Healthcare Information Architecture: Elements of a New Paradigm

Daniel J. Essin[1] and Thomas L. Lincoln[1,2]

[1] University of Southern California, [2] The RAND Corporation

An Electronic Medical Record (EMR) must provide a secure, permanent archive for an individual's medical records and also function as a multi-purpose database that supports the complex, varied activities of patient care. Meeting these objectives requires unusual flexibility in how data are retrieved and processed. Semantic and referential integrity must preserved both over time and as chunks of information are exchanged with other systems. Relationships between data entries must determined dynamically based on actual events, rather than statically through application design. Distributed data requires that new forms of system security be incorporated into an EMR at a structural level, with an emphasis on the labeling of elements to be secured behind a security barrier, with audit trails to document necessary overrides and monitor for suspicious use. A modular information architecture is proposed that integrates requirements for structure, content, processing and security.

1 - Introduction

To effectively reform healthcare, a paradigm shift will be required in healthcare computing. To meet new requirements, we will need new data management systems that are not merely a superficial rearrangement of existing hospital information systems.. Despite some computerization, the traditional paper-based medical record continues to serve all aspects of clinical care. It represents a kind of primitive blackboard system that passively organizes each patient's care and facilitates the solving of medical problems. In this sense, it both documents and communicates. Problems and their solutions are formulated on the chart, and various care providers consult it in order to coordinate the process. Thus some steps are procedural and some are cognitive. The paper chart also has some notoriously awkward characteristics. It is available in only one place when it is often needed in several simultaneously. It is fragmented, particularly with respect to imaging data, and is insufficiently indexed, with no single ordering satisfactory for all purposes. It is also often illegible.

Creating an electronic medical records system (EMRS) that can satisfy this same wide range of uses as the paper chart presents both a specific and a generic challenge to computing science. The task is to turn this classic paper source into one that will relieve the evident shortcomings without introducing new complications -- such as unwanted access -- all the while retaining the many advantages of the paper format. It is comfortably structured as a collection of documents, is able to encompass the variability and complexity of medical phenomena and health care practice, can be perused with minimal procedural navigation, it is portable to all venues, and it constitutes a single permanent legal document, appropriately signed by those responsible. This is a tall order, but we believe it to be possible using today's technology, given some decisions about certain policy parameters, plus some directed research and development.

To the present, attempts to create an EMR have fallen short. The most important reason for failure has been the assumption that clinical activities can be redirected into machine oriented formats and that the various rigidities introduced for the convenience of the computer will not interfere with clinical work. This assumption ignores the heuristic value of the approaches to information management embodied in the paper record. They are not arbitrary, but have been refined over time to deal with difficult issues. Thus, as advocated by Donald Norman in his book for a general audience: "Things That Make Us Smart: Defending human attributes in the age of the machine" [1], success of an EMRS will depend upon supporting the flow of clinical work as it is most effectively accomplished by numerous participants, through a careful choice of data structures and of the underlying architecture.

2 - Requirements

The behavior an EMRS should exhibit is complex: 1) It must provide a rich method of representing information so that content, meaning and context are not obscured, insuring that the raw data is not prematurely replaced by interpretation or conjecture. 2) It must be "open" so that a wide range of information management appliances (applications), each with its own set of functional requirements, can use the information as a resource. 3) It must inform users about the nature of the information that it contains. 4) It must be able to selectively retrieve information, either for human viewers or to serve as knowledge sources for automated process- control and decision-support systems. 5) Since different groups of users each have their own agenda and preferences, there must be great flexibility in rendering the information for presentation. 6) It must store the data in ways that meet permanence regulations. 7) It must structurally address the issues of privacy, confidentiality and security.

The challenge to information scientists is to devise an information architecture that will address these requirements. The first step is to isolate basic properties that can be combined to create systems that exhibit the desired behavior (Table 1).

Table 1
Properties of Database Systems Designed to Store and
Process Medical Records

Atomicity	Semantic Integrity	Flexibility
Authenticity	Security	Processability
Persistence	Performance	Interoperability

Atomicity

Each entry placed into EMRS should be self-contained, i.e., atomic. It must contain sufficient information to remain informative if removed from its host environment, and its authenticity must be preserved. Each entry must be registered using a time-base that is sufficiently fine-grained to allow an accurate chronology of events to be constructed. This is especially important when many participants are adding items concurrently in response to a single external event. In a certain sense an entry is an object.

