

Case Study: The Vanderbilt University Medical Center Information Management Architecture

William W. Stead M.D., Randall A. Bates, Jeff Byrd,
Dario Giuse, Dr.Ing., Randolph A. Miller, M.D. Edward K. Shultz, M.D.

Informatics Center, Vanderbilt University Medical Center, Nashville, Tennessee

I. Introduction:

Vanderbilt University Medical Center (VUMC) began to create an Integrated Advanced Information Management System (IAIMS) in 1991, when Dr. William Stead arrived in the role of Associate Vice Chancellor and Director of the new Informatics Center at Vanderbilt, and brought with him more than a decade of relevant informatics and IAIMS experiences. [1,2] Prior to 1991, Vanderbilt's strategy had been to select and deploy individual "best of class" information systems for admission/discharge/transfer and billing, dietary, materials management, pharmacy, laboratory, and radiology. These systems were used primarily to automate tasks. Admitting and charge data were exchanged through an interface engine to minimize redundant data entry. A limited database of billing data supported management decisions.

The VUMC IAIMS vision was to bring together people, data, knowledge, and tools at the place and time decisions were to be made. The goal of using tools to leverage data and knowledge for decision support represented a major shift from the prior focus on automation of tasks. The shift led to three critical early decisions. [3] The first was to invest in faculty in Biomedical Informatics, as a new primary academic discipline to conduct research and develop manpower, to translate the IAIMS vision into practice. The second decision was to create a novel organizational structure, the Informatics Center, to manage information as a shared resource, and to use VUMC's operational infrastructure as a laboratory to discover optimal interactions of people, process and technology. The third decision was to explicitly design an information architecture to manage information resources as enterprise assets separate from the applications that automate processes.

This case report describes the state of information management and utilization at VUMC a decade after these beginnings. It presents the architecture that has evolved and summarizes the functionality that has been developed, together with the impact of both upon the enterprise.

II. The VUMC Environment:

VUMC is an integral part of Vanderbilt University, which is comprised of ten schools (the College of Arts and Science, Graduate School, Blair School of Music, Divinity School, School of Engineering, School of Law, School of Medicine, School of Nursing, Owen Graduate School of Management, and Peabody College, Vanderbilt's School of Education), a public policy institute, the medical center's clinical facilities and The

Freedom Forum First Amendment Center. The university in 2001-02 has a faculty of more than 1,900 full-time faculty members; a part-time and clinical faculty of approximately 1,500; and, a staff of almost 12,600. Undergraduate students number 6,037 and there are 4,157 graduate and professional students.

VUMC's clinical facilities include the 678-bed Vanderbilt University Hospital (VUH), an 88-bed Psychiatric Hospital, the 75-bed Vanderbilt Stallworth Rehabilitation Hospital, and outpatient facilities in The Vanderbilt Clinic (TVC). Operating statistics for VUH & TVC for Fiscal Year 2001 are listed in Table 1.

Table 1. VUH & TVC Operating Statistics, 2001

Number of Staffed beds	678
VUH FTE/adjusted occupied bed	5.6
TVC FTE/400 visits	4.3
Mean number of inpatient discharges/month	2,657
Case Mix Index	1.75
Mean Length of Stay (days)	5.3
Outpatient Surgeries/mo	1,098
Number of outpatient visits/month	53,482
No. of visits to Emergency Department/day	175

III. VUMC Information Management Architectures:

An information technology architecture defines the philosophy underlying system development and maintenance, the components of a system, the boundaries of the components, and the communication among them. An architecture uses layering to reduce complexity, and modularity to increase flexibility while localizing change. Client-server configurations, mechanisms for mediation among clients and servers, and object oriented programming are examples of architectural strategies that increase interoperability and reuse; they are tools, not architectures per se. Such tools meet their objectives as long as all software is developed consistently with the overall architectural approach. However, to date, no single approach to developing applications has been adequate to support all of the information handling needs of a complex enterprise. Even if one could accomplish this, the success would be short lived as inevitable computer science advances produce better approaches that are incompatible with past implementations. Accordingly, such strategies are best categorized as *application component architectures* to denote their limited scope.

Each application, or suite of applications developed to a common application component architecture, is a discrete unit. It can have its own data model, business rules, and database. When more than one application component architecture is required by an enterprise, the increase in functionality comes at the price of potentially redundant implementation and maintenance burdens, potentially inconsistent definitions of content across applications or application suites, and inability to resolve such discrepancies. Interface standards can reduce the problem, but cannot solve it unless the meaning of messages exchanged is explicit and independent of the sending and receiving systems.

