Design of a Knowledge Driven HIS

T. Allan Pryor, Paul D. Clayton, Peter J. Haug, Ove Wigertz

LDS Hospital/University of Utah

Abstract

Design of the software architecture for a knowledge driven HIS is presented. In our design the frame has been used as the basic unit of knowledge representation. The structure of the frame is being designed to be sufficiently universal to contain knowledge required to implement not only expert systems, but almost all traditional HIS functions including ADT, order entry and results review. The design incorporates a two level format for the knowledge. The first level as ASCII records is used to maintain the knowledge base while the second level converted by special knowledge compilers to standard computer languages is used for efficient implementation of the knowledge applications.

Implementation of Hospital Information Systems (HIS) at large hospitals is now becoming routine. Initially they were conceived of as administrative and financial systems. They allowed the hospital to admit patients and track the orders and charges on those patients. As clinical systems became available, either as stand alone systems to be networked to the central HIS or integrated within the HIS, new challenges were placed on the design of the HIS. The software architecture of an HIS generally followed two models. The first was a traditional time sharing model where multiple independent processes performing the applications were active at one time and shared the hardware resources of the system. The second model, which has become increasingly more popular, is referred to as a transaction model. In this model the system supports a general screen driver and a series of application servers. By defining low level generic servers, the system greatly reduces the frequent process startup overhead often observed in the time sharing model. With the advent, however, of medical expert systems requiring efficient interaction with the user and a medical knowledge base, the advantages of the transaction model are somewhat reduced. This paper discusses a new software architecture model referred to as the **knowledge driven model** which provides for efficient execution of traditional HIS functions as well as newer expert system technology.

The design of any new software architecture should insure that the functional requirements of the applications incorporated in the HIS are adequately met. In our analysis of HIS applications, we have found that the following functions comprise the necessary and sufficient functions used to create todays HIS applications.
1. Data acquisition including screen formating
2. Data review and editing including hard copy generation
3. Database interaction including storage and retrieval
4. Data analysis including support of mutiple expert systems

With the knowledge driven model we have designed, we extended the content of the knowledge base to incorporate the logical structures necessary to perform all four of the HIS functions enumerated above. We chose the frame as our basic unit of knowledge representation. Through proper definition of the frame structure we are attempting to develop a highly modular knowledge base in which applications are defined in terms of structured groups of frames. Thus a single frame may be used for multiple applications. For example, a frame for acquisition of vital signs may be used by a nurse charting application or a medical expert system controlling hypertension management.

We have chosen a generic frame structure which depending on the frame type (data acquisition, review, diagnoses, management, etc.), will allow the application developer (frame writer) to enter knowledge into frame specific slots. The generic structure of the frame has slots for entry of 1) variable definitions, 2) declarative knowledge, and 3) knowledge documentation. Figure 1 is an example of a diagnostic frame.

The variable definition section of the frame allows the user to define local knowledge variables used in the declarative section of the frame. The user defines a variable by linking it to one or more data descriptors present on the system data dictionary. Our data dictionary defines only basic elements which are captured either through an automated machine interface or computer terminal. The variables, however, defined in the frame may be complex entities which are the result of searches and analysis of the patient's database. For example, a variable may be limited to a simple entity within a fixed time window. Within that time window the defined variable may be the first instance or the last instance or the average of the recorded element. Constraints to other events within the time window may also be stated in the variable definition. Additional information about the variable may also be declared in this section of the frame. This additional information may, for example, declare relationships between the variable and a disease. The relationships could include statistics such as sensitivities and specificities, fuzzy logic associations, associations similar to the concepts of Import and Evoking strength in Internist. Because of the volume and complex nature of the variables, we have chosen to include them in the frame rather than have a separate complex data dictionary. In so doing, the developer and reviewer of the frame is able to immediately see the variables being used in the declarative knowledge and understand their meaning. In figure 1

0195-4210/0000/0060$01.00 © 1987 SCAMC, Inc.

Title: Pneumonia diagnosis (7.141.1)

Type: Diagnosis

Author: Peter Haug.

