Components of the CPR: An Overview

by Tom Martin and Sandra Fuller, MA, RRA

To better participate in the implementation of the CPR, it's important that HIM professionals understand the nuts and bolts of a system. Here's an overview from the ground up.

Across the nation, the implementation of computer-based patient record (CPR) progresses -- slowly but steadily. Participants in the 1998 HIMSS Leadership Survey indicated that this year, for more than 50 percent of healthcare facilities, providing clinical applications support is a top focus.

The benefits of a CPR are well known -- such as an integrated view of patient data, access to knowledge resources, proactive influence on physician order entry and clinician data capture, integrated communications support, and clinical decision support.[1]

But reaching the computerized record is not an easy journey. It needs the dedicated attention of both HIM and information technology (IT) professionals to supply information services to the rest of the healthcare enterprise. Such a partnership requires an understanding of HIM and IT domains, joint planning, and a higher level of collaboration. To become better CPR "partners," HIM professionals need to understand the principles underlying today's information technology.

At the Root: Enabling Technologies

Network Architecture

The most fundamental parts of a computerized patient record are the hardware and the network. Hardware, software, and networks are applied together through network architecture to deliver functionality and data. Components of the network include transmission media, transmission protocols, hubs, routers, switches, bridges, and modems.

The network architecture is critical to the design of a computerized patient record system. It determines access to the network, connectivity, and the capacity to support the number of users and the amount of data (called "throughput"). Network architecture also specifies the standardization of devices and software and the availability of the network supported through sufficient redundancy to minimize downtime. ("Redundancy" means that adequate resources are available throughout the network so that there is no single point of failure.)

Hardware Components

Hardware may include servers, clients, wireless systems, legacy systems, and printers.

Servers and disks must be configured to support the number of users and the amount of data in a CPR system. Servers must be sufficiently redundant to ensure availability of patient data when needed, and there must be a sufficient number and type of clients or devices to support end-user functions.

Connectivity Software

It takes more than hardware and cabling to create a computer network. Specialized connectivity software allows devices to function on a network. Terminal emulation allows a client to look like a specific type of terminal. Socket-to-socket protocol software allows a process on the client to communicate with a process on the server across the network. Finally, file transfer protocol is used as a tool for transferring files between servers on a batch basis. In evaluating network architecture, address the following issues:

Availability: Is the system up when it's needed?

Access: Is the system available where it's needed?

Reliability: Can the results be trusted?

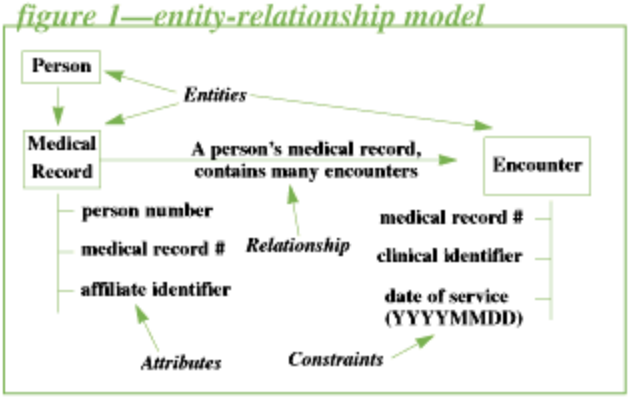
Performance: Is the response time adequate?

Security: Is patient confidentiality protected?

Supportability: Can you afford the resources required for ongoing maintenance?

The Next Level: Data Architecture

figure 1—entity-relationship model



Beyond the physical composition of a network, data makes the network necessary and adds value to the applications. In an application as data rich and dependent as a CPR, the data architecture is critical. Data architecture is a description of the data needed to operate an enterprise and the technical components used to maintain and deliver that data.

Data Modeling

Data modeling is a discipline within the development of information systems that focuses on identifying the static components of a system. HIM professionals need to understand how data is related and may be asked to assist in developing or evaluating a system using a data model. Knowing how to model data will assist in ensuring that the system will meet the needs of the institution. There are three common types of data models.

The entity-relationship data model (see Figure 1) defines the data components of a system in the process of systems analysis. It is generally accepted as a good way to describe data in a conceptual manner. The entity-relationship data model has four parts: the entity (something about which we store data); the relationship (rules about how entities relate to each other); attributes (the information that describes an entity); and the domains or constraints (valid values for an attribute).

A second type of data model, the relational model, is the most widely used. It is a mathematically rigorous model that describes data within a system using matrices as the fundamental construct. Advantages of the relational model are:

- Data is in one place
- Data is integrated and shared
- Controlled redundancy
- Complex relationships can be represented
- Standards and security can be reinforced

The model consists of tables made up of rows and columns (see Figure 2). The rows describe an occurrence of related information. The columns are attributes that describe a row. A primary key is a special attribute that uniquely identifies a row on a table.