Authenticity

All entries must be unalterable [2] and permanently archived. Each entry must be preserved, as it was entered, in order to meet medico-legal standards. Each document must be sealed with some type of encrypted checksum so that it can be verified that no changes have occurred since the document was committed to storage.

Updates are not permitted once documents have been committed. Corrections must be appended as new documents. Ordinary retrieval processes will only display those entries which, taken together, constitute the "official" correct record. Audit trails must be included, so that, with appropriate permission, the entire record can become visible, including those entries that have been superseded.

Persistence

The period during which legal unalterability must be ensured is over 20 years in the case of records that document care to infants but may be shorter in the case of adults. With the current interest in a "lifetime medical record," individual documents may have to be maintained for over a century.

Database technologies that require that the data be periodically copied and/or reformatted as part of system maintenance and database restructuring would violate the unalterability requirement. However, as long as the original is never altered, working copies could be made freely since they could always be verified for accuracy against the original. This suggests the use of robust write-once media for the originals.

This requirement also implies that the data stores are external to and independent of any particular processing environment. Data stored internally to a specific application or platform cannot be accessed directly by others and even complicates modifying the original applications as their requirements evolve.

Flexibility of Representation and Retrieval

The document structure must freely accept descriptive material of arbitrary length and it must be possible to qualify or annotate any or all quantitative items in the document.

Entries must accommodate fuzzy information such as approximate dates and times, information for which only qualitative definitions exist and statements of opinion. Retrieval functions must produce useful results even if Information is missing. Information may appear to be missing if, for security reasons, is unreachable in the absence of special access authorization.

The data contained within the persistent data store should be structured so that any conceivable query can be expressed as a first-order expression against such a database. The semantics of the data and the database must be explicitly recorded as part of the database and must be easily discoverable.

This is necessary because, although the typical database can be queried for a list of relation names, there is no way to determine their semantic nature or relationship [3, 4], because traditional data management techniques hide the semantics of the database from the users. Today non-technical users are unable to query most databases because much of the knowledge of the meaning of individual attributes or relationships is implicitly embedded within the logic of the retrieval programs. Furthermore, there is no way to determine how many relations exist within the database that might contain information relevant to a particular inquiry. When role or relationship information is confounded by being present in the names of both relations and attributes, semantic heterogeneity increases. In order to avoid obscuring the semantics of the data, one has to consider the database as a whole [4]. These authors assert that the first order normal form (1ONF) in which the database consists of a single relation and contains specific slots (i.e. attributes) that hold the information, would have been represented as by relation and attribute names if the database were in some traditional normal form, e.g. 3NF. They further state that any conceivable query can be expressed as a first-order expression against such a database. We take the matter even further, and assert that this concept can be applied to non-normal form databases (in which each entry is arbitrarily complex), provided that there is a mechanism to apply the first-order expressions to the output of intentionally defined functions that can be applied to the data.

Both developers and users need adequate tools to help them explore the semantics of the database and to determine what terms have been used before, and in what context. As the volume of stored data grows, discovering the semantics of past and present data models becomes an increasingly difficult task. But a lack of this capability leads to Keyword Drift [5], a phenomenon whereby the semantics of an application wander over time. Users who do not have good information about what terms are currently active continually invent new keywords and new rules for categorizing and indexing the same information that they have coded before. As old data become unrecognizable through this process, they become fossilized and unusable -- effectively non-existent for ordinary purposes.

Semantic Integrity

Medical documents make frequent reference to data that are coded and/or maintained by ancillary systems. In a "properly normalized" relational database, a medical document would store only the appropriate foreign keys needed to join with the relations containing the explanatory detail. However, many coding schemes change from year to year and often retain the same code numbers even though the underlying definitions have been altered. It is not always possible to insure that the necessary systems (or versions of systems) will be on-line to satisfy a relational query at the time a document needs to be viewed or copied to an outside agency. Therefore, all information that is necessary to insure the semantic integrity of a document must be copied and stored in the document itself at the time it is committed to storage. The intent is to copy just enough information to preserve integrity (readability and context) of the individual entry.

