ABSTRACT

Modern organizational information systems increasingly contain multiple subsystems of various types, such as database management systems (DBMS), decision support systems (DSS), and knowledge based systems (KBS). There is a need for rigorous systems development methods that can effectively support the analysis and design of systems with embedded, heterogeneous subsystems. We propose the use of box structure methods and principles for this purpose. The use of box structure methods is illustrated with a case study on an equine nutrition expert system (ENEX). The paper concludes with directions for future research and development.

1. Heterogeneous Information Systems

Modern organizational information systems increasingly contain multiple subsystems of various types, such as database management systems (DBMS), decision support systems (DSS), and knowledge based systems (KBS). Furthermore, for any given subsystem of an information system there may be alternative designs that span various system types. Thus, there are two distinct analysis and design activities to be considered. First, for a given subsystem, a selection of system type is made. Second, a number of heterogeneous subsystems must be effectively integrated within the overall information system.

Traditional methods of information systems development are primarily oriented toward highly structured data processing systems with well-defined procedurality (e.g., Structured Analysis and Design [Yourdon 1989], Information Engineering [Martin 1990]). If all subsystems are of this type, then these methods provide an excellent basis for systems development. However, if less structured subsystems, such as DSS and KBS, are required, then new concepts for system integration are necessary. (Note that we use the term 'knowledge-based systems' broadly to include systems that use domain knowledge and symbolic reasoning to solve unstructured problems. Thus, KBS include expert systems and deductive database systems, for example.)

Recognition of the weaknesses of traditional development methods for DSS and KBS development has led to a number of efforts to produce better design techniques, for example [Sage and Galing 1983, Hayes-Roth et al. 1983, Turban and Watkins 1986, Waterman 1986, Murray and Tanniru 1987, Ramamoorthy et al. 1987]. While many of this studies have proved quite effective for systems falling into specific domain types, they do not address the essential problems of determining when DSS or KBS are required and integrating such innovative system types with one another, and with traditional data processing system components.

What is lacking is a systems development method that:

a) can effectively support the design of various types of systems, including data processing systems as well as DSS and KBS;

b) can facilitate the analysis of what type of system can best satisfy given problem requirements;

c) can identify and specify the interactions among embedded subsystems of different system types; and

d) can support modern system engineering concepts, such as data abstraction, modularization, design verification, etc.

We propose using the box structure methods and principles of systems development for the selection and integration of heterogeneous subsystems [Mills 1988, Hevner and Mills 1992]. In [Basu and Hevner 1991], we have developed a comprehensive set of box structure guidelines for the selection and embedding of KBS.
subsystems in complex information systems. The main objective of this paper is to illustrate the use of these guidelines in a real case study. A brief overview of box structure concepts is presented in section 2. In sections 3 and 4, respectively, we discuss briefly box structure requirements for KBS selection and the application of box structures for KBS design. Then, in section 5, we present the case study, in which several heterogeneous subsystems, including a KBS, are integrated in a complex system. The paper concludes with directions for future research and development.

2. Box Structure Overview

Box structures have an underlying mathematical foundation that permits the scale-up of analysis and design to systems of arbitrary size and complexity. Box structures model system components as data abstractions in three, increasingly detailed forms [Mills et al. 1986]:

- **The black box** gives an external description of data abstraction behavior in terms of a mathematical function from stimulus histories to responses. The black box is the most abstract description of system behavior and can be considered as a requirements statement for the system component.
- **The state box** includes a designed state and an internal black box that transforms the stimulus and an initial state into the response and a new state. The state is designed from an analysis of the required stimulus histories and responses for the system.
- **The clear box** replaces the internal black box of the state box with the designed sequential or concurrent usage of other black boxes as subsystems. These new black boxes are expanded at the next level of the system box structure usage hierarchy into state box and clear box forms.

The effective use of box structures for the development of information systems is guided by the use of four basic box structure principles, referential transparency, transaction closure, state migration, and common services. Here we briefly define each of these principles.