The third type of data model is an object model (see Figure 3). The object model was developed to address data in a more "real-world" approach than the limits of the relational model; it represents the static and structural data aspects of a system. The components of an object model are the object (the discrete item depicting both data structure and behavior); class (describes objects with the same attributes and behavior); and instance or specific occurrence of a class and inheritance (sharing of attributes and operations among classes, based on a hierarchical relationship). As the newest type of data model, object models are less proven, but they are gaining popularity because they focus both on the data and behavioral attributes of an object. This results in more easily shared and maintained code.

Advantages of the object model are:

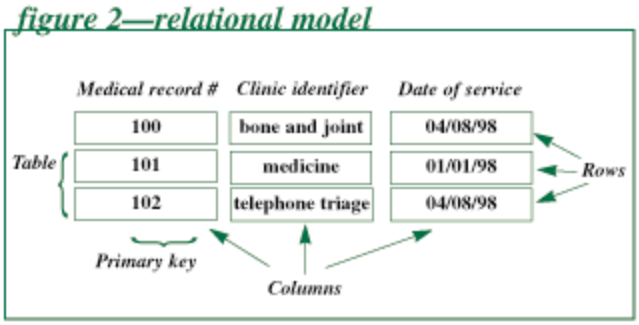
- Databases better integrated with programming language
- Complex data readily processed
- Automatic persistent object IDs; two different objects are never confused

Data Dictionary

A data dictionary is used to capture the results of data modeling and can provide graphical representation of the data model. It is a user-accessible catalog of the database and the application metadata -- information about the database, such as field length, data edits, and definition. It can also generate data definition language for the data management tool employed and can provide online documentation of the data model. A data dictionary includes:

- descriptions of entities, classes, or tables
- descriptions of attributes, including length, data type, definition, valid value, table(s) in which the attribute occurs, data source for each attribute, data restrictions, access restrictions
- processing rules
- relationships between entities or classes
- keys

The data dictionary defines the uniqueness of each data element within a system, so development of a data dictionary is a critical step. A data dictionary also supports the transmission of data through interfaces or data sharing and assists in the analysis of data by providing a clear definition of the data being analyzed. A data dictionary can be active (contents are automatically updated after changes are made) or passive (manually updated).



Data Structures

The data depicted in the data model and defined in the dictionary must actually exist within a data structure in the computer. The most common data structure is the file. A file is a physical structure where data is stored as defined by an application. A file contains records that are an occurrence of the related data. Within each record are fields that are attributes of the related data. File structures are prone to redundant data and are inflexible in representing the relationships of data, which may result in data maintenance problems. Although files are easy and less costly to create, they can be more expensive to maintain and usually will not be flexible enough to support the relationships needed to describe data in the real world.

Database management systems were developed in response to the limitations of files. Database management systems are the software and data structures used to support a database. The application that uses the data is actually separate from the data. Types of database management systems include hierarchical, network, relational, and object. Database management systems may be standard or proprietary and are the most frequently used way to support data.

Object databases are the newest forms of databases. These support the object model and object-oriented languages. They are not widespread, as the systems they support are not heavily used yet, but their use is increasing. The computerized patient record will place demands on data architecture that can be evaluated by questions like these:

- Does it support the types of data used (patient data, images, etc)?
- Does it support the types of access required (one patient at a time, multiple patients)?
- Does it support the performance requirements?

- Does it easily allow changes to the data model?

Application and Interface Strategies

Without applications, there would be little use for hardware, networks, or databases. Applications are software systems written to perform a particular function in support of some process. They exist to help an end user meet a business need. In healthcare, that end user may be a clinician, administrator, technician, or researcher whose need may be providing care, planning new services, administering a therapy, or analyzing outcomes. Each of these processes requires a unique software application.

Choosing the best application software is a critical element of constructing a computerized patient record. Key to that selection is the decision to buy an existing vendor package or build a customized application. Some factors to consider in making the "buy versus build" decision include:

- Are packages that meet the needs available?
- Does the institution have a relationship with a vendor that has an applicable package?
- Do the requirements represent something unique?
- Do the requirements represent something that will put your institution at a competitive advantage?
- Does the IT staff have a track record of successful custom development efforts?
- How quickly do the functions need to be available?

The answers to questions like these will facilitate your decision to buy or build your application.

