FRAMEWORKS - COTS INTEGRATION THROUGH ENCAPSULATION

Paula Renfro and Neal Walters
Loral Federal Systems
Manassas, Virginia

ABSTRACT

Loral Federal Systems is exploring the use of an Integration Services Architecture, ISA, as the basis for integrating Information Technology, IT, systems which are predominately COTS-based. The primary objective of the ISA is to lower the lifecycle costs associated with COTS systems. The ISA must address several dimensions of COTS integration including process, control and data. The ISA must also accommodate change. The system users must be isolated from the continuous impact of change in today's systems - changes associated with technology, COTS product end-of-life and business rules to name three.

In the past we have implemented COTS-based systems using ad hoc "glue code" architectures and found those architectures to be deficient in many ways. After giving consideration to developing our own ISA framework we have decided to proceed by adapting a commercially available application development and runtime environment to integrate COTS and non-COTS functions. This paper defines the requirements for an ISA framework and addresses the work required to encapsulate COTS applications in a commercial runtime environment.

AUTHOR BIOGRAPHIES

Paula C. Renfro is an Advisory Programmer in the Federal Systems Integration Laboratory at Loral Federal Systems in Manassas, Virginia. She is currently responsible for all SNAP development tasks for the lab. Prior to its acquisition by Loral, she worked for IBM commercial and Federal Systems Divisions for 20 years. Most of her career has been spent in all phases of the development of software systems for both federal and commercial customers. Her prior experience includes a wide variety of application areas including tools development, banking transactions, data model development, and multimedia applications. She received her B.A. in Mathematics from Spelman College in Atlanta, Georgia.

Neal L. Walters is a Senior Engineer in the Federal Systems Integration Laboratory at Loral Federal Systems in Manassas, Virginia. In 1993 Mr. Walters was one of the first professionals in IBM to be recognized as a "Certified Systems Architect". Mr. Walters is currently responsible for systems and software architecture of new systems in both the Information Technology and real time combat systems domains. Prior to its acquisition by Loral, Mr. Walters worked for IBM commercial and Federal Divisions for 28 years. Recent publications include: " *Using Harel Statecharts to Model Object-Oriented Behavior*", Software Engineering Notes, Oct 1992 and " *An Ada Object-Based Analysis and Design Approach*", Ada Letters, July, 1991. Mr. Walters received a BEE from the University of Virginia in 1960 and an MSEE from NC State University in 1970.

FRAMEWORKS - COTS INTEGRATION THROUGH ENCAPSULATION

Paula Renfro and Neal Walters
Loral Federal Systems
Manassas, Virginia

INTRODUCTION

Current project trends are tending to maximize the use of Commercial Off The Shelf, COTS, and minimize new software development. The development of systems with a high content of commercial hardware and software are likewise becoming more prevalent in the Federal marketplace. Much of the emphasis on COTS solutions is driven by the need to lower the cost and deployment time of new systems to meet reduced budgets. Increased focus on information system technology to reengineer the government also contributes to the COTS emphasis. These systems require an architecture that supports open system standards and accommodates the use of COTS products.

In the past the tendency has been to integrate COTS applications in an ad hoc manner, creating glue code as the need for it is discovered, to permit applications to interface with each other and external interfaces, passing control and data, implementing business processes, and standardizing user interfaces. This system structure is illustrated by Figure 1. Each glob of glue is unique to the set of interfaces between a pair of applications, and may require frequent modification as business needs and technology changes. Because this type of system is expensive to change and support, other mechanisms are preferred.

The purpose of this paper is to define an architectural approach for integrating COTS software applications. This architecture addresses COTS integration in terms of process, data, control, presentation and environment. We believe this approach will lower the lifecycle cost and risk of COTS integration and support.