- **Referential Transparency** - Referential transparency occurs when a black box abstraction is completely defined within the clear box at the next higher level in the usage hierarchy. The black box is then logically independent of the rest of the system, and can be designed to satisfy a well-defined behavior specification.
- **Transaction Closure** - The principle of transaction closure defines a systematic, iterative specification process to ensure that a sound and complete set of transactions is identified to achieve the required system behavior.
- **State Migration** - State data should be identified and stored in the system part (i.e., data abstraction) at the lowest level in the box structure hierarchy that includes all references to that data. At any time in the system development process, state data can be migrated upward or downward in the hierarchy in order to achieve some system objective, such as minimizing data scope.
- **Common Services** - A common service is a data abstraction that is described in a separate box structure hierarchy, and used in other box-structured systems. System parts with multiple uses should be defined as common services for reusability.

3. Requirements Determination for Knowledge Based Systems

Our goal is to capture and represent system requirements in a formal, yet intuitive, manner. The formal requirements are reviewed and modified by the system users and become the basis for the system design by the system developers. In this section, we discuss how the black box view of a system can be used to recognize the need for KBS. If this analysis determines that a KBS is viable, then the design decision of the KBS can be made in the state box and clear box.

A system requirement is characterized in terms of three black box features; the stimuli, the responses, and the transactions that relate stimuli with responses. We discuss how each of these features of a desired system (subsystem) can be used in the decision to design the system as a KBS.

3.1 Stimulus Requirements

There are some basic features of required stimuli that directly suggest the use of KBS; for example, the need to allow imprecise inputs and the need to interpret complex inputs (e.g., symbolic inputs such as natural language text and graphical inputs such as images). Actually, given the interactive nature of most KBS applications, it is perhaps more convenient to talk about the user interfaces per se, rather than just the external stimuli. A KBS requirement is suggested by the need for natural language interfaces, since interpretation and construction of natural language expressions generally requires knowledge based reasoning.
3.2 Response Requirements

A significant indicator of the possible use of a KBS is when the responses cannot be stated in terms of a well-defined objective function. Typically, KBS generate outputs that are in the form of acceptable solutions (e.g., suggestions, choices) rather than optimal solutions. While the absence of an objective function is not a sufficient condition for the use of a KBS, the presence of such an objective function can be a factor motivating the choice of a more structured solution than those used in KBS. Another factor to consider is whether there are stringent performance requirements on the system, such as response time and throughput. Given that KBS employ heuristic search based solution methods which have relatively unpredictable behavior, such performance requirements would be difficult to meet.

3.3 Transaction Requirements

Traditional procedural systems have well-defined sets of transactions specified in their black box descriptions. Each transaction is characterized by a functional (n:1) or relational (n:m) transformation. In contrast to these, it is usually difficult and often impossible to specify a transformation for KBS transactions, which are inherently non-deterministic. For instance, in data-driven (forward chaining) systems, there is no predefined relationship between the set of inputs for any instance and the corresponding set of goal states. In fact, it is usually difficult to predict even what inputs are needed until some amount of interaction with the system takes place. This last point is even more significant for goal-directed (backward chaining) systems, where interactive inputs during problem solving are usually hard to predict, since they depend upon the specific rules examined. It follows then that the use of a KBS is suggested if transaction analysis of a system's black box view does not yield well-defined functional or relational transformations.

Another indicator of possible need for a KBS is the identification of transactions that require the collection of additional knowledge from the user. This is likely to occur in a system whose knowledge base is highly incomplete to start, but is expected to evolve and grow as the system is used and the problem domain is better understood. A KBS is potentially useful if the requirements imply the need for a system with the ability to support gradual refinement and augmentation of transactions during system use. This is because of two features of KBS: the use of declarative languages for knowledge representation and the use of search based problem solving procedures that are context-independent (i.e., they are not affected by the modification of the domain knowledge base).

4. The Design of Embedded Knowledge Based Systems

Once the need for a KBS is determined, we use box structures in the design of each of the major components of an embedded KBS: the knowledge base, the inference engine, and the user interface. Of these, perhaps the most critical component is the knowledge base. It is generally accepted that the effectiveness of a KBS is largely determined by its knowledge base. Furthermore research on KBS design has largely focused on effective knowledge representation. In the design of a KBS, the state box of the system is first created. The state of the KBS is the knowledge base. The design of the knowledge base then supports the design of the KBS clear box. The clear box describes the internal processing, which determines the design of the inference engine and the user interfaces.