At the existing state of the art, an interface is merely a path between systems. Each receiving system must be taught through individual programming to be able to appropriately interpret data elements from each sending system in order to integrate and utilize the external information in the context of its internal environment. When each participating system manages its database independently, there is no way to insure they are synchronized. In other words, a collection of databases underpinning interfaced applications does not equal, and can never exceed, the sum of the parts.

III-A. Enterprise Information Architecture

VUMC specified an *enterprise information architecture* to separate the management of corporate information assets, such as data definitions, business rules and patient data, from the transaction processing systems that support operations. The enterprise architecture also encompasses important philosophical principles, such as “do not develop components that are not consistent with the ideal architectural configuration unless there is a viable known path that will go from the developed state to the ideal state in a reasonable amount of time”, “do not devote scarce developmental resources to build an information resource or a software program that could be affordably purchased in the commercial marketplace”, and “all electronic clinical information exchange at VUMC will be routed through the Generic Interface Engine (GIE)”. Data transfer among applications is managed proactively at the enterprise level (via the GIE), insuring the degree of synchronization appropriate to the logical unit of work. From the beginning, it has not been practical to stop and rebuild the VUMC applications and infrastructure to match the idealized information architecture. As noted above, the information architecture specification serves as a planning tool, guiding each successive implementation to approximate its principles to the degree possible.

Figure 1 is a logical view of the VUMC information architecture. It permits vendor applications and locally developed informatics tools to operate as components supported by a shared set of information resources and repositories.

The purpose of the VUMC *enterprise information architecture* is to decouple the management of content from the applications or tools that provide functionality for users. A key tenet of the architecture is to represent meaningful VUMC content outside of the various application systems, and to align the applications by importing and using this externally defined content in a standard manner throughout. Information, such as metadata and organizational knowledge that might otherwise be entered into application-specific master files, is externalized in generalized tables. This information is structured to make its meaning explicit and accessible; for example: external tables store the identity of medical center personnel and a mapping to their roles; clinically meaningful orderables are stored externally, with mapping to the administrative equivalents in individual ancillary systems; and, the set of clinical concepts that can be measured in the laboratory is stored externally, with mappings to the various billing codes associated with each concept in ancillary systems. Certain applications are able to use these externalized tables directly. It is often necessary to manually copy the information into the profiles of legacy systems. In either case, each new application reuses prior definitional work. Only

newly required information needs to be added to the generalized tables, and the relation of such new information to existing information can be made explicit as it is added. This approach saves implementation time while pre-aligning meaning across otherwise disparate applications.

Similarly, data that are captured or managed by an application, but which are used by more than one application, are externalized into generalized repositories. A set of disparate repositories exploit the strengths of their respective technologies. For example, highly structured, coded clinical data is represented in relational tables, and in contrast, an indexed text repository, organized according to a document paradigm, provides a single logical source for all clinical reports about a patient, be they binary data, images, or text. Some reports are stored in this repository as symbolic links (e.g., links from textual radiology reports to their corresponding images in the Picture Archiving and Communication Systems), while others are copied directly from primary sources and stored directly in the repository, as in the case of EKGs.

The indexed text repository is a non-relational, hyper-indexed database implemented in Perl on a distributed processing system. The lowest tier, known as the Star layer, implements distributed processing, queue-based transaction processing, process control and monitoring, and inter-process communication. The database layer, known as StarChart, implements permanent data storage, automatic replication across servers in different geographical locations, and conversion of clinical data from all sources into a common internal representation. Common views such as the assembly of documents and data related to a patient into a browsable electronic chart are cached to reduce search demands. The application layer implements transaction and business logic, such as the handling of corrections and updates in stored documents, and the handling of different evolving stages of individual data items (from pending to preliminary to final to corrected report, for example). This layer is shared by all applications that use the repository, and hence provides the single place where transaction and business logic is maintained and applied. It provides request broker functionality to support application interface services, report distribution services, and a number of Web-based interfaces.

The externalized repositories leverage database techniques to solve a class of problems that are difficult to handle through data processing strategies characteristic of transaction processing systems. For example, a common approach to an enterprise master patient index involves recording each facility caring for a patient, and for each facility, the patient's medical record number. When a facility issues a duplicate number in error, a complicated process is used to reconcile and merge the records. One of the VUMC repositories is the Enterprise Patient Index, a table of identifying numbers, a table of names, and linkages of those numbers and names to instantiate people. As mistakes are made and corrected, linkages are updated. An SQL query is all that is needed to assemble all record "fragments" for a patient.