CHALLENGES OF INTEGRATING COTS-BASED SYSTEMS

Information Technology, IT, systems focus on data - the movement of data, the security of data, the integrity of data. These systems usually have large data bases which tend to be distributed, often geographically. The systems are often built from legacy systems requiring the applications within the system to interface with both new and old data and applications. An IT system must manage all business data and information as a consistent whole, providing the required system functionality regardless of where

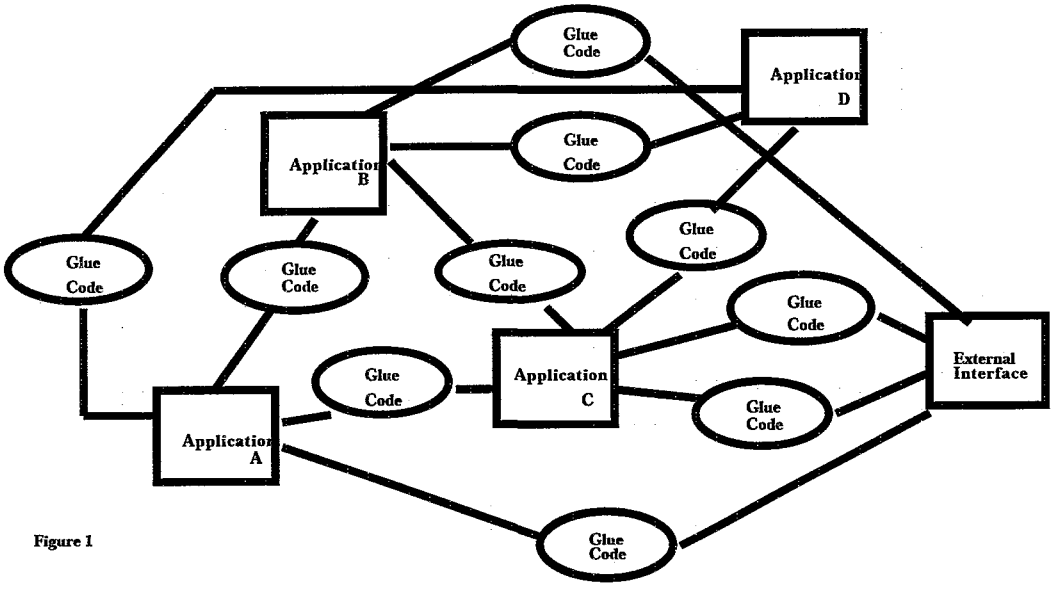

Figure 1

data elements are stored and the data storage technology used to store them.

When using COTS applications to implement an IT system it is important to recognize the challenges of COTS-based systems and design the system to mitigate these challenges. A set of COTS applications never provide the exact functions required for a given system. The COTS applications either provide a subset of the functionality required, duplicate functions found in other applications or provide functions not required at all. COTS applications are usually not designed to interoperate with each other. The data used in COTS products is stored on a variety of media including flat files, commercially available DBMSs and proprietary data bases. The sharing of data between disparate COTS applications is only one of the challenges of integrating COTS-based systems. It is impossible to control or coordinate upgrades to COTS products. Changes in any one COTS application can have an affect on the system as a whole. It is a frequently-voiced opinion that most COTS applications are not "Ada friendly", a requirement in some Government systems. In order to mitigate the challenges of COTS-based systems it is crucial to isolate the COTS applications from each other and the system data. This will help insure changes in any one application do not ripple through the entire system. This application encapsulation also allows the system to be able to take advantage of changes in technology and withstand the inevitable COTS application end of life.

To ensure customer satisfaction, the system must provide a consistent look and feel across COTS applications. Access to information and services for the customer should be based on the customer's needs and the business process. Access should not be driven by the individual COTS applications.

New IT systems can benefit from a workflow management approach to effectively route and distribute information among cooperating tasks in a business process. These tasks can be manual, automated or automated with human assistance. The workflow facilitates the integration of COTS applications by removing the responsibility of routing and distributing the data from the COTS products themselves or requiring COTS-specific glue code to accomplish this routing. The design of the workflow must be driven by the business rules and goals and be independent of the specific application implementations that comprise the system. The system must also be able to adapt to changes in the business rules over time with minimum impact to the overall architecture.