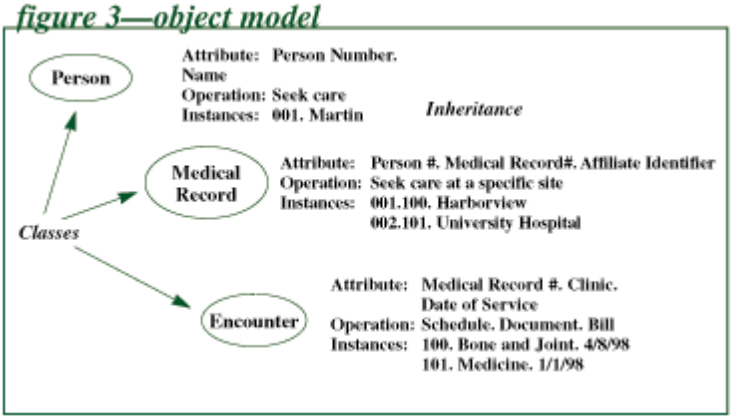
Types of Applications

Online Transaction Teleprocessing (OLTP)
OLTP applications are typically considered online applications. A transaction is usually a business event that requires a set of data be captured, edited, and stored in a database. OLTP applications can also be read-only applications. Typical OLTP applications include registration, ADT, or appointment creation. Characteristics of an OLTP application should be understood and considered prior to acquisition or development:

- stores detailed data
- accurate as to the last transaction
- disciplined, highly structured, and planned transactions

User Interface
Two types of user interface are commonly used in OLTP applications. Character-based applications generally use the keyboard and are considered better for data entry-intense applications. Character-based applications usually have faster response time because of less network traffic. Graphical user interfaces (GUIs) are where the market is headed, with Windows and Web-based products leading the way. Graphical interfaces are designed to be intuitive and are easier for infrequent users to understand. They also allow pictures, graphics, or video to be used.

figure 3—object model



Navigation
Users can navigate through a screen or an application in a variety of ways. Function keys and menus are the most common means of navigating in character-based applications. Pull-down menus, buttons, radio dials, and slide bars are navigational tools created specifically for a mouse. The key to navigation is consistency; the user should not have to move between keyboard and mouse frequently or wonder how to move within an application. Navigation should be tested with end users for intuitiveness and speed.

Editing
Editing -- checking the data entered for completeness and accuracy within established ranges-can be done a character at a time or a screen at a time. "Character at a time" editing is most typical in PC applications and mini-computer-based applications. As data is entered, it is reviewed by the computer and edited based on the application code. Editing "a screen at a time" is more typical with mainframe applications and Web forms. All data is entered without edits, then sent to a mainframe for editing upon completion of the screen. The screen returns showing any errors.

Update strategy
An update transaction -- moving information from the application to the database -- can be done a screen at a time or at the end of an entire business transaction. "Screen at a time" updating implies that once you have left a screen, all the data entered on that screen has been committed to the database. This kind of updating is more intuitive for users. "Business transaction at a time" updating implies that if a business transaction takes more than one screen, and there is a failure midway, the data is updated in the database for that transaction. Business transaction updating may be useful where partial transactions are not acceptable.

Concurrency
Concurrency means locking records while waiting for updates. There are two strategies for locking records. In the first, records are read and locked while the user is updating them, so no one else can update them. The record stays locked until the update is completed. In the second, records are read but not locked; before updating them, the application rereads the records and checks to see if they have changed; if not, it updates them, otherwise it warns the user.

Online Analytical Processing (OLAP)
OLAP applications allow analysis of sets of data on an ad hoc basis. Typical healthcare OLAP processing includes retrospective decision support, research, quality improvement data analysis, and outcomes analysis. OLAP applications generally run on data generated by OLTP applications. As with OLTP applications, characteristics of OLAP applications should be understood to match system features with user requirements.

Normalized versus denormalized data
Normalized data is preferred if the data is volatile, because every normalized relation has a single theme. Any relation having two or more themes should be broken up into two or more relations, the essence of normalization. Changes need to be made in the fewest number of places. Denormalized data is easier for writing reports or queries because the database already knows the pertinent relations. For example, patient to discharge date is a relation; patient to principal diagnosis code is a relation; and diagnosis code to description is a relation. To write an ad hoc report, users need to know about each of these relations and be able to join them in a relational database.

Summarized data versus detailed data
For ad hoc reporting, most data is used in summarized form. Summarized data takes less disk storage and less time to analyze. However, sometimes data loses some of its usefulness if there is not sufficient detail.

Retention
Data may need to be stored for a long time for research purposes. An OLAP database may also be part of the computer-based patient record and may be subject to state and federal retention laws.

Physical database size
Long retention and detailed data contribute to large database size, which can be hard to navigate and are hard to administer (e.g. recover, change, or add attributes).

Onward and Upward: Application Architectures

Ways to Share Data

The first element of application architecture is data sharing. At one end of the data-sharing continuum are standalone systems that are developed or implemented irrespective of other systems. Some data is redundantly entered and kept in multiple systems using this model. Standalone systems may be easier to develop, but they are costly to maintain and operate because they create silos of information that is not shared.

