

COMPUTER SOCIETY
PRESS REPRINT

BUILDING BLOCKS FOR DISTRIBUTED INFORMATION SYSTEMS IN HOSPITALS

Thomas E. Rutt

Reprinted from PROCEEDINGS OF SCAMC — SYMPOSIUM ON
COMPUTER APPLICATIONS IN MEDICAL CARE,
Washington, DC, November 1-4, 1987



The Computer Society of the IEEE
1730 Massachusetts Avenue NW
Washington, DC 20036-1903

Washington • Los Alamitos • Brussels



THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

COMPUTER
SOCIETY OF
IEEE

Building Blocks for Distributed Information Systems in Hospitals

Thomas E. Rutt

AT&T Bell Laboratories
Holmdel, New Jersey 07733

ABSTRACT

In order to provide a consistent view of work functions which can be supported by successive releases of hospital information systems, a hospital business model was developed. The business model describes potential system functions along with their data access needs. This model was used to determine an optimal set of mutually exclusive logical databases. These logical databases are seeds around which design of distributed systems should be based.

The logical databases derived by this work can be used as a starting point for defining application protocol standards for hospital system interface transactions.

1. INTRODUCTION AND SUMMARY

1.1 Background

AT&T is interested in supporting the use of distributed information systems in hospitals. A distributed system relies on a host-to-host communication network to carry system interface transactions.

We carried out an extensive study of the information needs of hospitals, to help us partition support of a hospital's work functions between a canonical set of system building blocks. We called these building blocks system modules.

We found utility in finding natural groupings of data which simplified design of distributed system interface transactions. This paper reports outcomes of our work.

1.2 Overview of Work

The following outline describes work efforts which led to this paper.

- A non-redundant set of functions was collected into a functional decomposition which is described in Section 3 of this paper. We created a functional decomposition of the hospital business using three levels. The first two levels of

decomposition are shown in Fig. 4.

- We determined the data entity access requirements for each function in the business model, using logical data entities. Data entity access requirements were mapped to work functions at the third level of the functional decomposition.

The resulting catalog of hospital functions is called a Business Model, and shows how each function uses hospital data. It can be used to help resolve system integration problems, since it provides a common language to discuss the functionality of existing or proposed systems.

- We conducted a data-entity affinity analysis, which had as output a set of logical databases containing entities used together often. The analysis is described in Section 4.

The results of the data-entity affinity analysis based on the business model is presented in Fig. 6.

2. TERMINOLOGY

This section defines the terminology used in this paper, and Figs. 1 and 2 facilitate understanding of these definitions. Fig. 1 depicts each term as a bubble. The arrows between bubbles specify relationships among the terms. A double headed arrow specifies a one-to-many relationship between the bubble pointed from and the bubble pointed to. For example, a System Module (MODULE) is "composed of" many FUNCTIONS. The recursive relationship specifying that functions are composed of other functions is indicated by the arrow pointing both from and to the function bubble.

