HyM: a Hybrid Methodology for the Development of Integrated Hybrid Intelligent Information Systems

S Kendal, X Chen. A Masters
School of Computing, Engineering and Technology
University of Sunderland, St Peters Campus, Sunderland, UK SR6 0DD

ABSTRACT

HyM is a hybrid methodology for the development of large-scale and complex integrated hybrid intelligent information systems, which combines traditional information system development methods with knowledge-based system development methods. The methodology is an integration of four existing methods using two integration process approaches: intra-process and inter-process. An intra-process approach is then used to integrate these techniques together using consistency rules. In the design phase, the inter-process approach is used to transform the requirements analysis to object-oriented design. Finally, the object-oriented method is applied to the design and implementation of hybrid information systems. This methodology takes advantage of the four individual methods to overcome the limitations of each. It is applicable to the development of traditional information systems, knowledge-based systems, and large and complex cooperative and intelligent information systems.

This paper describes the HyM methodology and its successful application when used to develop a hybrid intelligent information system in a complex medical domain.

1. INTRODUCTION

A new challenge for our information hungry society is the study of hybrid information systems (HISs) that combine traditional information systems (ISs) with knowledge-based systems (KBSs) [Brodie and Ceri, 1992]. However due to the complex nature of HISs, it is unrealistic to expect that they can be developed using one existing standard method. The use of several independently developed methods has a number of drawbacks, such as inconsistency, redundancy, amount of effort required and possible loss of information. The development of an integrated methodology is a possible solution [Kronlof, 1993].


Besides the integration of structured analysis and object-oriented design, methodology researchers considered other combinations of methods. For example, Chen [Chen and Chen, 1994] provided an integrated methodology to integrate Jackson structured programming method and object-orientation. Nielsen [1992] and Kuo [1994] described integrated strategies to combine information modelling techniques with the object-oriented method. Hallmann and Coors [Kronlof, 1993] built up an integrated methodology that combined three methods: structured analysis (SA), information modelling method (IM) and hierarchical object-oriented design (HOOD), called SA/IM-HOOD.

However, these integrated methodologies only consider problems in conventional software engineering; they do not consider problems related to the development of KBSs, so they are more suitable for traditional IS development than an HIS. Almost all of these integrated methodologies only support the integration of two different methodologies.

Early integrated methodologies proposed for the development of KBSs focused on the combination of two or more knowledge acquisition methods (KA) and hybrid knowledge representations to overcome problems in the requirements analysis and system design phases [Leung and Wong, 1990; Debenham, 1990; Czejdo et al., 1993; Barghouti and Sokolsky, 1991]. However, almost all these methodologies emphasise knowledge engineering concepts and methods, and pay little attention to traditional IS development.

Since the early 1990s, some researchers have worked on integrated methodologies for HIS development [Gillies, 1991; Lehmann, 1993; Connors, 1992; Harris-Jones et al., 1992; Plant and Tsoumpas, 1994; Angela, 1990]. These integrated methodologies mostly emphasise the integration of one conventional method in software engineering and one KBS development.
method, and ignore many advantages of other recent methodologies.

In an attempt to provide at least a partial solution to these problems, we propose a new Hybrid Methodology (HyM) for developing hybrid information systems. HyM combines methods for developing traditional information systems with methods for developing knowledge-based systems (KBSs). The new methodology provides a hybrid lifecycle process model to combine the conventional waterfall process with rapid prototyping and model-based approaches.

2. HIS ARCHITECTURE AND LIFE-CYCLE

2.1. The Hierarchical Architecture

For the development of complex software systems, we propose a software system hierarchical architecture concept which is based on four levels. From bottom to top, the four levels are:

- Code and repository level: The bottom level. It deals mainly with technologies for the design and implementation of the database, the knowledge base and procedures.
- Component level: It consists of all models that integrate data, knowledge, and procedures based on modelling objects and entities.
- HIS level: It implements combinations and linkages of objects to build an HIS, and requires good interoperability. Conventional information systems or knowledge-based systems are subsets of an HIS.
- LICs (large Intelligent and Co-operative Information Systems) level: The top level. It studies systems integration, which integrates existing software systems or extends new products to old ones.

The hierarchical architecture, proposed here, provides two views for the development of complex systems. From the technical view, these four levels are independent of each other, e.g., there seems no direct relationship between techniques of database design in the lowest level and modelling technologies in the component level. On the system view, they are interrelated, e.g., every HIS in the HIS level consists of components in the component level and their data, knowledge and procedures are stored and implemented in the lowest level. Any complex software system can be constructed using this architecture. Many current software techniques can also be mapped onto these levels. From the view of the software development process, the traditional top-down process starts in the HIS or LICs level. Model-based processes are used to form the component level.