Interoperability

Documents transferred (or accessed) between sites, or used at the same site at different times, must be interoperable (processable at the recipient site and informationally equivalent). It must be possible to access the information content of documents, independent of the nature of the host system or in the absence of any sophisticated data manager (i.e. humans can read them with a low-level disk editor if all else fails).

Processability

Each document must also include meta-information that describes its semantic content and organization. This information must be computable and accessible through queries. Existing documents may be candidates for inclusion in queries or new transactions on the basis of an arbitrarily large number of rule-based criteria. Similarly, the result sets produced by arbitrarily complex transactions must be accessible through query languages and application programming interfaces (API) so that they can be used as input to other queries and processes.

Performance

The speed with which database operations can be accomplished is always an important non-functional requirement. Slow responses commonly violate the cognitive tempo. In addition, there are many medical situations in which rapid access to information is critical.

Security

In order for the healthcare process to be most effective, the medical record must contain accurate and complete information that reveals the details of people's lives and their medical histories, what was done for them and why and who was involved. In order to elicit the maximum detail, each participant must feel confident that the information will not fall into the wrong hands and be used against them. For this reason, just as there are legal requirements for record retention, there are legal and ethical requirements that the records be kept secure and confidential so that each individuals privacy is preserved.

3 - Information Representation

None of the requirements or properties discussed above addresses the structure of the atomic unit of data storage. It is clear that in order to treat this disparate but highly inter-related data as a single resource it must be unified into a single structure that contains not only the data but a variety of semantic information (meta-data) to guide its subsequent retrieval and use. We theorize that the atomic unit of storage should be an encapsulated complex object with specific structural properties which we will now describe. We hypothesize that objects, so constructed, have the properties necessary to enable this unification. We call these objects Loosely Structured Documents [6]. The term "loosely structured" refers to the fact that there may be wide variations in content and modest variations in structure within individual documents without obscuring their similarity to other documents of the same type.

The accumulated details that can be found in a collection of medical records exhibit complexity that is unbounded. The information may come in hundreds .[1] of formats and the content differs widely depending on the domain from which these data originated. Viewed from a somewhat greater distance, the paper medical record is a collection of separate loosely structured documents [7,8]. Some data is highly quantitative, often organized in a tabular format. Some information is semi- quantitative data and is commonly collected using questionnaires and check lists. Records of interviews are almost entirely narrative. The most common records, those documenting ambulatory care encounters and admission to a hospital combine quantitative, semi-quantitative and narrative components into documents that have a loosely structured quality. Headers are in reality labels (or tags) that identify the content of different sections (such as Heart, Lungs, Impression, etc.). Some entries include logical links to physiological monitoring data and/or image data that are stored in other places. Within each type of form, flowsheet or document, some well established convention is used to structure the information. A variety of these forms are kept handy to that the users can easily switch between variants that organize the information differently or that impose more or less structure as each case dictates.

Structured documents have become a familiar convention. TEX and WordPerfect use internal markup to denote formatting and style regions. CLOS (the Common Lisp Object System) and various frame-based knowledge representations define slots within objects. Boxer, a computational medium for elementary school students, creates structure with nested boxes [9]. Tex and other markup languages also include conventions for representing arrays and tabular data structures as streams of text with embedded tags to denote the position of each datum within the array. The Standardized Generalized Markup Language [10] derives its openness and flexibility from the use of meta-level descriptors of document structure. Each of these markup conventions is intended to introduce a structure into data in order to enhance its ability to be processed computationally.

The internal structure of each document type found in the medical chart has many of the characteristics of the machine processable structures mentioned above, i.e. the structure usually is (or can be) indicated by topic headings inserted into the text. Tags such as CC: (chief complaint), and PMH: (past medical history) are immediately familiar to all practitioners and isolate specific regions of content. In effect, these tags constitute a markup language that emphasizes medical content. Missing tags imply the absence of significant material (in the opinion of the original observer). Other tagged sections may be optionally or conditionally inserted into specific documents in much the same way that a paper record may contain an annotation in the margin. More importantly, with appropriate tagging, highly structured tabular data can be represented using the same conventions.