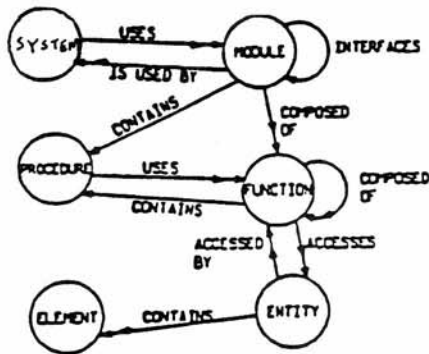


FIGURE 1

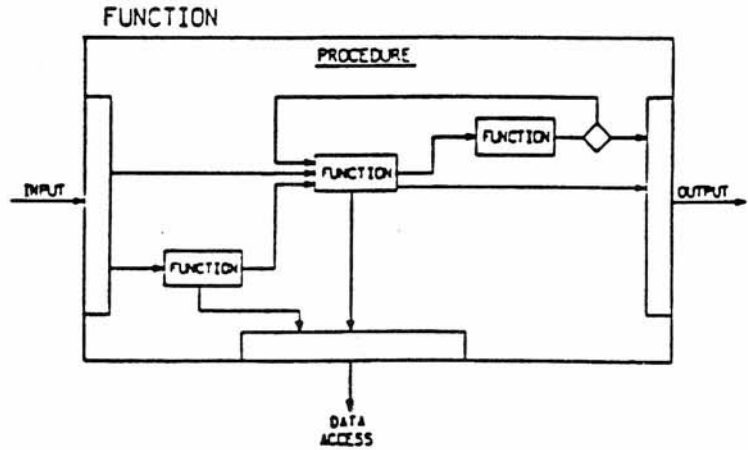


FIGURE 2

2.1 System

A system is a packaged unit of documentation, software and hardware which is sold as a unit, with the possibility of options. Systems may depend on the existence of other systems which they interface with.

2.2 System Module

A system module is a high level grouping of functions that may be part of one or more systems. System modules allow the definition of systems with overlapping functionality, while keeping the overlaps well understood and under control. Ideally a system function should only be part of one system module; replication of a function in two systems should be equivalent to a system module being part of both systems. By strictly defining system modules in this manner, costly duplication of development effort is minimized.

A system module is described as a group of functions, which can be viewed as a feature superset (i.e., an umbrella model). Not all of the functions in a module description will be supported in a product release of a system. The user communities may place different feature requirements on each implementation.

A system module may be implemented on different processor types for each system it is part of. For example, a system module could have either an IBM CICS

implementation or a UNIX System V implementation in a given hospital. To insure maximum interconnection, the module interface definitions should be the same in either case.

2.3 Interface

The need in one module for data which resides in another module, requires that an interface be developed between the two modules. A complete interface specification includes the definition of each data element which flows in each interface transaction. Code has to be written by the development teams on both sides of the interface for it to be implemented. Interfaces may use the services of a data-communication network for physical transport of the data.

A successful interface implementation requires ongoing coordination of the definitions of the data elements which are to be passed between successive releases of two interfaced modules. Synchronization of interfaces between multiple system releases is a difficult task requiring careful planning and robust interface design techniques [1]. ISO DIS 9072 defines a Remote Operations Standard, which is becoming widely used to implement extensible interface transaction protocols.

2.4 Function

A function is an action which has specified inputs and generates specified outputs.

Functions can be carried out by people with or without the assistance of system features, or by a system features alone. A function can be described by decomposing it into lower level functions. The number of decomposition levels required depends on the intended purpose of the functional description.

A system feature is a mechanized system function, which has inputs and outputs consisting of user interface transactions, module interface transactions and internal database transactions.

2.4.1 Procedure Procedures are used to realize functions. A procedure is a group of component functions and associated sequence and control constructs which are used to perform the required operations to generate a higher-level function's outputs (see Fig. 2).

Procedures are often specified in structured English or flowchart form. Examples of control concepts are interrupts, conditional execution sequences, and repetition.

2.5 Data Entity

An entity is a type of thing, physical object or abstraction about which information is stored. Examples are "patient", "doctor", "test order", and "specimen event".

From a modeling standpoint entities can be thought of as normalized data relations. They are data groups which are non-overlapping.

2.6 Relationship

A relationship is a way of specifying linkages among data entities. Relationships have cardinality of one-to-one, one-to-many, or many-to-many. Data entity relationships are defined between pairs of data entities in a logical data model, which can be used to design physical databases [2]. For example, linked lists are often used to implement one-to-many relationships.

In constructing a data model, a designer should try to resolve many-to-many relationships. When many-to-many relationships are proposed by system designers, intersection data entities should be defined. A proposed many-to-many

relationship can be decomposed into two one-to-many relationships involving an intersection data entity.

A particular example involves "test dir" and "specimen event". Some specimens can be used for multiple tests, while there are some tests which need multiple specimens. This potential many-to-many relationship between "test dir" and "specimen event" is resolved by introducing a new entity "test event", which has a many-to-one relationship with each of the data entities "specimen event" and "test dir".