In addition, the externalized application-independent repositories serve two types of middleware function. First, when an application combines two concepts into one variable, the meaning can be decomposed into a set of granular data on the way into the

repository, or assembled from multiple data on the way back into the application. In this fashion, required translation is limited to a “plug-in” between the application and the repository without burdening other applications. As applications converge around common metadata, the plug-ins are removed. Difference in the use of the concept of case, encounter, and visit among applications is a simple example where it is helpful to disambiguate through the repository. Second, when two different processes provide different views of a related datum, those views can be represented “side by side” instead of picking one or the other. For example, the admitting office may be responsible for updating the attending physician field that is used for billing purposes, while the clinical care team on the patient’s ward may represent the most reliable source of this information. However the ward team does not possess the admitting office’s understanding of the correct timing of recording changes to comply with billing requirements. The solution is to record both views of the data (administrative and clinical versions of “attending physician”, and to have a process for reconciling differences just before midnight, the deadline for billing corrections).

Application components connect to the collective externalized tables and repositories through a single logical point, serviced by communication management engines. The Generic Interface Engine (GIE) utilizes IBM’s CICS™ as a transaction-processing environment. The GIE provides transaction logging, protocol conversion, one to many routing, and request broker functionality. It uses queuing mechanisms to loosely couple inter-application communication. At the same time, it manages acknowledgements so as to insure serialization and logical unit of work across components. Since the proactive end-to-end management of interface transactions provided by the GIE requires application-specific development, a commercial interface engine provides an alternative path for less demanding situations. In addition, efficient query services exist to provide applications with access to some commonly used repositories for transactions that do not result in updates.

Whenever possible, common tools provide generic functionality across the enterprise. A Clinical Workstation (CWS) desktop is used for shared devices in high volume clinical areas. The CWS is “locked down” to minimize support problems and enables use of tools such as ZenWorks™ to distribute updates from a central management point. It includes an authentication service that enables the user to sign on once to the CWS, which in turn signs them into the applications they select, avoiding the time required to sign on to multiple independent applications.

WizOrder (developed at VUMC [4] and marketed by McKesson as Horizon Expert Orders) is the clinical decision support and order capture interface for the inpatient services. Through order sets, protocols, and guidelines developed and maintained by local clinical experts (who utilize the literature and national guidelines), WizOrder brings together information about the patient with information about best practices appropriate to the clinical context. As the provider makes decisions in this supportive environment, WizOrder translates them into actionable orders, which are in turn distributed as appropriate by the GIE.

PathworX is another tool linking pathway management, standardized nursing documentation, and variance tracking. [5] PathworX enables customized implementation of best-practice care protocols, cueing caregivers to perform and document needed services. By tracking variances in achievement of therapeutic goals, PathworX identifies the services at particular phases of particular pathways that were less effective than anticipated. Goal variance data support the generation and testing of hypotheses for continual quality improvement.

Web based interfaces to StarChart [6] provide access to the electronic patient chart and support for related workflow. A one-patient-at-a-time interface provides access to the integrated clinical picture for one patient. A panel management or practice-oriented interface allows entire sets of patients to be accessed and reviewed in a single logical operation, and includes a result notification component. A number of additional interfaces are used for data entry, validation, document submission, document signature, report generation, and ad-hoc querying.

To facilitate rapid development, a general set of services is made available. Due to different client needs, sometimes multiple methods are available to access the same service (e.g. Java RMI, HTTP, CORBA). All these access methods call the same code however. Representative services include:

- Enterprise Patient Identifier Service: Data to include additional fields such as social security number, date of birth and address information. Increase query functionality by supporting queries with multiple partial fields (e.g. partial name and medical record number).
- Census Service:
 - Inpatient - Service to query inpatient census information by location.
 - Outpatient - Service to query outpatient census information by location, attending, medical record number, patient name
- DocInfo, EJB: Service to allow clients to query physician information using a variety of parameters (Name, SignOn, ResourceId).
- OrderableFinder: Service providing a route to identify what clinical items/functions can be ordered.
- ICDFinder: Searches ICD-9 code for a diagnosis.
- ComplianceInformationService: Matches CPT to Medicare compliant ICD.
- Medication Lookup: Service to allow lookup of a medication, potential doses, routes, frequencies.
- CORBA Interface Layer: To support compiled applications (like Visual Basic or C++), a CORBA interface layer has been added to the above services. This layer provides protocol translation only. The request is then delegated to the EJB service layer.
- Big Brother Monitoring: Added test and reporting functionality to integrate with a public domain monitoring service, BigBrother.