Thus, the challenge for the Integration Services Architecture, ISA, framework is to provide the mechanisms to integrate COTS products, legacy software and data and newly developed software in the same system. The system must appear to be a homogeneous design to the users even though it may contain many heterogeneous hardware and software entities.

INTEGRATION FRAMEWORKS

When applied to integration mechanisms the term "framework" has been used in several ways[1]. In this paper framework, in the context of an ISA, refers to the integrating mechanisms and services for a COTS-based system.

Over the past five years the hoped for productivity breakthroughs using CASE tools met the reality of integrating disparate tools. A significant amount of work has been done and numerous papers written to define integration frameworks for CASE tools. Examples of work in this area include the NIST/ECMA Reference Model[2], IEEE Standard 1175, and the NGCR PSE Reference Model[3]. These models and standards propose integration mechanisms for tools. Borrowing from the types of integration mechanisms suggested for CASE tools, we suggest the following mechanisms be provided by an ISA framework for COTS-based systems:

1. Process Integration,
2. Data Integration,
3. Control Integration,
4. Platform Integration, and
5. Presentation Integration.

Process Integration

Process integration describes how application software, support services and humans interact as a system to implement the defined business goals. In this definition a process consists of "process steps" which are the elemental units of work performed by the system. Good process integration occurs when entities of the system cooperate efficiently to accomplish process steps without duplication or omission. A major key to implementing process integration in a system is sufficient recognition and control of the process steps. The process step control mechanism is an Event Manager. The Event Manager has no knowledge of the business goals but works with a Process Manager who does. When the Event Manager recognizes the completion of a step it initiates (zero or more) other steps based on direction of the Process Manager.

Data Integration

The goal of data integration is to provide all the entities of the system with a seamless, accurate and understandable representation of the data which is appropriate to their accomplishment of the process steps they perform. Mary Shaw[4] writes about system integration: "Large software systems are often integrated from pre-existing systems. The designer of such a system must accommodate very different - often incompatible - conceptual models, representations, and protocols in order to create a coherent synthesis." Even through Ms. Shaw was not directly dealing with COTS-based systems, the "pre-existing systems" in her statement accurately reflects the influence of COTS on a system solution. Data integration has four facets:

- Interoperability - is a measure of how common the data view (structure and/or schema) is among the system entities. Interoperability can be good with the use of a single, common, shared file system to poor with a common data interchange format.
- Consistency - is a measure of the semantic integrity of data as understood by the entities of the system. An example of poor semantic integrity: a "labor-month" may represent 160 hours of labor to one software application and 172 hours to a different software application.
- Redundancy. System entities are well integrated if there is little duplicate data and automatically derived data in the system. Redundancy jeopardizes data consistency and leads to refresh-time problems.
- Synchronization - represents the temporal nature of the data such that cooperating entities may work concurrently on the same data. Time, events or conditions may require data synchronization in a system.

Control Integration

Control Integration is accomplished by the Event Manager and System Services (e.g. Operating Systems, DBMS). The control mechanisms perform communication services, time management, software invocation, data retrieval, security and other services. Good control integration occurs when there is a uniform, cooperative structure of control through out the system. All non-exception events are managed to accomplish the business goals of the system.

Platform or Environment Integration

Simply stated, platform integration occurs when the entities of the system have no dependence on the underlying hardware and its associated software (e.g. operating systems). This is a common goal in today's distributed, heterogeneous, open systems.

Presentation Integration

The goal of presentation integration is to improve the efficiency of the human users of the system by reducing the cognitive load of those users. This is accomplished by :

- Reducing the number of interaction and presentation paradigms,
- Providing interaction and presentation paradigms that match the user's cognitive models,
- Meeting the user's response time expectations,
- Ensuring correct and useful function and information is available to the user, and,
- Hiding the system implementation mechanisms, especially those mechanisms subject to lifecycle technology change, from the user.