2.7 Data Element

A data element is an item that is a property of a data entity. Examples are "patient name", "patient address", "admitting diagnoses". A synonym for element is attribute. Data elements are represented by fields in database records.

Data element redundancy is eliminated in a logical data model by allowing each data element type to be an attribute of only one data entity type. For example, the patient's address is in only one entity, namely "patient". The fact that a patient's address may eventually reside in record fields of several system files or databases is of concern, since the records should be kept in synch when an address is changed.

A non-redundant model of data elements is required for interface specification and validation purposes. A data dictionary should be used to store the field attribute characteristics of each data element used in system application programs.

3. FUNCTIONAL DECOMPOSITION/BUSINESS MODEL

A business model is the core of many modern system development methodologies [3,4,5]. Fig. 3 shows how the business model fits into a complete dictionary driven development methodology. It guides the design decisions pertaining to the grouping of functions into systems and programs, and the grouping of data elements into database records and screens.

A logical business model helps users and system designers get a clear and common picture of the current environment, giving

- 5 Provide Hospital Resources
 - 51 Manage hospital services
 - 52 Manage facilities
 - 53 Develop HIS environment
 - 54 Manage medical records
 - 55 Manage information systems
- 6 Provide Professional Environment
 - 61 Provide educational environment
 - 62 Provide research environment
 - 63 Maintain Medical Knowledge Base
- 7 Manage Hospital Business
 - 71 Manage Patient Accounts
 - 72 Manage Accounts Payable
 - 73 Manage Fixed Assets
 - 74 Manage General Ledger
 - 75 Manage workforce
 - 76 Analyze Third Party Reimbursements
 - 77 Plan Hospital Business

333 secure blood availability [blood work event, test event, blood work dir, blood schedule, blood dir, outside agencies dir, external vendor dir, internal supply order, blood inventory event,]

334 match blood [specimen, test event, blood work event, blood dir, blood work dir, blood schedule, blood inventory event,]

335 provide blood [blood event, blood work event, functional area dir, blood inventory event, blood dir, charge dir, blood workstation, blood schedule, charge event,]

336 maintain blood knowledge base [blood dir, blood area dir, blood job dir, blood work dir, blood workstation, functional area, job dir]

This type of representation is known as a Business Model. These data accesses convey intrinsic data flows needed and produced by each function. By treating data needs in a uniform representation for all the functions of an organization, we get a consistent view of its operations.

In the business model data needs are captured without concern of how accesses will be implemented. In this view there is no difference between an interface transaction and a database access. If data is not available in a local database, an external interface transaction may be used to transfer data between systems, just as a phone call is used in a manual environment.

In Fig. 5, data entity access requirements are shown in a list enclosed between the square brackets. Data entity names are used rather than data elements to keep the model at a manageable level of detail.

The full list of data entities in our hospital business model is presented in Figure 6, grouped into logical databases resulting from a computer analysis described in the next section. In the entity names the common words "dir" and "event" are used and need explanation. A "dir" is a static directory entity, which has knowledge and representation data about the subject referred to in the "dir" entity name. An "event" is a dynamic instantiation of a piece of data recorded at a given epoch in time about some subject matter. The "event" data entities are created by application software using information represented by corresponding "dir" entities. In a non-computerized operation the "dir" entities are in books, and the "events" are often

The functional decomposition in our model has three levels (only the first two levels shown in Fig. 4 due to space). Although it is possible to decompose the lowest level functions further the three level decomposition suffices for the intended purpose.

The same function may be implemented as a feature on more than one system, but designers should be made aware of any feature duplication to coordinate development. A business model helps to provide this awareness.

3.4 Linking Functions and Data Entities

The data entities required by the work functions are shown at the lowest level of the functional decomposition in the example in Fig. 5. The example of Manage Blood Products was selected because it is representative of the level of detail in our model, and space doesn't allow the entire model to be shown.