Vendor “best of class” transaction processing systems continue to operate as components within Vanderbilt’s information architecture. However, such “legacy” systems are

acquired and used in the context of the overall architectural philosophy, not just to address a specific task. For example, a vendor system that is otherwise “best of class” will not be purchased if it is unable to interface with other important aspects of the overall architecture. Major vendor systems include:

- McKesson HealthQuest Patient Management for inpatient ADT, outpatient surgery ADT, ED registration; Healthquest Patient Accounting for technical billing; and HealthQuest Medical Records for case abstract and chart tracking
- Epic Cadence and Resolute for outpatient clinic scheduling, registration, and professional billing
- TripleG for clinical labs, microbiology and anatomic pathology
- Hemocare for blood bank
- IDXRad for the radiology information system
- AGFA for radiology picture archiving and communications (PACS)
- McKesson Series system for inpatient pharmacy
- IPATH ORMIS for operative services
- Eclypsis for management decision support

The use of these transaction processing systems is restricted to workflow within an operational area such as the laboratory. Data of interest to clinicians, educators, or administrators outside of an operational area is externalized to the repositories where the data can be accessed by appropriate tools that integrate into other workflows. Although it is increasingly possible for such applications and tools to share directly a single data repository, it is useful to treat the “application” database and the external database repository as distinct architectural layers to isolate corruption and to separate management of data that is work in progress from data that is in publishable form.

For example, laboratory test orders generated by WizOrder are printed on the patient's ward (as a part of the "requisition generation" process) with a bar coded patient and test identifier, used for sample collection on the ward. At the same time WizOrder generates the order, an electronic copy is sent via the GIE into an enterprise repository. Specimens arriving in the laboratory system are processed via LabTalk2, a middleware application that reads the bar code on the requisition, obtains the corresponding detailed order information via the GIE from the enterprise repository, passes required test-related and patient-related information into the laboratory system (via the GIE), and returns (via the GIE) an "in lab" status to update the repository. As results are generated in the laboratory system, they are handed to the GIE for transmission to StarChart for result reporting and escalation. Because all tests that are ordered do not result in specimens delivered to the laboratory, the foregoing system eliminates the need for a complex interface queue management between the order capture system and the laboratory system. It also provides a status tracking mechanism for all stages of laboratory test processing, from ordering, to performance, to resulting. Since integration is provided by the enterprise-wide ordering and repository layers, the laboratory has the freedom of using multiple simple systems that rely on the enterprise architecture, rather than requiring a large, complex, vertically integrated, and expensive laboratory information system.

III-B. Application Component Architecture:

VUMC has adopted application component architecture standards for internal development of enterprise services and purchased software where practical.

The simplest user interface is used that meets the application requirements, starting with static HTML pages created offline; progressing to server generated pages; then to pages that include Java applets; then to pages with compiled ActiveX components; and finally to a locally installed Java, C++ or VB client application.

The preferred framework for development of rules is the Enterprise Java Beans (EJB) running under BEA WebLogic. CORBA OMG may be used as an alternative to the native EJB client access protocol.

Transactions that update the data repositories are routed through one of the communication engines described above, coupled with a transaction-processing monitor when practical. Reads of the data repositories are often direct from the database for performance reasons.

Oracle and DB2 are the enterprise structured data stores. Use of triggers and stored procedures is discouraged. Use of Java coded stored procedures is the preferred compromise since the actual Java procedures can be moved to the middle tier as a migration path if the database management system changes.

Decision support is provided by a set of inter-related tools from four functional areas: data storage, metadata, reporting and access. Data is extracted from departmental financial, clinical and research systems and stored in an Oracle enterprise data

warehouse. Higher level data is aggregated and served to users by an array of Microsoft based datamarts. The array of datamarts on smaller servers allows scalable reporting for departments while maintaining high performance for real time analysis. Data about the contents of the warehouse, the datamarts, and current reports are stored and web accessed from a MetaData Repository, which was created using Lotus Notes. This is used by both technical staff as well as end users to see what data elements, transformations, and reports exist. Tiered levels of reporting and analysis are provided. Static reports can be in any format, and currently include reports generated from Crystal Reports, Microsoft Excel, Business Objects, TRMS and Adobe PDFs. Sets of logically linked web based executive reports are published using ProClarity. Web based ad hoc reporting is done using Business Objects. On Line Analytical Processing, with data cubes, uses a combination of Microsoft OLAP Services and ProClarity clients. All reports and metadata are accessed through a web based information portal, which also hosts departmental documents such as calendars of events, standard procedures, notices, etc. This portal was constructed using Lotus Notes, but is accessed using any web browser.