In a COTS-based system which uses different products to interface with the human user there will likely be "look and feel" differences between application sessions. (The notable exception is applications written for a common environment such as MicroSoft Windows™). In such a system, the goal of presentation integration is to give the users a common navigation system for signing on, checking security, starting and stopping applications, and performing customized functions. This common navigation system should have the same look and feel regardless of hardware platform, operating system or location in the system. The basic presentation should not have to change when technology insertions are done at a later time. Training updates for using the system should only have to address new function, not changes to the navigation system.

THE ROAD TO AN ISA FRAMEWORK

In early 1992, Loral Federal Systems began development of the Launch Team Training System (LTTS) for NASA. This system would increase the competence of the shuttle launch team engineers through multimedia training. LTTS was to be a self-paced Interactive Computer Based Training System (CBT) supporting courseware authoring, management and delivery within a client-server environment. The system to be developed would integrate COTS hardware and software products to create a seamless environment.

The configuration proposed consisted of 21 workstations running OS/2. COTS integration was required for navigation between products (presentation

integration) and between the Authoring tool and the database (data integration). The integration of the COTS products relied completely on OS/2 which provided the services necessary to integrate the data from the disparate COTS products. As a result, very little glue code was developed.

In 1992, the number of products running on OS/2 were limited and capabilities were slow to migrate to the operating system. Approximately six months into development, the Authoring product announced a DOS/Windows upgrade that greatly improved the capabilities of the product. The upgrade for the OS/2 version, however, was not available nor expected to be available by the time LTTS would be in production. In addition, two other COTS products were announced which ran under DOS/Windows, but not OS/2. These two products also significantly enhance the capabilities of the system. The customer, understandably, wanted to be able to take advantage of these improvements in technology. Because of a requirement that the system support a wide variety of vendor and third party software products, it was necessary to move to a Microsoft Windows environment. Although the change in operating system did not effect the hardware for LTTS, all but one of the COTS products and six months of development work had to be discarded.

The architecture and development team redesigned the system to mitigate the effects of COTS upgrades and to ensure that the COTS products were isolated from each other and the operating system. To this end a framework was designed which consisted of a standard interface and wrappers for the Authoring tool and the database. The standard interface encapsulated the data. The interface was designed to insulate the courseware from the physical and logical structure of the database and from the underlying operating system. The wrappers converted the data to and from this standard interface. The implementation of a framework allowed the outward operation of the system to be unaffected by changes in the Authoring tool or the database over the life of the system and thus the system could remain "state-of-the-art".

Although the framework developed for LTTS was a major step in COTS integration, there were a few shortcomings. The framework was specific to LTTS and therefore not transferable to any other system. Also a major portion of the software developed for the framework supported the communications and database access services specific to the Novell LAN and the Netware SQL Server Database. The framework would be improved by encapsulating these services. The development of the framework for LTTS helped define the requirements for an ISA framework.

REQUIREMENTS FOR AN ISA FRAMEWORK

This section addresses requirements for the ISA for a COTS-based IT system. These requirements are based on the integrated software development reference models discussed earlier and our own experience in integrating and supporting COTS-based IT systems for a variety of customers, applications and environments. Both the framework for software development integration and the framework for COTS-based IT systems have similar needs for the underlying integration mechanisms. The major difference between the two is an emphasis on function in the software development framework versus the emphasis on data in the IT framework. The broadly stated requirements for the IT framework are:

The framework must accommodate distributed, scaleable, client-server architectures. One significant trend in the 1990s is distributed processing. Workstation capacity can be added in small, economical, and powerful increments. Workstations provide advanced interactive, graphic interfaces required by business process applications. A distributed architecture provides a practical means of integrating existing heterogeneous components.

The framework must provide Control Integration mechanisms that will enable Process Integration. The basic framework services must include event monitoring and event control. The detectable events must include: time of day, elapsed time, creation of data, movement of data, access to data, creation of a message, receipt of a message, asynchronous human action, application exceptions and system exceptions. Ideally the Process Manager can be defined and changed through graphical tools that depict the desired workflow of the system. The workflow model should show all entities (software and humans) of the system and the work items (data) they need to complete their process steps.