Dayal [11] discusses documents as an example of complex objects and identifies a number of requirements for managing data objects with a complex internal structure. Complex objects are "highly structured objects that are composed of other objects." For example: "a document may be composed of sections ... and the sections themselves may be composed of section headings, paragraphs of text, and figures." In this sense, medical charts, and their component entries, are clearly complex data objects. "In many applications these complex objects are the units for storage, retrieval,

[1] A sales brochure for [6] offered "800 useful nursing forms"

update, [and] integrity control.... The most fundamental requirement of a complex object is that the user be allowed to manipulate it as a whole." Attempting to store medical records in conventional (e.g. relational) databases results in each object being reduced to a number of tuples scattered among a variety of tables. Because "there is no way to specify to the DBMS [database management system] that all of these linked tuples form a single complex object" operations on complex objects require complex sequences of relational commands.

Since objects can be arbitrarily complex, the potential number of relationships between objects is potentially unbounded [11]. Therefore, Dayal suggests that databases provide a general facility for specifying relationships between complex objects and/or their components in terms of functions defined over sets of complex objects instead of being limited to a small number of distinguished relationships with fixed semantics as is common in relational databases.

Objects, whether simple or complex, may have attributes that are not recorded directly within them but which must be derived indirectly from other data. This implies that data models in general should be extended to include the capability to return information by inference (but, in certain instances, also to block it). In other words, the results of database queries may include data that is not ever stored in the database but is derived dynamically from indirect sources or that is computed by arbitrary procedures. Scientific and medical databases have a corollary requirement - the ability to view complex objects at different levels of abstraction. This capability can be obtained by defining views over the output of retrieval functions alone or in combination with values derived from extensionally defined functions that are stored within the database.

Dayal [11] notes that "in some cases, it is too expensive to compute every intensionally-defined function on demand (i.e. at query execution time). It may be cheaper to precompute and cache its values instead. For querying purposes ... the function may be treated as being an extensionally-defined function. However, updates to the function's arguments may cause the cached values to become obsolete, requiring propagation of the update."

4 - The Proposed Architecture

Structure

The above requirements emphasize the need to treat the persistent data store as a discrete entity, separate from any application. In order to translate those requirements into an implementation a convention for structuring the documents must be adopted that can 1) accommodate variations in complexity, 2) allow application and knowledge evolution, 3) provide interoperability and open, self-describing semantics and 4) allow a wide variety of domain specific applications share, and be applied to, the same data. One candidate for a structuring convention is HyTime - the Hypermedia/Time-based structuring language (ISO/IEC 10744-1993) [12]. It is built on the Standard Generalized Markup Language (SGML) (ISO/IEC 8879-1986) and introduces two abstractions that together provide a notation for defining a generalized hierarchy of occurrence types and a means of recording them. Using SGML, the internal structure of documents are specified by Document Type Definitions (DTD) - formal, computable statements that describe how documents will be structured and what mandatory and optional components will be present. The first level of abstraction is the DTD itself. The syntax of DTD's is expressed as a nested set of elements. Each element has its own generic identifier, an optional set of attributes and attribute data types, and a BNF-like production stating what sort of data can be placed inside each element or level of the element hierarchy [12]. The second level of abstraction is provided by HyTime's architectural forms. Architectural forms are element meta-declarations that define the elements that can appear in DTD's or meta-DTD's.

Architectural forms define the class hierarchy of documents that can be entered into the data store and thus, in this case, distinguish a system as a medical system. At the architectural form level, developers can specify the structure of the various components of a document, i.e. "who," "what," "where," etc. These elements can then be assembled as necessary to create the DTD's that will actually control what information is collected and how it is rendered (displayed). An architectural form specification defines the minimum information that must appear, any anticipated but optional information that may appear, and how any additional notations should be "marked up" so that they can be located and classified. It must specify what rules in the knowledge envelope of the system can be used to validate input and which rules define the syntax that can be used to enter and flag nonconforming data and annotations. A given architectural form may be used by zero, one, or many DTD's. Forms that are implemented by zero DTD's function as abstract types from which subclasses can be derived.