4.1 Knowledge Base Design

The knowledge base of a KBS contains the domain-specific knowledge available to the system in the domain of its application. Some of the specifications of the knowledge base are:

- the specific problems to be solved by the system,
- the knowledge representation framework to be used,
- the source of knowledge,
- the size, completeness, and stability of the knowledge base,
- the relevant means for knowledge acquisition, and
- the control knowledge (i.e., meta-knowledge, imprecision metrics).

In the state box view, the information obtained about the stimulus history, responses, and transactions from the black box are used to develop an explicit description of the system's internal state. The first cut of state analysis consists of identifying a relevant set of state variables and specifying their structure. Then the state and stimuli are analyzed to determine if they are sufficient to support all transactions. If not, the state description is augmented/revised and the process repeated until transaction closure is achieved.

As described earlier, the transaction specifications are typically incomplete for KBS subsystems. The process of state analysis helps in various ways to organize the knowledge base for this incomplete set of transactions. First, the iterative process of transaction and state refinement enables better understanding of what knowledge is needed. In [Basu and Hevner 1991], we have developed a structured procedure for development of a rule base for KBS based on box structured information. The state
analysis phase also provides a basis for re-examining the feasibility of a KBS. For instance, if in a rule-based setting, we find that the set of known rules is very small and much of the necessary responses and state information have to be in the form of data (either state or stimulus), then an effective KBS may be infeasible and requirements may have to be modified. A KBS is feasible only if a relatively substantive knowledge base can be constructed.

4.2 Inference Engine Design

The inference engine determines how heuristic search through the knowledge base is used to solve problems. A central feature to be determined is the control strategy to be used, either backward-chaining or forward-chaining. The choice of control mechanism primarily depends upon the input-output requirements which are part of the black box view of the system. The knowledge base representation of the state box must support the chosen control strategy. Additional features of the inference engine, such as conflict resolution strategies, search strategies, and additional control structures, are analyzed from the clear box view.

4.3 User Interface Design

Since KBS users are typically non-programmers, it is important that system interaction be as easy and flexible as possible. Natural language understanding is usually a desirable feature, although the extent to which it is supported in an important design decision. In systems designed for very specific domains with extensive domain lexicon, effective interaction may be achievable with relatively limited natural language support. In such situations, use of sophisticated facilities for general purpose natural language processing would be expensive, cumbersome, and unnecessary. Where natural language understanding is needed, its complexity and heuristic nature motivates the use of knowledge based techniques for the interface itself. In fact, it may be feasible to design an embedded KBS for language processing within the larger system, regardless of whether the latter is a KBS or not.

The principle of referential transparency is useful since it enables both the interface of the overall system (synthesized from those for all components) to be represented in its black box view and also the interface requirements of each subsystem to be represented in the corresponding black box descriptions. Furthermore, the requirements of the different subsystems can be analyzed, and where relevant, common services can be defined for user interface functions. For instance, in the case study system described in the next section, the interface require-

5. A Case Study: Development of ENEX

In order to demonstrate how box structures can be used to recognize and develop embedded KBS, we apply them to the development of an integrated decision support system for equine feed ration determination called ENEX (Equine Nutrition Expert) [Wangle 1989], which has been implemented at the University of Maryland. This system is intended for use by horse owners and veterinarians, to help determine the optimal feed mix for a horse.

5.1 Requirements Analysis for Equine Nutrition Expert (ENEX)

At the most general level, the desired behavior of the ENEX system is as shown in the black box of Figure 1; that is, given information on horses and feeds as stimuli, the system must use these to determine a horse feed ration for each horse. State analysis of this black box indicates that all the necessary stimuli need not be external inputs. For example, information on available feeds is largely predetermined and stable and, thus, can be stored as elements of internal state in the system. Thus, at the top level, ENEX has two major subsystems, ENDB (Equine Nutrition Database) and HFRD (Horse Feed Ration Determination), as shown in the clear box of Figure 1.

Analysis shows that the feed ration determination process for any horse can be decomposed into two distinct steps:

1. determination of the different individual nutrients required by the horse, and the amount of each nutrient needed in each scheduling period (e.g., day, week, month); and
2. determination of a feed ration that combines appropriate quantities of available foods to provide the necessary nutritional mix at the lowest cost.
The second step is needed because the different feeds available have different amounts of various nutrients. Thus, a diet typically required nutritional balance at the lowest cost. Since the two steps are distinct, the clear box of HFRD can be modeled as a sequence of two component black boxes, Horse Requirements Determination (HRD) and Feed Ration Determination (FRD), as shown in Figure 2. Each of these can be analyzed individually as box structured systems.

