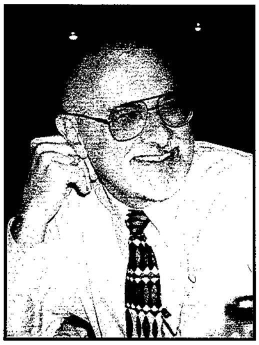
FEATURE ARTICLE

Migrating the CHCS Clinically Relevant Database and Business Rules to Object Technology



Terry L. Wiechmann

by Terry L. Wiechmann

Background

ESI Technology has been contracted by Military Health Services Systems (MHSS) of the DOD Health Affairs to migrate their Composite Health Care System (CHCS) application business rules and database to object technology using ESI's integrated Distributed EsiObjects™ product suite.

The CHCS system evolved from the work done at the Veterans Administration on the Decentralized Hospital Computer Program (DHCP). Both application systems are File Manager-based and run on standard M platforms.

Distributed EsiObjects is ESI's object oriented, client server development environment based on open, standard middleware and M Technology. All popular, commercially available presentation packages are supported, such as Visual Basic, Delphi and Java.

Health Affairs of the DOD has elected to evolve the CHCS application system to an object repository using EsiObjects. Government and/or commercial-off-the-shelf (COTS) application systems will have access to the database through an Integrated Software Suite interface.

Scope of Paper

The scope of this paper is confined to ESI's role in the Object Oriented Technology (OOT) project underway at MHSS. The work tasks undertaken by ESI form a small part of a much larger program. The following sections will describe, in a general manner, only those components that ESI built to accomplish the tasks listed in the previous section.

Project Tasks, Goals

To date, four work tasks have been awarded to ESI.

1) The goals of the first task were to open the existing VMS-based M systems using standard, open middleware technology, specifically CORBA, RPC and DCE. This task was completed and delivered as scheduled. It provides a common multi-thread gateway into the current VMS-based M systems. It was designed to adapt any M system by simply writing an M adapter for the target system. Connectivity via these paths is made available to a client application system via the Client Transport Adapter or to a Windows GUI environment via an OCX/ActiveX that is registered on that PC.

2) The goal of the second task was to build a generalized object repository that can be used to store clinically relevant data element definitions and associated business rules. A compiler that generates an object oriented runtime environment uses this information, along with generalized common code and mapping specifications. Additionally, the repository must support input from other sources such as commercial modeling tools and other database systems.

There are two general constraints that must be religiously adhered to. They are:

i) The object runtime environment must run in parallel with existing CHCS systems without affecting its operation.

ii) The business rules runtime wrapper will display the same behavior as its CHCS system counterpart.

This set of tools was delivered and is currently being used to accomplish task 3. The tools will evolve as the data definition and business rules extraction continues to produce unforeseen patterns. These tools are collectively referred to as the Application Repository Tools (ART).

3) The goals of this task are to use the tools to convert clinically relevant patient registration data elements and business rules to the new object environment as a proof of concept. The generated runtime environment has been delivered for laboratory testing.

4) The goal of this task is to continue to convert the remaining modules (Admission Discharge Transfer, Pharmacy order entry, Radiology order entry and Laboratory order entry) in a systematic and logical fashion. This task has just started. As each task is complete, it will be delivered for extensive testing and integration into the Integrated Software Suite.

General Client Server Overview

The diagram below illustrates a general client server view of the components outlined in the Project Tasks Goals section. It consists of the middleware, client side systems and the server side development and runtime environments. Each component will be explained.