The framework must accommodate data integration through commercial database management systems and ASCII flat files. IT systems typically manage large volumes of data in mass storage that persists beyond the duration of the users sessions. Much of the data in a "new" IT system is often retained from a previous system. The framework should provide a database mapping facility. This will enable COTS and non-COTS applications to access and manipulate

data that other entities have created. Multiple mappings of data should be supported.

Today's IT systems must plan ahead for technology insertion in the context of computing power and storage media. The framework should provide portability across widely accepted hardware platforms, operating systems, windowing systems, databases, and open system standards. The framework should also provide a published application programming interface, API, for new application development.

FRAMEWORK IMPLEMENTATION

Make or Buy ?

To the authors' knowledge there is no commercial product which is marketed as a COTS integration framework and addresses the requirements listed above. Even those products marketed for tool integration such as SoftBench™ from Hewlett Packard, do not address all the integration mechanisms required for an IT system, especially those for data. Therefore we were faced with the decision - *make* an ISA from scratch; or, *buy* a commercial product and enhance it to support COTS application integration.

Make an ISA. New development would ensure total control of all aspects of the ISA - functionality, schedule, technology and change cycles. The risks associated with lost of control of a commercially based ISA would be alleviated. Dependencies on vendor schedules could be eliminated. A consistent look and feel to the system could be ensured during system design. However, long and often costly development cycles are the down side of new development. Developing major parts of ISA such as communication services would "reinvent the wheel". Our experience with LTTS has shown that a newly developed ISA would cater to the first large project using it, adding customizations that would be unnecessary and unwanted by other systems.

Buy an ISA. Even though there is no COTS integration framework being marketed there are commercial products that address many of the requirements of the ISA. The candidates we have looked at are listed below. Most of the candidate products are marketed as (rapid) application development tools/environments. The disadvantages of using a commercial product as the foundation for the ISA is basically a loss of control. All the challenges associated with using COTS applications also apply to using a COTS-based framework. Vendors change their products on irregular schedules; a new release may not be upward compatible with the previous release; old releases are not supported after a few months; and, vendors can go out of business.

The following paragraphs give a brief description of the commercial products we considered for the foundation for the ISA framework. These products provide development environments and/or tools for application development. Many of these products interface to popular DBMSs and Graphical User Interfaces. Many provide standard communication services. Some of these products incorporate or interface to word processors and spread sheet type applications. The descriptions of these products is based upon information obtained from the vendor, and is provided without independent evaluation or validation by the authors or Loral Federal Systems.

SNAP and Workflow Template (WFT) from Template Software[5]. SNAP is an object oriented development environment which supports distributed and client/server architectures, rules-based applications through the use of an inference engine and migration to a variety of platforms. WFT is a tool for the development of enterprise-wide production workflow systems. WFT enhances and complements SNAP in the automation of complex business processes. SNAP is primarily intended for new C application development, but provides a mechanism integrating with any language that can be linked to C code. SNAP also provides an API so existing systems can call and interact with SNAP objects.

Universal Network Architecture Services (UNAS) from TRW[6]. UNAS is intended primarily for Ada real-time systems. A software architecture tool, SALE is available with UNAS for defining the system topology. UNAS rapidly produces the system middleware (communications and other services) after the system topology is defined. UNAS does not have provisions for incorporating COTS software applications.

UNIFACE Six from Uniface Corp[7]. UNIFACE Six provides an integrated development environment for creating technology-independent client-server applications. UNIFACE Six provides a set of tools to define, store, and maintain the enterprise model as well as the ability to rapidly generate applications that utilize an integrated model repository. UNIFACE Six supports many commercial database systems and hardware platforms. UNIFACE Six is primarily an application building toolset versus a COTS integration framework.