Application Independent Resources

It is not sufficient for the semantics of documents to be open. For a given domain, e.g. the EMRS, all applications accessing these documents must apply consistent logic during processing in order to maintain semantic integrity. This requires an explicit mechanism for creating data transformation and retrieval functions. This must be done at the domain level so that all applications in the domain can share them. There are a growing number of systems that do this. Hypercard and a variety of other Macintosh applications can share XCMD's. Dynamic Link Libraries in Windows, NT, and OS/2, are language independent and can be shared any application capable of calling them. Stored procedures in SQL databases and Remote Procedure Calls in various UNIX systems address the same need for application independent, system-wide resources. All applications using such resources achieve consistent behavior and, as a consequence, consistent semantics.

Under this model, application development would have two components. 1) The persistent data store -- the document types and their semantics and the layer of function resources must share an evolutionary development path. 2) User-interfaces -- query languages, API's and wrappers (temporary encapsulations of process-related data in non-document form) may be application-specific. The relationship between these components is represented diagrammatically in Figure 1.

Application Independent Database Structure

The requirement for interoperability anticipates that future systems will rely heavily on distributed processing. Workstations will perform computationally intensive tasks that are departmentally or functionally specific in nature. Logically, this implies that portions of the database will be accessed frequently by some applications and rarely (or never) by others. Security, confidentiality and network efficiency will each be promoted by logically partitioning the database and distributing certain portions to the site of most frequent use. Federations of application specific systems will replace the monolithic information systems of today. If the requirements for atomicity and processability are met, it will be possible to aggregate the data when necessary.

Meeting the atomicity and persistence requirements separates the traditional problem of database concurrency control into two parts. Since no updates to existing entries are allowed, all users are free to add new records at will. Entries that are intended to correct other entries may produce user views that only display the most current information.

The atomicity requirement also supports fault-tolerant, high-performance designs. Since existing records are never physically updated or deleted, the process of replicating the data to remote locations can be approached in a more leisurely fashion. It will be possible to queue transactions requiring replication, establish priorities, and even temporarily suspend the process if there is a physical disturbance on a portion of the network. Parallel processing [13] and blackboard systems [14] are almost accident byproducts. Resilient medical systems have high availability requirements. Many are expected to be "up" continuously. This requirement is incompatible with the current generation of DBMS and operating systems that require periodic "down-time" for maintenance and the installation of new software versions. Externalizing the persistent data store offers a wide range of new opportunities to design methods that can provide continuous access to the data when various applications, or system components, are taken off-line for testing, modification or repair.

Creating an Open Environment for Application Development

The layered architecture allows application development to proceed asynchronously and in parallel on many fronts overcoming a traditional bottleneck. To allow this parallel activity without elaborate coordination, applications will use the DTD's and other explicit representations of database semantics as the control mechanisms. Because properly constructed DTD's are computable, this approach should detect and eliminate designs or actions that would violate system integrity. Semantic checks can occur both in compilation, for the static elements, and during execution for the dynamic constructs. Once such tools are in place, it will be possible to engage end-users into the application development process without a loss of control. This scenario is compatible with both the desire for better engineered software, a more productive less error-prone development methodology, and the changes that are occurring in the way organizations and work are managed [15].

5 - Security Issues.

Constructing security for an information architecture such as the one described here is a multidimensional problem. The appropriate security level for individual pieces of information is not stable over time and is frequently context dependent. As Ware has emphasized

[16], that security technology will always fall short, and that the greatest risk is unauthorized use by authorized users. Confidentiality and privacy must be considered to be at ongoing risk even when data systems themselves are otherwise secure. Complicating matters further, the very techniques that one hopes to use to improve patient care, namely aggregating data and by drawing inferences from it in order to gain diagnostic and therapeutic insights, are considered threats in other settings.

The architecture is modular and layered rather than monolithic (Figure 1). This provides the basis for systems in which the components are mutually distrustful [17]. Separating applications from data makes it possible to model work processes and construct data flows that clearly define boundaries of trust, for example, no one ordering supplies to restock the warehouse should ever be connected to any data source containing patient specific information. Each layer has specific security related tasks to perform.

