characteristics and properties of "projects" are mapped to operating system, and application containers. Customized code, data, reports, graphs tables, models, analytics, etc. for a distributed project can reside separately or concurrently on any machine the user has access to. This architecture, coupled with the previous two, allows functionality to be perceived to be local to the viewers. In, addition, this architecture is applicable (as is all the others) to any industry, any enterprise, and any business unit in an enterprise. Thus a CRM application and a portfolio management application are essentially identical except for the way the models (in the server processes) are viewed.

Layered Parallel Processing Architecture (LPPA)

Every server process executes in parallel with every other server process. In addition, every server process can in turn spawn other processes (sub-processes) anywhere in the distributed environment to execute in parallel. In turn, every sub-process can spawn other processes that execute in parallel. In addition, any server process may be threaded for additional processing capabilities and inter-process communications. Parallel processes that collaborate tp accomplish a common goal are managed by the “Director” process.

Multi-process Interconnect Architecture (MPIA)

The multi-process Interconnect Architecture connects processes, be they be GUIs, any type of server process, database instances, web processes or documentation processes, on three different levels: inter- and intranet-work connections, inter-process connections on the same machine and inter-

object connections. The tools that make these connections are ORBS, operating system inter-process communication mechanisms (IPCs, e.g. pipes), sockets and TCP/IP, and language threads. The interconnect architecture is managed by a central "DIRECTOR" process. The process creates and/or requests, tracks and monitors all connections

The "DIRECTOR" process(es)

The "DIRECTOR" process (or processes) is one which manages, tracks and audits objects and processes It is clonable which can allow dynamic load balancing a among machines regardless if they are part of an intra- or internet. Thus scalability is ensured. It consists of three components: A Server Domain Manager (SDM), A Data Management Manager (DMM), and a System Interaction Manager (SIM). The SDM offers services for load balancing, resource management, and server administration (creates, monitors, tracks, and manipulates server processes). The DMM offers services for data validation and verification in all virtual layers (utilizing a meta-data repository), data management *per se*, and communications with the document and web server. The SIM offers services for load balancing, resource management and monitoring, interprocess communication monitoring, event and message management and security. Many of these services exist in CORBA compliant ORBs. In addition, the SAS proprietary ORB also provides some of these services. In either case, neither ORB type by itself is not sufficient to address the enterprise (and inter-enterprise e.g. e-commerce) completely.

INTEGRATION OF SAS PRODUCTS

AF/EIS

SAS AF and EIS are products that allow O-O applications to be created and then executed within SAS sessions. In terms of the presented framework, these applications can exist in every layer. GUIs can be created that reside in GUI processes (data presentation and visualization), applications (server sessions) with business logic can exist in any server session, vertical domain applications (Finance, Health, Transportation, Manufacturing, etc.) can be created and parsed across the virtual layers, customized algorithmic classes can be implemented for analysis, customized data processing classes can be created to access data warehouse - in particular SAS datasets to create an "object-relational" database, and so on. These applications can have access to the entire SAS product line. What is of significance here is that SAS now has a proprietary ORB that utilizes CORBA IDL. Thus, other applications, written in CORBA compliant languages (JAVA), can access the entire SAS system via object mechanisms. Likewise, these applications can access these other non-SAS O-O applications.

Base

Like AF and EIS, BASE SAS processes can exist is all virtual layers. The MACRO language and the DATESTEP are powerful tools for tasks such as extraction, transform and load (ETL) processing, data validation and verification, and extraction from source systems. Thus, processes with BASE SAS work very well in the XS Virtual Layer. In addition, due to its ubiquitous nature (every SAS product requires it) and the fact that it can be accessed by AF and EIS applications via SUBMIT blocks, BASE may be found in any layer that applications based on these products are found in.