Date: 12/12/86

Message: "<disease_prob (val; #.##)> Pneumonia (history)".

```
Variables: chest_pain as (DO YOU HAVE CHEST PAIN?),
        cough as (HAVE YOU HAD A COUGH WITH THIS ILLNESS?),
        fever_or_chills as MAX(fever, chills)
        where fever is (HAVE YOU RECENTLY HAD A FEVER?)
        and chills is (HAVE YOU HAD CHILLS RECENTLY?)
        if Exist (fever) or Exist (chills),
                              •
                              •
                              •
Statistics:   for fever_or_chills with TPR(YES, 0.85;  NO, 0.15),
              and FPR(YES, 0.3;  NO, 0.7),
              for cough with TPR(YES, 0.9;  NO, 0.1),
              and FPR(YES, 0.2;  NO, 0.8),
                              •
                              •
                              •
Evoking Criteria: If chest_pain EQ YES or fever EQ YES or chills EQ YES or cough EQ YES.
Logic:        disease_prob = 0.014.
        If Exist(fever_or_chills) then disease_prob = Bayes(disease_prob, fever_or_chills).
        If Exist(cough) then disease_prob = Bayes(disease_prob, cough).
                              •
                              •
                              •
        If disease_prob LT 0.014 then finish.
Ask:  Patients(fever, chills, cough) Heirarchical.
Urgency: 5/9
Gold Standard: If ICD_pneumonia and pneumonic_infiltrate
References: Harrison's Principals of Internal Medicine. Braunwald E, et al (editors)
Validation: Tested experimentally (DD method)
```

Figure 1: Diagnostic Frame for Pneumonia written with the general purpose HELP decision editior. The frame is processed if the criteria in the **Evoking Criteria** slot is met. The Ask slot indicates which information the frame may interactively collect. The **Urgency** slot indicates the relative urgency of recognizing this disease. The **Gold Standard** slot specifies criteria available by the time of discharge which would prove the existence of the disease modeled. **References** refer to literature support for the frame and **Validation** indicates the degree of evaluation to which the frame has been subjected.

variables chest_pain, cough, etc. have been declared. Statistics relating those variables to the disease Pneumonia are also included. The text in parentheses refer to the text associated with the simple entities defined in the data dictionary.

Also within the variable definition section of the frame is a slot used to enter evoking logic. This logic defines the criteria for automatic evoking of the frame when the data contained in the evoking logic is stored in the patient's data file. Because of the need to have formatted screens, special variables called windows are also defined in the variable definition section of the frame. These windows define the screen format of the terminal display and link the entered data to the previously defined knowledge variables. The example in figure 1 has the evoking criteria which would be used to data drive the knowledge of this frame.

The second section of the frame, the knowledge section, contains the declarative knowledge of the frame. The contents of the section is frame dependent. That is, depending on the type of frame being created the knowledge slots available for entry are changed. For example, if the frame were to be used as part of an expert system for disease diagnosis, the knowledge slots would require entry of the associational model knowledge. This could include if-then rules for the diagnosis, Baysesian statistics, or any other appropriate scoring logic. If, on the other hand, the frame were a order entry frame, knowledge concerning the critiquing of the order would be requested. As new applications arise we anticipate that newer frame types will be required which define new slots within the knowledge section.

The knowledge documentation section of the frame is used to enter information regarding the author of the frame, revision dates, references to pertinent literature articles, level of validation of the knowledge, and Gold Standards. The level of validation of the knowledge is intended to provide the developer with a slot where he can declare the clinical status and possible utility of the frame. For example, if the frame was merely the developers first educated guess without any serious validation studies this would be indicated in this section. Thus, anyone using the frame would understand the level of confidence which one might give to the performance of the frame. Likewise if frame had been tested on a large population with satisfactory results, the frame user would be aware of those tests and results. The Gold Standard slot is intended to contain information regarding any gold standard data which may be used to compare the results of the frame. For example, if the frame suggests a new treatment for some disease, the gold standard slot could contain a definition for the successful outcome of that therapy.

Many of the slots we have added to the structure of the frame are intended for use by knowledge management routines. These routines will be used to scan the knowledge base to report to the systems manager critical information on the use and state of the knowledge base. For example, linkages between the data dictionary and the knowledge frames will be maintained, allowing rapid alteration of the frame variables when associated entities in the data dictionary are changed.