The security behavior of the data layer must be adaptive, internally controlled and self-protective. It must decide whether or not to release information and it must control the permanent filing of new entries. It should also generate audit trails of database access (whether it stores then or not). The decision to release or accept information may be total or partial and it may be independent or mediated via trusted interactions with other system components. Loosely Structure Documents provide a mechanism to encapsulate security related information within them that can drive this activity. The approach is to tag those areas of content that have special security implications, such as the identities of the author and patient, the circumstances of document creation and any areas of the content that have a higher or lower level that the document as a whole. The default behavior of the persistent data store would be to only release information to the creator of the information, the individual to whom the information referred, or to properly cleared system administrators - unless there were additional restrictions encapsulated within a particular document that blocked this process.

The meta-data layer contains the information that defines the formatting and content architecture of the documents stored in the data layer. This layer also contains application specific wrappers. Wrappers are temporary encapsulations of process-related data in non-document form. This is an ideal place to assemble the information needed to drive access-control in a production environment as well as to maintain working copies of work in process and the knowledge bases needed to support production applications. In part, access can be controlled by creating application specific wrappers that only maintain working copies of a limited amount of data. Applications that only have access to these abstracts, but not the data layer, cannot violate it. The volume of information will be too great and the number of users too large to create specific authorization matrices. Most access control to the data layer will have to be handled on the basis of role information. Some roles can be defined statically, others are defined by prior events. In this model, role information would be supplied to an access controller by a wrapper (Figure 1). This wrapper might (among other things) construct association tables based on:

1) role assigning events
   - Dr. Smith is credentialed as a member of the active staff for 2 years and has privileges in general surgery.
   - Dr. Garcia is credentialed as a member of the active staff for 2 years and has privileges in pulmonary medicine.
   - Nurse Adams has a valid license and is hired by the facility.

2) administrative events
   - Nurse Adams is assigned to the Surgical Floor.
   - Mrs. Jones is admitted to the Surgical Floor by Dr. Smith
   - Mr. Jackson is admitted to the Medical Floor by Dr. Garcia
   - Dr. Smith requests a pulmonary consultation

The access control function would infer that:
   - Dr. Smith and Dr. Garcia can read all records and create entries relating to Mrs. Smith.
   - Nurse Adams can read and create entries relating to the current admission of Mrs. Smith as well as read her history for the past year.
   - Mr. Jackson has a cardiac arrest. Dr. Smith and Nurse Adams respond to the emergency. They assert that it is an emergency and have full access to the records for the duration of the event. They both have subsequent read-access to any entries that they made during the event.
   - If the facility was informed that Dr. Cohen, also on the staff, had joined Dr. Smith in practice, Dr. Cohen would have access to Mrs. Jones' records.
   - After Mrs. Jones is discharged Nurse Adams and Dr. Garcia no longer have access to Mrs. Jones' records.

The relationship between any access controller and the data-layer is clearly a trusted one.

Another wrapper might keep a working copy of the last week's vital signs and lab results on hospitalized patients for rapid retrieval and manipulation and an index of the data available on those patients that could be retrieved from the data layer. The difficult trade off is to decide how much information to disclose - too little and patient care be compromised, too much and the potential for an inferential attack is increased.

At the function layer and the application layer, the implementation of security will depend on the approach taken to creating the access controllers. While existing techniques may be applicable there are some challenging clinical requirements at the application layer to consider. One such challenge is presented by the need for virtual sessions. In a fast-paced chaotic environment like a trauma center, a physician may need to start work on several patients simultaneously and have the ability to continue the work on any patient from any available workstation. This calls for some type of virtual session manager that keeps all database connections alive, saves the state of all visual displays and can restore the operational state of the program on any terminal when requested by the initiator of the event or by another authorized individual.