Figure 5 - Example Blood Product Business Model Section

33 Manage blood products

331 Manage blood bank work [blood order, blood work event, blood work dir, blood job dir, blood schedule, blood workstation, doctor dir, doctor schedule, employee dir, employee schedule, job dir, building location dir, timesheet,]

332 obtain blood sample [blood sample order, blood work event, blood schedule,]

slips of paper in the patient chart.

4. LOGICAL DATABASE DETERMINATION

The value of this business model analysis is that it is independent of the definition of system modules. All database accesses are considered to be on a large pool of data entities. These entities have to be subdivided among the product modules which they will reside in. Then after the functions are assigned to modules, the function entity accesses have to be subdivided into internal database transactions and module interface transactions.

The functions which each product module will support, as well as which module will own each entity (ownership implies update control responsibility) are all decisions which are part of the system design process. The access requirements from the business model can be used to infer the module interfaces, once these entity ownership and function assignment issues are resolved.

Remote operations protocols are being developed [6,7] with the express purpose of transferring data entity records between hospital systems. There is a need to develop standard application protocols for interchange of data between open hospital information systems.

4.1 Methodology for Determining Logical Databases

We used the business model as input to conduct a computerized data-entity affinity analysis [5]. Two entities have a higher affinity (affinity is ≥ 0 and ≤ 1) if they are used together more often. The affinity of entity E1 to entity E2 is defined as the number of functions using both entities E1 and E2, divided by the number of functions using entity E1.

Elements $D(i,j)$ of a symmetric distance matrix were calculated by subtracting from one, the average of the affinity of E_i to E_j and the affinity of E_j to E_i . This distance matrix was used as input to a hierarchic cluster analysis using the "S" system statistical utilities [8] within UNIX.

4.2 Logical Database Results

The output of the hierarchic cluster analysis is shown in Fig. 6. The computer generated the groups of entities, and the names were selected afterwards to convey the nature of the grouping.

These logical databases serve as "seeds" around which we can define hospital system modules. A canonical set of interface transactions can be defined for transferring data between these logical database building blocks.

Figure 6 - Hospital Logical Databases

ADT : admission request, patient, patient encounter, image dir,

BLOOD : blood dir, blood inventory event, blood area dir, blood job dir, blood schedule, blood work event, blood work dir, blood workstation,

CENSUS : bed dir, bed event, bed order,

CHARGES : charge event, charge dir,

CLINICAL DATA : medication event, surgical dir, surgical event, test event, therapy dir, therapy event, blood event, disease dir, diagnosis dir, diagnosis event, assessment dir, history dir, history event, image event, nurses notes event, doctors notes event, patient education, patient care plan, patient treatment plan, patient vital sign event, patient assessment event,

CLINICAL PROCEDURES : care plan dir, doctors notes dir, nurses notes dir,

DIETARY : diet area dir, diet job dir, diet schedule, diet work events, diet work dir, diet workstation, food dir, meal dir, patient meal schedule,

DISCHARGE : discharge summary event, medical record deficiency event,

FACILITIES : internal supply order, systems plan, external vendor dir, vendor product dir, capital equipment dir, computer operations procedure dir, maintenance area dir, maintenance job dir, maintenance schedule, maintenance work event, maintenance work dir, maintenance workstation,

FINANCE : patient bill, billing case, gaurantor, hospital dir, insurance company dir, patient account summary, patient financial information,

HOUSEKEEPING : housekeeping order, houskeeping area dir, housekeeping job dir, housekeeping schedule, housekeeping work event, housekeeping work dir, housekeeping workstation,

INSTITUTIONS : outside care facility dir, institution dir,

INSTRUCTION : student dir, course dir, course

schedule, curriculum dir, professor dir,

INVENTORY : inventory event, inventory area dir, inventory job dir, inventory schedule, inventory work event, inventory work dir, inventory workstation,

LABORATORY : specimen event, technician dir, test dir, lab work event, lab area dir, lab inventory event, lab schedule, lab work dir, lab workstation,