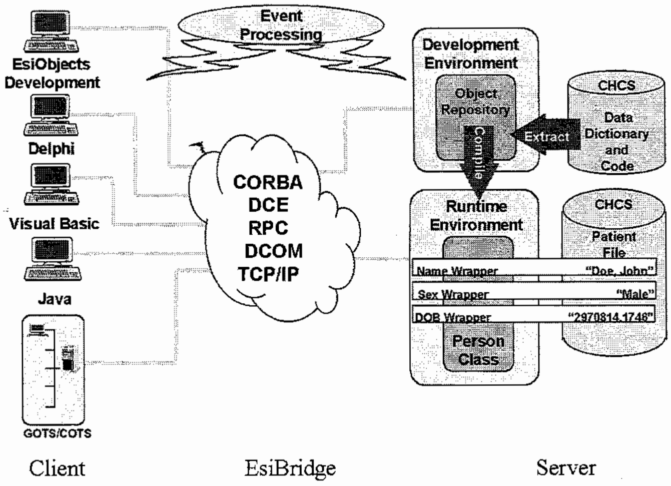


Figure 1

Middleware

The middleware consists of a generalized gateway that supports standard, open technologies—specifically, CORBA, DCE and RPC. DCOM is used within a Microsoft environment. Collectively, all these implementations are called EsiBridge. They all use a common server gateway built as a result of the task 1 described above.

The gateway can access any supported M system on the server side either in a 2 or 3-tier client server configuration.

Client Side

The client side consists of two environments:
1) Development
2) Runtime

Development

The EsiObjects Development side contains all the development tools to complement and enhance the Application Repository Tools used for transformation of legacy data definitions and business rules to objects. Description of this environment is beyond the scope of this paper. However, it is critical to the object development lifecycle. Integration of commercial and homegrown tools into this environment offers the DOD a clear path into the future rather than a one-time solution.

Runtime

Client runtime systems can be varied. Government or commercial-off-the-shelf (GOTS/COTS) systems can be used as long as they support an object interface.

Additionally, any of the popular GUI development and runtime systems can connect to the server side simply by registering an OCX/ActiveX control, dropping it on a form and selecting the type of connection via the controls property sheet.

Server Side

The server side is simply illustrated as two environments:
1) Development
2) Runtime

Development

The Development Environment also contains the ART tool set. Additionally, all Class Development Tools (not shown here) are an integral part of the EsiObjects Development environment. Libraries of reusable classes are used to support ART. The ART tools set is explained in more detail later on.

Runtime

The Runtime Environment is produced from the Development Environment and is a deployable entity. It does not depend on the Development Environment components being present.

Data elements are exposed to the enterprise as properties (wrappers) of a class. This forms the public interface that client systems have access to. In Figure 1 above, patient demographic information is exposed through the Person class that contains the Name, Sex and DOB (to name a few) properties. These properties expose the respective data elements of the legacy database.

Application Repository Tools Functional Overview

The diagram below gives a more elaborate view of the Object Repository. It consists of three distinct environments that are logically linked by two functions: the FM Extractor and the Compiler.

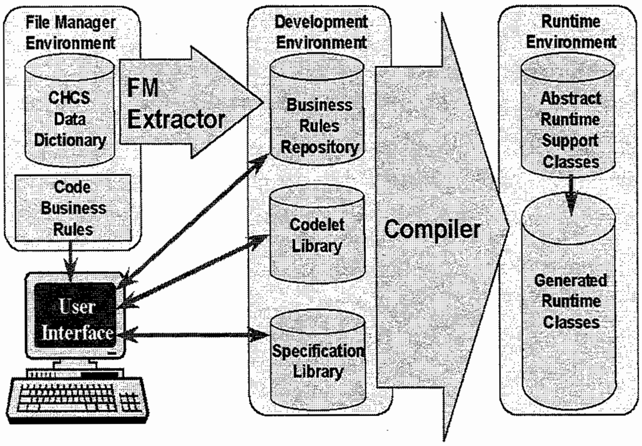


Figure 2

File Manager Environment

The File Manager Environment consists of the CHCS Data Dictionary and the CHCS applications code that contains the business rules. This environment must remain untouched and inaccessible at runtime. The data dictionary definitions and business rules are immutable.