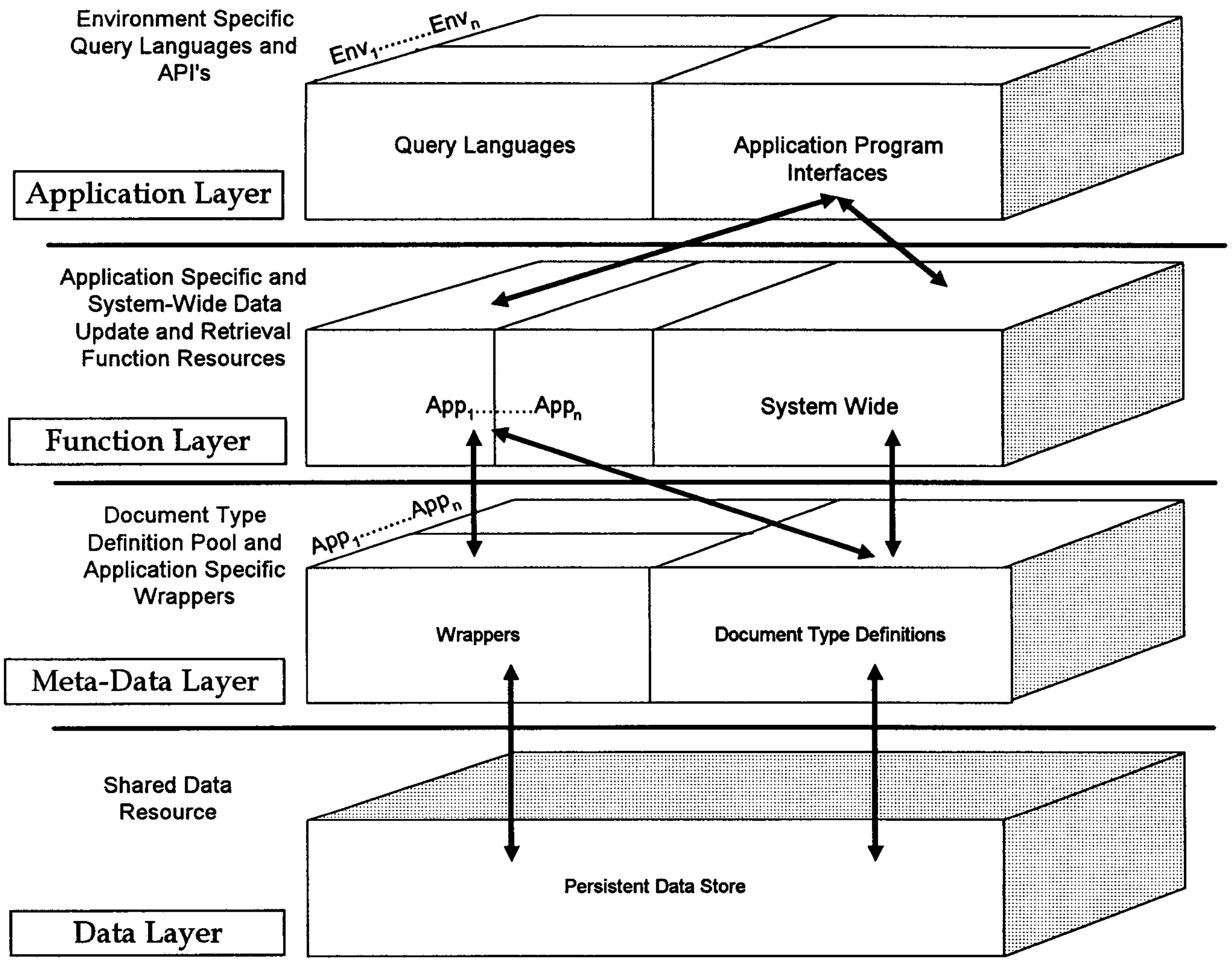
Access control has different meanings depending on context and time. As described in the scenario above, in an emergency context a wider range of individuals are allowed access. The value of some data is time limited. For example, the number ounces of liquid that Mrs. Jones consumed on her first day in the hospital day typically has a low security during the hospital stay and the level decreases steadily after discharge. Although the data must be retained for a legally prescribed length of time, the likelihood that this data will ever be retrieved is extremely low.

Several other security questions must be addressed that have little to do with the information architecture presented here but remain as open issues in the healthcare domain. The first relates to authorized copying of data. Various groups and the federal government would like to have access to various portions of the medical record to support a variety of research and planning activities. Should copies of this information be released or should these organizations be required to use it under secure conditions? If copies are released how should they be tracked? Can the systems controlling the data layer automatically apply transformations to the data as it is released to prevent the use of aggregation techniques to reestablish the identify of individuals? Is it possible to produce specially encrypted copies with built-in expiration dates on the decryption keys? Can data be released in an active form so that it can detect if it has been removed from a controlled environment and "self destruct"? Can data be released while maintaining control over the retrieval functions by allowing users a remote sites to "borrow" functions via remote procedure calls?