Based on the proposed hierarchical architecture, the development of HISs goes through three levels, i.e., the code and repository level, the component level and the HIS level. An integrated methodology is proposed to support this. The proposed methodology has not yet been expanded to support the LICs level.

2.2. Proposed HIS Development Process

HyM proposes a hybrid process model that combines three conventional process models: waterfall process, rapid prototyping and model-based approaches, as shown in Figure 1.

Figure 1. The HyM lifecycle model

This new lifecycle model has an internal process (the shaded parts of the above diagram): a rapid prototyping process, which crosses phases of requirements analysis and system design. Both phases are not separate in this lifecycle process. Instead there is a gradual move from analysis phase to design phase. The internal process includes four steps related to system models: model analysis, model design, model testing and model evaluation. There are three types of basic models in an HIS, i.e., data model, procedure model and knowledge model.

3. HyM METHODOLOGY

3.1. Requirements Analysis and SA/IM/KA Integration

The first step in the proposed HIS development lifecycle process are to establish feasibility and then a combined analysis / design phase. Because an HIS consists of three types of components: procedures, data and knowledge it is necessary to integrate three existing methods in the requirements analysis stage using an intra-process approach.

The basic criteria for method selection in the HyM are based on method practicality, efficiency and compatibility. The strategy for choosing methods to be integrated is based on existing standard and popular methods that are easy to use and familiar to most software developers. In the HyM, structured analysis methods (SA) are selected to analyse system structured procedural functions. The IM method is selected for the data analysis. The knowledge analysis method (KA) of model-based KADS is selected to perform the knowledge analysis.
In order to describe the SA/IM/KA integration process clearly, there are some basic concepts to be defined. They are associated with three types of system models: procedure, data and knowledge.

**basic procedures**: procedures that can not be refined, i.e., terminators.

**macro procedures**: procedures that consist of other some smaller procedures, i.e., unified procedures.

**atomic data store**: data store that can not be refined, i.e., basic data store.

**molecular data store**: complex data store that consists of atomic data stores.

**domain knowledge**: static knowledge that describes the declarative theory of an application domain.

**control knowledge**: dynamic knowledge that includes inference knowledge, task knowledge and strategic knowledge.

The proposed integration strategy for SA/IM/KA is based on the Extended Data Flow Diagram (EDFD). The concepts usually found in a conventional DFD of SA are still used in an EDFD, such as clients/users, procedural processes, data flows and stores. Besides those notations for procedure and data components, the EDFD also provides new elements suitable for knowledge components. The KADS methodology categorises knowledge into domain knowledge and control knowledge. Furthermore, control knowledge is classified into inference knowledge, task knowledge and strategic knowledge. From the view of functions, control knowledge can be considered as a deductive or hybrid procedure that functionally is the same as a structured procedure in the traditional DFD of SA. In an EDFD, domain knowledge is identified by the knowledge store notation, i.e., static knowledge. Control knowledge is presented by the deductive procedure notation, i.e., unstructured procedures. The data flow concepts of DFD in SA are also extended to support the data/knowledge flow concepts of EDFD.

Considering an intra-process approach, it is necessary to provide complete and strict rules to guarantee consistency during SA/IM/KA integration. From a rigorous view, every element in an EDFD should correspond to its related element in an ERD or a structured knowledge analysis diagram. For example, an atomic data store in an EDFD is related to an entity in an ERD, which needs one consistency rule. In general, there are eight types of element in an EDFD: atomic data store, molecular data store, domain knowledge store, data flow, knowledge flow, structured procedure, deductive procedure and hybrid procedure. These require eleven consistency rules as follows:

- Each atomic data store in an EDFD is declared as a collection of instances of an entity in an ERD.
- Each molecular data store in an EDFD is a unification of other data stores and can be refined.
- Each static knowledge base in an EDFD corresponds to a domain knowledge model.
- Each domain knowledge model is declared as a collection of knowledge representations of a knowledge base in knowledge analysis diagrams.
- Each data flow in an EDFD has to be declared within the data view, i.e., by an entity type, a relationship or an attribute type in an ERD.
- Each data flow in an EDFD going into or out of a data store must be a part of the data store entries.
- Each knowledge flow in an EDFD has to be declared within one knowledge representation method, such as rule, semantic net, frame or other type.
- Each knowledge flow in an EDFD, going into or out of a static knowledge base, must be a part of the knowledge base entries.
- Each structured procedure in an EDFD is declared as a procedure in a traditional DFD.
- Each deductive procedure in an EDFD corresponds to control knowledge that may be an inference engine, task knowledge or strategy knowledge.
- Each hybrid procedure in an EDFD is a complex procedure with both properties of structured procedures and deductive procedures.