Enterprise Miner, Enterprise Reporter, ETS, STAT, GRAPH

Enterprise Miner is the award winning data mining product from SAS. In addition to utilizing this product interactively via its GUI, Enterprise Miner can fit into the Virtual Compute/Modeling Server layer and the Virtual Database Server Layer. In both layers users can invoke multiple Enterprise Miner sessions as one logical job in any phase of development. For example, data from multiple databases (data warehouses) can be accessed and modeled simultaneously by any user and any number of users. In addition, pre- and post-processing can be done sequentially within single server session or in parallel between server sessions. Pre- and post-processing may involve exploratory analysis (BASE), post-hoc statistical analysis (STAT, ETS), validation and verification of models as well as data, and generation of tables (BASE) and graphs (GRAPH) and reports (Enterprise Reporter). Results from models can be written back to the data warehouses that data was extracted from or to warehouses other than the source. All processing can be done either in batch or interactively via APIs. Though activities in this layer are conceptualized to take

place in a production environment, there are no restrictions on these activities in development and test environments for development and testing. Similar activity can occur in the Virtual Compute-Modeling Server layer. It is conceptualized that this layer is more of a development and test environment for modeling and simulation. Otherwise it can also be treated as production environment of all other activities. Although these servers do not access data warehouses located remotely, these servers are allowed to access data residing locally. It should be kept in mind that all layers are virtual layers and exist in a distributed environment. Server sessions in any layer need not reside on the same machine.

Warehouse Administrator

The Warehouse Administrator is a tool to administer multiple data warehouses by managing enterprise meta-data associated with each warehouse. This is a natural product for inclusion into an enterprise application integration framework. Given that it is a SAS product, it is already integrated with other SAS products and in addition, it is integrated with databases from which meta-data is obtained. It is conceptualized that this product be integrated with a "DIRECTOR" process in this framework and in any process where data validation and verification is performed in any virtual layer.

SAS Integration Technologies

SAS has a number of integration technologies available to users, developers, enterprise integrators and object-oriented architects. The foundation to these technologies is the Integrated Object Model. This technology allows CORBA and COM/DCOM technologies to be integrated with SAS products.

COMPARISON TO CURRENT ARCHITECURES

The overall architecture here diverges from current architectures in that it layers a process centric architecture into a complete enterprise application integration architecture. Two immediate benefits are derived from this design: inherent parallelism and the ability to integrate applications with diverse functionality from multiple vendors into a framework such as the one presented. This second benefit cannot be achieved via a thread centric architecture due to the nature of the architecture: threads reside in applications. Another benefit is that viewers are explicitly detached from models allowing for multiple concurrent view of the same data. These models can reside in any of the server layers anywhere in the distributed environment giving the viewers access to all of the analytical, graphical and data warehouse data across the network regardless of the domain (e.g. Financial, Pharmaceutical, Defense, Education, etc) the data belongs in. Traditionally, domain business logic is "stoved pipe" into an application. The architecture presented here allows for business objects, either customized or exiting in vendor applications, to be seen by objects in other applications. In addition, any process can have access to any other process in the network allowing for complex parallel processing among processes. No other architecture has inherent parallel processing built in. Finally, most if not all other architectures are designed around the user. This architecture is designed around a project. This provides for maximum reuse of code in new applications in that they are essentially new facades for GUIs and allows for a more efficient use of the network and resources in that data can reside where its most appropriate.

One final point, this is a framework and not an application *per se*. It is the components of the framework that are distributed and not another distributed application. It is the way the components are connected (aggregated), in a network topology (via CORBA, server sessions, IPCs, sockets, IOM), which gives rise to the various emergent behaviors of the network (enterprise) as a whole. It is these emergent behaviors that allow sharing of data and functionality across the enterprise. The one component that resembles an application is the "DIRECTOR" process that can utilize CORBA compliant ORBs and/or the SAS proprietary ORB.

CONCLUSION

CORBA and Enterprise Application Frameworks allow an enterprise to integrate its resources and applications for information sharing across the enterprise. This paper presents five architectures for an enterprise application framework and the underlying tools for construction such as CORBA. It is concluded that SAS products, in particular IOM technologies, can be successfully integrated into these frameworks. This gives an enterprise access to all of the functionality provided by SAS products.

REFERENCES

http://www.sas.com/rnd/itech/doc/dist-obj/index.html – This is the root page for distributed objects.
http://www.omg.org/ - This is the URL for the Object Management Group.