6 - Conclusion

The model presented in this paper suggests an approach to the development of medical records databases that focuses on creating tools: 1) to establish and maintain a persistent store of data that is external to all applications, 2) to allow those involved in medical events to accurately and efficiently document what has occurred, 3) to allow individuals and processes access to the accumulated information, and 4) to address at a structural level the need to insure that the database is permanent and secure.

Some of the requirements described here raise fundamental research issues that will need further study. Others, especially the reliance on raw text searching (even if assisted by content delimiting tags) and the assumption that security and open access are not contradictory are frequently perceived to present overwhelming obstacles. Many of these apparent obstacles are being overcome in the research lab, some have already been implemented and others are in search of new paradigms in system security.


Environment Specific Query Languages and API's
Env1.........Envn
Query Languages
Application Program Interfaces
Application Layer
Application Specific and System-Wide Data Update and Retrieval Function Resources
App1.........Appn
System Wide
Function Layer
Document Type Definition Pool and Application Specific Wrappers
App1.........Appn
Wrappers
Document Type Definitions
Meta-Data Layer
Shared Data Resource
Persistent Data Store
Data Layer

Modular and Layered Structure of the EMR
Illustrating Data Flows to a Specific Application
Figure 1.

References

[1] Norman D: "Things That Make Us Smart: Defending human attributes in the age of the machine," Addison-Wesley, 1993.

[2] Prosser RL: "Alteration of Medical Records Submitted for Medicolegal Review" JAMA Vol. 257, No. 19, May 20, 1992, pp. 2630-2631.

[3] Krishnamurthy R, Litwin W, Kent W: "Language Features for Interoperability of Databases with Schematic Discrepancies," Proc. of ACM-SIGMOD Conf. SIGMOD Record, 20, 2, (June 1991), 40-49.

[4] Litwin W, Ketabchi M, Krishnamurthy R: "First Order Normal Form for Relational Databases and Multidatabases," SIGMOD Record, 20, 4, (December 1991), 74-76.

[5] Essin DJ: Unpublished material. 1987.

[6] Rowland HS: Nursing forms manual. Aspen Systems Corp., Rockville, Md. 1985

[7] Essin DJ: Intelligent Processing of Loosely Structured Documents as a Strategy for Organizing Electronic Health Care Records. Methods of Information in Medicine, in press.

[8] Lincoln TL, Essin DJ and Ware WH: The Electronic Medical Record: A Challenge for Computer Science to Develop Clinically and Socially Relevant Computer Systems to

Coordinate Information for Patient Care and Analysis. The Information Society, Vol. 9, No. 2 (Apr-Jun 1993), 157-188.

[9] Soloway E: Reading and Writing in the 21st Century, Commun ACM Vol. 36, No. 3, May 1993, pp. 23-30.

[10] Goldfarb C: "The SGML Handbook," Oxford University Press, 1990.

[11] Dayal U, Manola F, et al: "Simplifying Complex Objects: The Probe Approach to Modeling and Querying Them," Proceedings of German Database Conference, Burg Technik and Wissenschaft-87. reprinted in Zdonik, S. and Meyer, D. Readings in Object-Oriented Database Systems, Morgan Kaufman Publishers, San Mateo, CA 1990.

[12] Newcomb S, Kipp N, and Newcomb V: "The 'HyTime' Hypermedia/Time-based Document Structuring Language," Commun. ACM, 24, 11, (November 1991), 67-83.

[13] Carriero N and Gelernter D: Linda in Context. Commun. ACM, 32, 4, (April 1989), 444-459.

[14] Nii HP, Feigenbaum EA, Anton JJ and Rockmore AJ: Signal-to-Symbol Transformation: HASP/SIAP Case Study. in: Englemore R and Morgan T. Blackboard Systems. Addison Wesley 1988.

[15] Tapscott D and Caston A: "Paradigm Shift: The New Promise of Information Technology" McGraw-Hill, 1993.

[16] Ware WH: personal communication.

[17] Computers at Risk: Safe Computing in the Information Age. National Research Council. National Academy Press, 1991.