5.1.1 Horse Requirements Determination (HRD)

Black box analysis identified the stimuli to this box to be various types of information about horses; the system response is then the quotas of each nutrient required by the horses analyzed. At the state box level, some of the stimuli, such as historical data on horses, can be maintained as state (if the system is to be used by a single owner, for instance), while current data on horses are provided as inputs. An input into the analysis process of this subsystem is a set of mathematical models (in the form of equations) developed by the National Research Council (NRC), which can be used to estimate the basic nutritional requirements [NRC 1989]. However, these models are based on the assumption that the horse is a normal, average horse in some well-defined category. In addition, the horse may need supplementary nutrition, depending upon its stage of growth, work characteristics, lactation, and gestation. The NRC models consider all these factors in computing the overall basic requirements, which can be used by a veterinarian or horse owner.

As noted earlier, the NRC models are designed for normal, average horses. However, relatively few horses can be classified as such. Consequently, the basic nutritional requirements need to be modified, typically by an equine nutrition expert or veterinarian, to account for various factors such as abnormal weight, injury, and unusual work loads. In effect then, the HRD system has two transactions, basic nutrition requirements determination (BNR) and refinement of nutrition requirements (RNR), as shown in Figure 3.

Examining BNR, we find that the stimulus to it is a request for the nutrition requirements of a specific horse, to which the system responds with a list of nutrients and optimal amounts per period for each, under the assumption that the horse is a typical horse for its age, race, etc. The state consists of horse information on factors such as age and race, and the embedded black box contains state transitions defined by the NRC equations. Since these are explicit and well-defined, the clear box for BNR is structured and fully specifiable.

The other subsystem, RNR, is considerably more complex. To illustrate the analysis of this subsystem using box structures, we outline the analysis of its features in terms of the stimulus, response, and transaction requirements:

**Stimulus Requirements**
- Given that the specific factors relevant to a particular horse is possibly a small subset of the large set of known factors, a significant requirement for RNR is the determination of this subset itself, so that the user can be prompted for only that information that is relevant. Thus, interactive user inputs are clearly important.
- Since the user (who might be a horse owner or veterinarian) has to respond to diverse prompts for information that are very specialized to the equine domain, the user interface should be in terms that is relatively flexible and English-like.
- The problem domain under consideration is relatively complex and ill-structured. For example, the nutrition analysis process has to account for a variety of special cases and exceptions, all of which are unlikely to be encapsulated in ENEX. Hence, in order to be useful, RNR must be built so that situation-specific rules, data, and other knowledge can be added easily.

**Response Requirements**
- At first glance, the intended response seemed well-defined, namely the appropriate daily quota of nutrients for a specific horse. However, on closer examination, it is clear that the notion of appropriateness is situation specific. The factors that apply to any particular horse are a subset of a potentially large set of factors, and furthermore, the evaluation criteria for a solution are also instance specific. In some cases, the goal might be peak performance for a horse, in others resistance to disease. The resulting conclusion from the analysis is that the output from RNR can be a satisfactory solution at best.
- It is clear that a typical user of RNR, or even ENEX, would not mind reasonable computational delays (even of the order of minutes), if the system generated useful results. Hence, the efficiency requirements are not very stringent.

**Transaction Requirements**
- The specific transactions implemented in RNR were only specifiable at a very general level. In fact, as suggested by the above discussion, the transactions are both incomplete and evolving.
Due to the need to store, access and even potentially modify various rules used to analyze the nutrition needs, it is clear that these rules cannot be conveniently embedded in procedural modules, but must instead be stored declaratively in complex data structures. Thus, it seems that a KBS might be an appropriate system type for RNR. However, determination of the feasibility of such a system requires further analysis, at the state box and clear box levels, to determine whether the available domain knowledge is in fact sufficient to build a useful system. Fortunately, the various factors that need to be considered for nutrition refinement (e.g., general appearance and configuration, weight, discharges, anatomical irregularities, injuries, work characteristics, environment, metabolism) can in fact be identified by experts in the area. In other words, we can identify the major embedded black boxes in RNR. Moreover, although there are no means of interconnecting these boxes in a functional manner (to yield algorithms or procedures), domain experts could provide heuristics that characterize the impact of each of the factors upon the nutritional needs of a horse. This indicates that in fact, an expert system for nutrition refinement is feasible.