FOUNDATION from Andersen Consulting[8]. FOUNDATION provides a methodology and tools for building host based or distributed client-server applications. API services include Dynamic Data Exchange, Environment Loader, Error Handler, Program Front Ends and Shared Data Manager. Foundation provides a message-based architecture that provides many communication services to the applications. FOUNDATION is primarily intended for new C and COBOL application development, not COTS integration.

FactoryLink IV Software (FLS) from United States Data Corp[9]. FLS is a real-time application and technology enabler software system for configuring scaleable, platform-independent applications in an open, heterogeneous environment. USDC provides application Modules to perform the system functionality. FLS interfaces to most of the commercial database products and supports many communication protocols. The application domain for FLS is Production Control, as opposed to IT. There are no provisions for incorporating COTS application software.

Galaxy Application Environment from Visix[10]. Galaxy is a multi-platform application development environment designed specifically for constructing large-scale commercial applications. The API supports both a kernel-based architecture and a fully distributed client-server model. Galaxy provides network-independent services such as hypertext help and inter-application communication. Galaxy has no direct provisions for COTS software product integration.

Genesis Systems Integrator from MTI Financial Systems, Inc[11]. Genesis provides the means for disparate systems to work together in ways which were not necessarily envisaged by the constructors of these systems. Genesis accesses a database containing a description of the messages produced by every application on the network, the messages(s) to be produced and delivered to other applications. Genesis is entirely configured via its database. New translations can be accommodated with writing new programs. Support for COTS application programs is unknown.

Conclusion

We conclude that using a commercially available product as the basis for the COTS-based ISA framework is the right choice. We believe the risks and costs of starting with a commercial product are less than starting a new development.

We have chosen SNAP™ and WFT from Template Software as the foundation for our candidate COTS ISA Framework. SNAP provides the basic integrating mechanisms for data, control, presentation and platform. SNAP supports most of the major RDMS including Sybase™, Oracle™, and Informix™, the Open System Standards and the defacto standards of VMS, MicroSoft/Windows™, and Windows/NT™. Ind additon SNAP supports the Intel X86 processors and many of the Unix-based processors. A Process manager can be implemented using the workflow template. The Workflow Template provides graphical tools to define tasks and the movement of data throughout the enterprise. SNAP/WFT also interfaces with several IDEF tools to define business rules and data schemas. However, SNAP does not support Ada and has no direct support for COTS integration. SNAP does provide an API to support the integration of C and C++ legacy software. Our plan is to build "wrappers" to interface the COTS application products to the SNAP API. Each COTS product must have a personalized wrapper. See Figure 2.

Application wrappers provide another layer of software between the API and the COTS product. The wrappers insulate the application from the rest of the system and help establish application independence from changes in the system. Equally important, a wrapper protects the rest of the system from changes, e.g., a new version or a replacement application . The wrappers provide the mechanism for integrating COTS products. In general the wrappers will:

- Provide hooks to interface to the ISA event manager, detecting application actions and change of state,
- Perform data transformations unique to that application, and
- Map the application's environmental dependencies to the ISA.

SUMMARY

There are many challenges associated with building systems from COTS products. System integrators and customers alike have underestimated the effort to address these challenges over the last several years. At Loral Federal Systems our early implementations of COTS-based systems were naively architected. Custom code was developed as needed to link products together; the solution was driven by functional requirements rather than business rules and goals; and, the solutions were customized for specific hardware platforms and operating systems. In retrospect these architectural "shortcuts" made changes to the system more costly.

Our experience with LTTS and other applications has lead us to better ways to integrate COTS-based systems, especially in the domain of Information Technology where the majority of all software and hardware in the system is COTS. We have defined the requirements for an Integration Services Architectural Framework which addresses the specific needs for COTS integration. We have decided to implement the ISA using a commercial product as the foundation. We chose SNAP™ and Workflow Template from Template Company for that foundation. We believe the key to adapting the foundation product to be a general COTS ISA is in the design of wrappers that encapsulate each COTS application product, isolating it from system changes and vice versa. Over the next several months we will continue to refine the definition and design of wrappers in the context of the API provided by SNAP.