IV. Results:

The information management infrastructure developed at Vanderbilt is in use throughout VUH and TVC. Indicators of its penetration include:

- Shared registration data across all inpatient and outpatient areas.
- Shared electronic patient chart across all inpatient and outpatient areas, contains about 150 document types, and is used interactively 9,000 times /day by 2,200 distinct users.
- All inpatient units except the neonatal ICU and the Clinical Research Unit utilize WizOrder for all categories of orders, generating seven to ten thousand orders per day, on average 70% directly by physicians.
- 50% of inpatient units utilize PathworX.
- There are over twenty thousand reports on the portal, and over two thousand new reports are automatically generated quarterly.

Indications of the architecture's acceptance by the institution are reflected by two recent decisions:

- To complete the conversion to filmless radiology by the end of 2002.
- To take the remaining paper-based processes out of the clinic by the end of 2002.

However, the change in organizational culture that it is nurturing is a more important indicator of its effectiveness. Access to data about VUMC practice patterns is being used to bring home to both clinical leadership and to individual physicians the fact that practice variability is a major fixable problem. For the first time, the organization has the will to systematically introduce evidence into practice. The Pharmacy and Therapeutics Committee is going beyond limiting the formulary, to recommending situations where drugs should be substituted because of increased efficacy and decreased cost. The Care Improvement Committee is developing complex treatment advisors for interventions such

as the initiation of anticoagulation. The Resource Utilization Committee is specifying which imaging procedure is appropriate in a particular clinical situation. As knowledge such as these practices is developed throughout the organization, it is represented in the generalized external tables, and then linked into workflow by the appropriate tool to make it available when it can have an effect. The resulting change is tracked as a by-product of use of the tool, leading to refinement if needed.

V. Conclusions:

Based upon the experience at VUMC with an *enterprise information architecture*, the following principles may generalize to other sites:

- Externalize data and business logic from “legacy” systems in application-independent repositories. A “legacy” system is one where meaning is implicit in code or process, or where data retrieval and storage are restricted within the system. The content of the repositories should be self explanatory.
- Adopt industry standards where they exist, such as the HL7 messaging standard. If an existing resource is the best surrogate for a standard (such as SNOMED for clinical concepts) extend it. Where a resource is grossly inadequate (such as ICD for medical problems) map to it.
- If there are multiple sources for an item, represent both versions. An item has multiple sources if it is acquired through different processes with variation in meaning or accuracy.
- View individual systems as components (transaction processors, data capture devices, etc.); not as data repositories.
- Centralize global functions (ordering, notification, etc.) that can scale up and provide consistency across the enterprise. Role-specific user interfaces can shield the user from the multitude of transaction processors needed to carry out their decisions.
- Use interface engine technology, augmented to support logical unit of work among distributed components, to mediate and to avoid a direct interface for each combination of participating systems.
- Support for evolutionary development and adaptation is more important than getting it “right”.

References:

1. Stead WW. Systems for the Year 2000: The Case for an Integrated Database. *MD Computing* 1991;8(2):103-110.
2. Stead WW, Borden RB, Boyarsky MW, Crow DS, Mears TP, Stone AA, Woods PJ. A System's Architecture Which Dissociates Management of Shared Data and End-User Function. In: *Proceedings 15th Annual Symposium on Computer Applications in Medical Care*, 1991:475-480.
3. Fast Track to IAIMS: Transition from Planning to Implementation. *Journal of the American Medical Informatics Association* 1996;3(5):308-317.
4. Geissbuhler A, Miller RA. A New Approach to the Implementation of Direct Care-provider Order Entry. In: *Proceedings of the 1996 AMIA Annual Fall Symposium (Formerly SCAMC)* 1996:689-693.
5. Ozbolt, J, Brennan G, Hatcher I. PathworX: An informatics tool for quality improvement. In: *Proceedings of the 2001 AMIA Fall Symposium (Formerly SCAMC)* 2001:518-522.
6. Giuse, DA, Mickish A. Increasing the Availability of the Computerized Patient Record. In: *Proceedings of the 1996 AMIA Annual Fall Symposium (Formerly SCAMC)* 1996:633-637.

Captions:

Figure 1:

Logical view of VUMC Information Architecture. a) Enterprise information assets are maintained in externalized “master files” and repositories; b) Global knowledge resources are linked in via the Internet; c) Access to and update of repositories and logical unit of work are managed via a communication subsystem; d) generic applications support functions such as patient look-up, ordering, reporting and escalation across the enterprise; and e) niche applications are pre-aligned and reduced to components to support departmental work flow.