5.1.2 Feed Requirements Determination (FRD)

In FRD, the second subsystem of ENEX, we found that at the black box level, the stimuli consist of the nutrition requirements for a specific horse and the composition, cost and other characteristics of various available feeds. The desired response is a feed ration that achieves the necessary nutritional balance at lowest cost. At the state box level, we found that feed information is stable and predetermined, and can be stored as part of the state. The nutrition information clearly cannot, since it is generated by the HRD system. Analysis of the embedded black box in FRD indicates that we have a well-defined objective function which can be stated as a linear expression, as well as a well-defined set of constraints, each of which is also linear. This implies that the necessary transitions can be specified as a linear program, and thus the subsystem does not need further decomposition for purposes of analysis. The linear program is of the form:

$$\text{Min } \sum_{i=1}^{n} Cx, \text{ where } Ax \geq B, \ x \geq 0.$$  

where $c_i$ is the unit cost of the $i$th feed, $x_i$ is the amount of that feed in the ration, $A_{ij}$ is the amount of nutrient $j$ in each unit of feed $i$, and $b_j$ is the stated requirement of nutrient $j$ in the horse’s feed ration.

The box structured analysis of the ENEX system illustrates the utility of the approach for one aspect of the development of embedded KBS, namely recognition of appropriate use of embedded KBS (in this case RNR) within a larger information system. This in itself is extremely useful, since it enables the scope of the KBS to be clearly defined. It should be easy to see that if the preceding analysis were bypassed, and ENEX predetermined to be an expert system, likely consequences would be either an ineffective design, or extensive redesign once the structured characteristics of systems such as BNR and FRD were exposed, or both.

5.2 Design of ENEX

The utility of box structures extends throughout the design and implementation of the ENEX system. After the initial box structure description is developed, analysis of the various components leads to the specification of appropriate control structures in the different clear boxes, and the generation of an initial system specification in box description language. While space constraints preclude a detailed discussion of the use of the methodology for the complete design of ENEX (see [Wangle 1989] for details), we describe how some of the box structure design principles guide the system design.

- The principle of state migration is used effectively to streamline the system design. For instance, we find that horse information is only required in the HRD subsystem, and feed information only in FRD. Furthermore, within HRD, information on horses is required both in BNR and RNR. Given this, and the fact that additional information may be acquired from the user in RNR, the principle of common services can be applied. This implies that a common service box should be used with all the horse information. Further analysis of the state descriptions of BNR and RNR lead to the decision to use a DBMS to create and maintain the horse information database.

- The principle of referential transparency provides the ability to replace subsystems if the black box behavior of the subsystems are equivalent. In ENEX, we built the first prototype using a Prolog database for the horse information, since that was easier to interface with the Prolog-based expert system for RNR. In the final implementation, we employed a relational DBMS because of its enhanced flexibility and functionality.

- The integration of all subsystems in ENEX is achieved through a common user interface. As mentioned previously, the design details of the
three principal subsystems; the DBMS (ENDB), the expert system (RNR), and the linear program (FRD); are hidden from the user, who views the system as a comprehensive information for equine nutrition.

5. Conclusions and Future Research Directions

Box structure development methods support the selection of subsystem type during requirements determination. We discuss the black box requirements that indicate the choice of a KBS. Further, box structure design methods and principles provide the necessary ability to design and implement complex systems with heterogeneous subsystems. The ENEX case study illustrates the development of such a system that includes an embedded KBS.

Although the conceptual basis for the use of box structures is sound, a number of issues must be addressed to make box structure methods effective for KBS development:

- New methods for requirements gathering, modeling, specification, and analysis are needed to formalize the requirements determination process.
- Box structures can be enhanced to include representation and analysis of the unique aspects of KBS development (e.g., inference engine selection, knowledge base representation).
- Improved CASE support for box structure methods are needed.

References


FIGURE 1: BLACK BOX AND CLEAR BOX VIEWS OF ENEX
FIGURE 2: BLACK BOX AND CLEAR BOX VIEWS OF HFRD
FIGURE 3: BLACK BOX AND CLEAR BOX VIEWS OF HRD