Author Contact

Matthew K. Hettinger
Mathet Consulting
Suite 1400, Zurich Towers
1450 E. American Lane
Schaumburg, IL 60173
Phone: 847-330-63 75
Email: hettinge@mcs.com

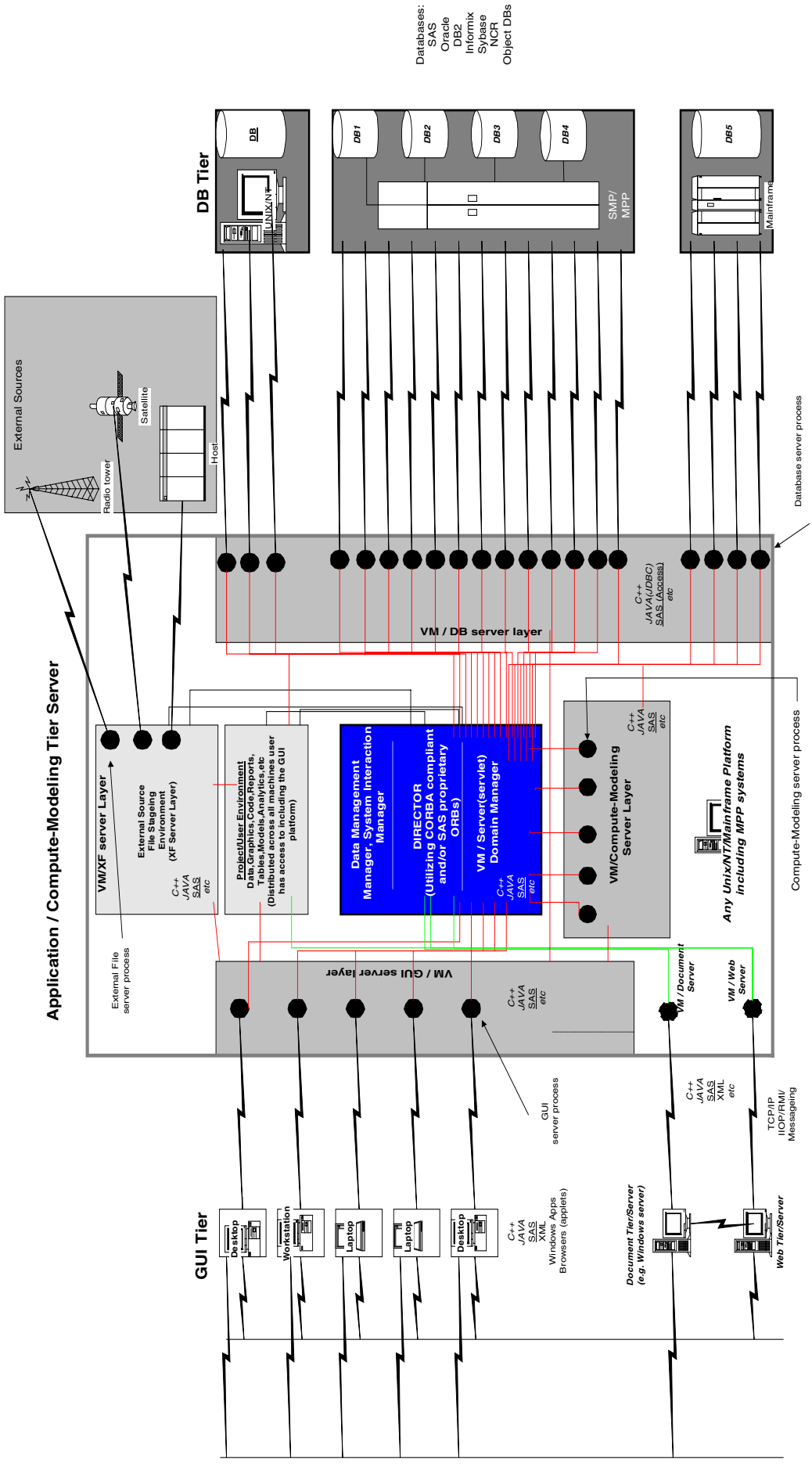

Figure 1. Enterprise Application Integration Architectures.