REFERENCES

[1] Wallnau, K. , Feiler, P., *Tool Integration and Environment Architectures,* Technical Report, CMU/SEI-91-TR-11, Software Engineering Institute, Pittsburgh, Pa., May, 1991

[2] *Reference Model for Frameworks of Software Engineering Environments,* Technical Report ECMA TR/55, 3rd Edition, August, 1993

[3] *Reference Model for Project Support Environments,* Version 2.0, September 2, 1993

[4] Shaw, M., *Software Architecture for Shared Information Systems,* Technical Report, CMU/SEI-93-TR-3, Software Engineering Institute, Pittsburgh, Pa., March, 1993

[5] Template Software, Inc., 13100 Worldgate Drive, Suite 340, Herndon, VA 22070-4382. (703) 318-1000

[6] Rational, Inc., 4041 Powder Mill Road, Suite 200, Calverton, MD 20705. (301) 937-4900

[7] Uniface Corp., Bay Colony Corporate Center, 1000 Winter Street, Suite 1600, Waltham, MA 02154. (617) 890-1615

[8] Andersen Consulting FOUNDATION - Suite 2004, 69 West Washington St., Chicago, IL 60602. (800) 458-8851

[9] USData Corp., 2435 N. Central Expressway, Richardson, TX 75080. (800) 527-0593

[10] Visix Software Inc., 11440 Commerce Park Drive, Reston, VA 22091. (703) 758-2702

[11] MTI Financial Systems Inc., 335 Madison Ave., 11th Floor, NY, NY 10017. (212) 557-0022

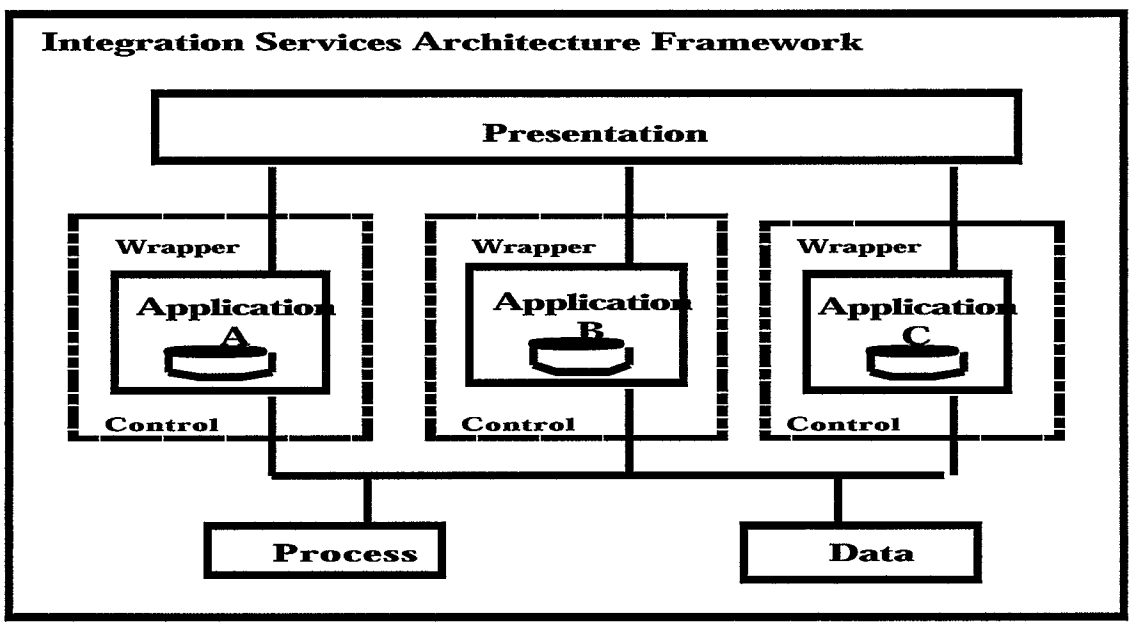

Figure 2