MATERIALS : shipping invoice, vendor account dir, vendor invoice, vendor remittance, purchase order,

MEDRECS : medical record area dir, medical record job dir, medical record schedule, medical record work event, medical record work dir, medical record workstation, medical record, medical record abstract, medical record location, medical record order,

ORDERS : social services order, test order, therapy order, blood order, diet order, hospital service order, image order,

PHARMACY : medication dir, medication order, pharmacy area dir, medication inventory event, pharmacy job dir, pharmacy schedule, pharmacy work event, pharmacy work dir, pharmacy workstation, patient medication schedule,

PLANNING : budget, diagnostic related groups dir, drg event, icd9 dir, payment, price master,

RADIOLOGY : radiology area dir, radiology inventory event, radiology job dir, radiology schedule, radiology work event, radiology work dir, radiology workstation, image dir,

SCHEDULING : surgical reservation, surgical area dir, surgical job dir, surgical schedule, surgical work event, surgical work dir, surgical workstation, test reservation, therapy area dir, therapy job dir, therapy schedule, therapy work event, therapy work dir, therapy workstation, clinic admission request, clinic area dir, clinic job dir, clinic schedule, clinic work event, clinic work dir, clinic workstation, ancillary reservation, bed reservation,

SERVICES : hospital service dir, hospital service area dir, hospital service job dir, hospital service schedule, hospital service work event, hospital service work dir, hospital service workstation,

SOCIAL_SERVICE : social service area dir, social service job dir, social service schedule, social service work event, social service work dir, social service workstation,

UNIT_MANAGEMENT : care unit area dir, care unit job dir, care unit schedule, care unit work dir, care unit workstation,

WORKFORCE1 : timesheet, doctor dir, doctor schedule, functional area dir, employee dir, employee schedule, job dir, building location dir,

WORKFORCE2 : outside agency dir, nurse dir, pharmacist dir,

(THE FOLLOWING DATA ENTITIES WERE LEFT UNGROUPED) : blood sample order, clinic inventory event, patient discharge order, ledger account dir, maintenance order, medical record abstract order, postal address dir, patient condition, patient critical information, surgical order, treatment plan, patient transfer order,

5. CONCLUSIONS AND RECOMMENDATIONS

This paper is an important step in the overall definition of a hospital application level interface standard. The logical databases discussed in this document provide seeds around which system modules can be built.

Since data that is used together often are in the same logical data base, system modules designed around logical data bases should minimize the necessity for system interface transactions.

In particular these logical databases should be helpful for specifying interface transactions between hospital information systems.

REFERENCES

1. Parnas D.; "On the Criteria to be Used in Decomposing Systems into Modules"; Comm. of ACM Vol. 15 No. 12, pg. 1053, December 1972
2. Martin, J.; "Computer Data-base Organization"; Prentice-Hall, 1977; ISBN 0-13-165423-3
3. Wasserman A. I. ; "Interactive Development Environments for Information Systems"; Proceedings of Tenth Annual Symposium on Computer Applications in Medical Care, Oct. 25-26 1986, pg. 316; IEEE Computer Society Press; ISBN 0-8186-0739-4
4. Iyer S. and Wilson A. ; "A Dictionary Driven Development Methodology"; IEEE CH2053-7/84/0000/0096 ; Proceedings Trends & Applications 1984, Making Database Work; IEEE Computer Society Press; ISBN 0-8186-0668-8
5. Martin J.; "Managing the Data-base Environment"; Prentice-Hall, 1983; ISBN 0-13-550582-8
6. Tolchin S. , et. al.; "A Distributed Hospital Information System", Johns Hopkins APL Technical Digest, Volume 3, No. 4, 1982
7. Tolchin S. and Barta W.; "Local Network and Distributed Processing Issues in The Johns Hopkins Hospital"; Journal of Medical Systems, Vol. 10 No.4, 1986
8. Becker R. A. and Chambers J. M. ; "S An Interactive Environment for Data Analysis and Graphics"; Wadsworth 1984; ISBN 0-534-03313-X