Development Environment

This environment is object oriented and a part of the EsiObjects development tool set. It consists of a collection of classes that support three general aspects of the repository.

1) Business Rules Repository
2) Codelet Library
3) Specification Library

The Business Rules Repository is a sophisticated set of objects that stores the data definition of each data element as well as a collection of business rules that apply to that data element. The FM Extractor is a specific extractor that transfers the definition from the data dictionary on a data-element-by-data-element basis. The concept of a File Manager file is lost. However, the integrity of the underlying data structure is maintained.

Currently, the business rules are extracted from the code semi-manually. A GUI interface into the repository offers the programmer an easy-to-use interface for entering the rules. One of the ongoing enhancements to the tool set will be to develop parsing tools to expedite the process. Additionally, extractors can be written for different sources such as commercially available object design tools like Rational Rose™.

The Codelet Library consists of generalized codelets. The compiler expands it into real code. This code is specialized by the compiler for each data element based on its type. This library is built manually and used over and over in a different context.

The Specification Library contains all the mapping specifications required by the compiler. Basically, this library tells the compiler where to put the wrapped data elements – what class and property the data element is to be mapped into. Data elements in the legacy database are accessed via the concept of an EsiObjects property. A property contains chunks of code called accessors. Accessors correspond to components of the M language such as `Assign`, `Value`, `$Get`, `$Order`, etc. The code that is generated for each accessor is executed whenever an external reference is made to the object's property. For example,

```
S A%Name=A%OID.Name
```

will cause the Value accessor code to be executed for the property Name because the property reference is on the right side of the `SET` statement.

Runtime Environment

The runtime Environment runs independently from the development environment. It contains the object interface generated by the compiler. It consists of a common set of classes referred to as the Abstract Runtime Support Classes. These classes contain all the common methods that support the runtime environment.

The Compiler uses the Business Rules Repository, the Codelet Library and the Specifications Library as input. It generates property accessors by expanding generalized codelets and then maps them into specified classes. The classes are subclassed to the Abstract Runtime Support

Classes so as to inherit all the common support services.

Data elements in the legacy system are wrapped using virtual objects. This important feature permits aspects of the legacy system to be exposed as objects without the overhead of an actual object. Virtual objects have no instance variables.

Benefits

Some of the benefits of converting the existing CHCS system to an object oriented client server system are listed below.

1) Currently, the legacy system contains data element definitions and some basic business rules in a data dictionary. Unfortunately, the remaining business rules are imbedded in a large, monolithic code structure. Extracting all data definitions and business rules to a central object repository makes them easily accessible, extensible and maintainable.

2) Existing CHCS data can be accessed via object wrappers. These data elements look like any real object created within the EsiObjects environment. As a migration path, the DOD can add new, real objects to systems as a natural extension, giving it a clear path into the future.

3) The object repository will evolve into a seamless suite of tools that will integrate the modeling work done by the Object Oriented Analysis and Design (OOAD) team. Additional tools will be integrated into the environment as the project evolves, integrating the workflow.

4) Data within the M systems is exposed to the enterprise via standard, open middleware. This data can then be converged with other data, representing a homogeneous object view of enterprise data, independent of its source.

5) Last, but not least, all of the fundamental benefits of object oriented development become obvious. Objects are a way to deal with the complexity wall. One long-term benefit is that complex systems can be built and extended without hitting that wall early in the product lifecycle. Another big benefit is the concept of reusability. The consequence of reusability is that it shortens the development process and reduces testing cycles as well as maintenance time. This should lead to a decrease in the overall cost to the customer. Expending a little more time up front to implement published, generalized object patterns and frameworks pays off down the development path.

To quote the project leader of this endeavor; "I can't imagine doing this project without the powerful features of object orientation and the tools to implement it." ***M***

Terry L. Wiechmann is President of ESI Technology Corporation. He can be reached at: twiechmann@esitechnology.com.
Browse www.esitechnology.com for more information.