The frames, as described here, contain the declarative knowledge of the application. To generate an actual HIS application an additional frame type is required. This frame, referred to as the meta-frame, contains procedural control knowledge necessary to implement an efficient application. For the meta-frame the knowledge section consists of slots controlling the execution of those frames required in a single application. A slot is used to enter the procedural control knowledge of the application. This knowledge may be as simple as the declaration of the initial frame to be executed in a goal driven application or as structured as an ordered list containing the frames and their execution sequence for the application. A second slot defines the inference engine for the application. This again can be as simple as nothing (ie., execute the frames according to the control logic and exit the application) to a hypothesis driven frame scoring algorithm similar to Internist which determines the frames to be executed based on an iterative process of data collection and frame evaluation. Logic pertaining to the storage of results in the database are contained in another slot of this section. The example in figure 2 is a meta frame for control of an arrythmia protocol. This frame contains the logic for control of the execution of a branching protocol. Execution of the frame will cause execution of the declarative knowledge frames of the protocol under the proper conditions.

Title: Arrythmia^protocol

Message: Patient has criteria for arrythmia management protocol"

Frame Type: Meta (protocol)

Evoking Criteria: If Acute^MI and not NSR

Procedural Logic: Evoke IV^Dextrose,

```
        If Hypokalemic then Evoke KCL,
        If rhythm = PVC then Evoke PVC^management and exit,
        If rhythm = Bigeminy then Evoke PVC^management and exit,
        If rhythm = Vtach then Evoke Vtach^management and exit,
        If rhythm = Afib or rhythm = Aflutter then Evoke
        Afib^flutter^management and exit,
        If rhythm = Sinus^tach then Evoke Stach^management and exit,
        If Heart^rate < 60 then Evoke Brady^management.
```

Figure 2: An example of a meta frame used to implement a protocol for the managment of arrythmias. The logic section of the frame controls the execution of declarative frames of this application.

Having developed a design for a frame structure, a conceptual model for knowledge acquisition has been developed. Using this model we are now able to create a knowledge base sufficiently universal to support the applications of an HIS. Examples of the applications we are currently developing using this frame structure include:

1. Physician orders. This application will use the frames to capture knowledge concerning the test/procedure to be ordered. Critiquing knowledge will also be present in the frames. In using the frame based applications the entry of an order will cause the appropriate frame to be executed assisting the physician in entry of the order.
2. Nurse care plans. In this application the frames will contain the knowledge defining the problem and the creation of an appropriate working care plan. For example, many of these frames will be data driven from the physician order frames described above.
3. Monitoring/Alerting. These applications will define frames which are data driven by the receipt of data into a patient's file. They will contain knowledge about the data being stored to determine if alerts need be transmitted to the appropriate personnel.
4. Diagnosis. Application programs here will be driven by frames containing models of the diseases being diagnosed.
5. Treatment Planning. These applications are similar to the nurse care plan application in that knowledge about different treatments will be entered into the frames and executed as the appropriate conditions arise.

This list of applications is only a small fraction of the HIS applications which lend themselves easily to the frame model, but hopefully gives a flavor of the direction of our system. Our knowledge base will be stored as a set of ASCII records. In this form the frames are easy to access, modify, display and transmit to other centers involved in this research. Because of the design of the frames and their stored form they are also machine independent.

The final step of our HIS design is the implementation of the operational form of the applications. In our design, the frames will be transformed into a traditional computer language for execution. We are currently exploring two modes of implementation, a compiled form and an interpreted mode. In both of these modes an intermediate computer language is used as a target language by an application generator. The application generator is driven by a meta-frame which contains the list of frames and the procedural knowledge of the application. On identifing to the application generator the appropriate meta-frame, the application generator assembles the required set of frames and inserts the necessary procedural logic resulting in a single application program in the target language. The derived application program can then either be interpreted or compiled.

Because of the overhead associated with either process startup of compiled applications or interpretation of code, the decision to compile or interpret appears to be application dependent. For example, in an application such as hypothesis driven disease diagnosis where the application would be active for many minutes, the limiting factor is execution speed and not startup time. In such cases compiled code is more efficient, but in a data driven application where only a single frame is to be executed the processing time may be minimal compared to the startup time. Here the interpreted mode is optimal. In either case the important concept is the creation of application programs. The ability to create application modules from the general knowledge base is key to our design and assumes that regardless of the size of the knowledge base, utilization of the knowledge will be limited to specific applications synthesized from a subset of the knowledge present.

We are currently midway through the development of the necessary knowledge editors and compilers. Design of the frame structure has been completed. Simulation studies involving hand compiled frames into an application program written in a target language (Pascal and PAL) have been tested. Our testing has involved actual implementation of some applications on our exisiting HELP HIS at LDS Hospital. These implementations to date, have shown the design described in this paper to be feasible. We are continuing to evolve and plan to have the design operational within one year.