In the second data-sharing model, standalone systems are linked together through interface systems. While there are standards for sharing data between systems, like HL7, a common vocabulary is necessary for mapping data contents. The interface model allows best-of-breed applications to be purchased and data linked through the interface. However, developing and maintaining multiple interfaces between systems can be very costly, so specialized software called "middleware" creates an interface engine that directs the data from one system to another.

In the third, fully integrated data-sharing model, applications are written using a shared database. Data is entered once and is then available for all applications. Although this is the most economical model for construction of a database, frequently it requires compromise on functionality as one system tries to fit the needs of numerous and varied users.

Ways to Distribute Data

The second component of application architecture is data distribution -- where the data is physically located. A centralized database has all associated data in one location. This is the most typical database approach and the easiest to maintain and support. However, centralized data is the hardest to scale as the database grows. A centralized data approach also creates a single point of failure for a system.

A distributed database spreads data across multiple servers by logical groupings, determined by either geography or by use. Data distribution allows data to be managed more efficiently based on its use and reduces the risk for one point of failure. A distributed data model requires more sophisticated software but allows for smaller investments of hardware as the database grows. A replicated database copies data and utilizes special software to recognize data that has been changed, added, or deleted and stores that data in a separate server.

Application Functions

Applications typically have three types of technical functions: presentation, business logic, and the database. Presentation controls how the application looks and feels. Business logic contains all the business rules, edits, and logic for processing input and data. The database controls how the application organizes, stores, and retrieves data.

Just as database models vary based on their degree of centralization, the functionality of an application may vary in its distribution. Host-based applications contain the presentation, business logic, and database all on the same computer. All three functions may even be within the same program. This may make the application hard to maintain, because all three functions are more tightly coupled to each other. This model does not respond as quickly to technology changes.

A client-server function distribution divides the presentation, business logic, and database over two or more devices. Typically, in this model there is a request from a client for services from one of the other functions. Message-based applications involve the most decentralized functionality because messages for functionality are sent to a broker and then to the appropriate server to be completed.

Although applications are the most familiar part of information systems for most end users, they are complex and varied. The use, amount, volatility, and presentation of the data all play a critical role in determining what application design will best fit the business function.

Success Strategies for Implementation

Regardless of how a new application is acquired, some key success factors apply to both purchased or developed applications.

Executive Sponsorship

A sponsor from the executive office is a key element in the successful implementation of any system. The sponsor is critical in policy decisions, resource allocations, institutional commitment, and operational accountability.

User Commitment
A group of dedicated users who are willing to actively participate in the acquisition, evaluation, testing and implementation phases of your project are critical. At the application level, users are the only ones equipped to perform some of the required activities.

Well-defined User Requirements
Whether you build or buy, a complete understanding of the application requirements must be defined at the outset of the project. User requirements may be the biggest factor in your decision.

Stable User Requirements
Any system implementation is going to take time. If the system requirements continue to change during development or acquisition, you may never deliver functionality to the user. Managing the dynamics of project time against expectations of functionality is a critical element of system project management. Building in project phases is a good way to address these needs.

Issue Resolution
Any implementation of application software will result in problems and issues. Before those problems arise, a standard issue tracking system and resolution process should be developed and agreed to by user representatives, the executive sponsor, and the implementation team.

Established Technical Standards
A set of technical standards to simplify both user functionality and the development team's approach can streamline any implementation. Technical standards may be the assignment of F1 as the help key, consistent highlighting for a screen alert, or consistent location of header information across systems and screens.

Adequate Staffing
Beyond staff whose primary responsibility is development of installation of software, technical user support can entail testing, developing policy, training, and troubleshooting. Don't forget technical support staff to deal with the hardware, network, communication, and database.

Project Management Discipline
The final key success strategy is the use of project management tools to orchestrate the various pieces of the implementation. These include schedules, work plans, resource requirements, status reporting, and issue tracking.

Note

1. Dick, Richard S., Elaine B. Steen, and Don E. Detmer. *The Computer-Based Patient Record: An Essential Technology for Health Care.* Washington, DC: National Academy Press, 1997.

Reference

Kroenke, David M. *Database Processing: Fundamentals, Design, and Implementation*. Upper Saddle River, NJ: Prentice-Hall, 1998.

Tom Martin *is the director/CIO of the University of Washington Medical Centers Information Systems.* ***Sandra Fuller*** *is AHIMA's vice president of practice leadership.*

Article citation:
Martin, Tom, and Sandra Fuller. "Components of the CPR: an Overview." *Journal of AHIMA* 69, no.9 (1998): 58-64.

Driving the Power of Knowledge