The integration approach takes the data or procedures (procedural, deductive or hybrid) in SA or KA, and replaces it with, or relates it to, the data view represented by IM. Domain knowledge in KA has been taken by deductive or hybrid procedures and replaced with, or related to it, a static knowledge base.

### 3.2. Design and SA/IM/KA-OO Integration

An inter-process approach is used to transform SA/IM/KA requirements analysis to an object-oriented design, i.e., SA/IM/KA-OO integration. The key strategy in the inter-process approach is in the definitions of transformation and tracing rules. As mentioned above, the methodology applies three analysis methods for hybrid system analysis. Therefore, the transformation process is based on three views: SA, IM and KA, corresponding to EDFD, ERD and structured knowledge analysis diagrams respectively. The transformation rules are defined based on stores, flows and procedures, which makes the inter-process more rigorous. The results from SA/IM/KA integration produce corresponding relationships among them so that some transformation objects need to be merged or inherited by others. Tracing rules are defined to solve these problems.

- Each atomic data store in an EDFD corresponds to a simple data object in an ERD.
- Each simple entity in an ERD, which maps to an atomic data store in an EDFD, is transformed to a data object exporting the relevant entity data types.
- Each static knowledge base in an EDFD, which maps to a domain knowledge model in a knowledge
analysis diagram, is transformed to a knowledge object exporting domain knowledge.

- Each domain knowledge model corresponds to a knowledge object in a structured knowledge diagram.
- Each basic procedure in an EDFD is transformed to a simple procedural operation object.
- Each procedural operation object corresponds to a structured procedure in an EDFD.
- Each control knowledge model is transformed to a knowledge operation object.
- Each knowledge operation object corresponds to a control knowledge model, i.e., a deductive procedure in an EDFD.
- Each knowledge flow in an EDFD is also transformed into messages of the objects that are linked by it.
- Each data flow in an EDFD is transformed into messages of the objects that are linked by it.
- Each molecule data store in an EDFD corresponds to a complex data object in an ERD.
- Each complex entity in an ERD, which maps to a molecule data store in an EDFD, is transformed to a complex data object that inherits other small data store objects.
- Basic procedures, or macro procedures to handle atomic stores, are transformed into objects that inherit the relevant procedural operation objects and data objects.
- Knowledge objects and knowledge operation objects create complex inheriting objects that encapsulate relevant domain and control knowledge.

According to the concepts and the development lifecycle process, each model analysis is first transformed to a model design and then these model designs are transformed into an HIS system design. Based on the ideas of object-oriented layer, this transformation can be performed using an algorithm. Considering an inter-process approach, the proposed transformation algorithm is made up of several steps:

1) The transformation process starts from the bottom layers of an EDFD, and then gradually extends to the upper layers. It is an iterative procedure.
2) This step identifies the layer of OO classes or instances to be transformed. When the transformation happens on a bottom layer, the process is performed by step 3. Otherwise, the process is performed by step 4.
3) On the bottom layers, the transformation and tracing rules are used, which are described above. In this transferring process, transformation rules are first applied from three views: function (SA), data (IM) and knowledge (KA). This step realises the transformation of bottom layers that may consist of EDFDs, ERDs and structured knowledge analysis diagrams. Then, according to their mapping relationships, tracing rules are applied to produce new inheritance objects. On the bottom layers, the transformation processes mostly consider a simple or an inheriting process.
4) On the other layers, there are no atomic data stores and basic procedures or domain knowledge. Every structured or unstructured procedure is transformed to a complex object that encapsulates all components in its lower refining layers. The data or knowledge flows are transformed to triggers and messaging operations, which are also components of objects.
5) Repeat the Step 2 to Step 4 above until all the data, procedures and knowledge are transformed and designed.

4. AN EXAMPLE APPLICATION

HMISD, called Hybrid Medical Information System for Dizziness, is a typical HIS for the diagnosis of vertigo diseases. It was developed as part of this research project. This system involves many activities that often happen in hospitals and GP offices, such as registration, medical record management, clinical diagnosis, laboratory information management, clinical research and drug data management. This complex software system consists of those components in traditional medical information systems and medical knowledge-based systems.

The HMISD software was based on the requirements of North Riding Infirmary (NRI), Middlesbrough, a hospital specialising in Otorhinolaryngology.

The new integrated methodology was successfully applied to the development of HMISD throughout all development process phases. HMISD was developed in three levels: HIS, component, code and repository level.

4.1 Domain Analysis and Feasibility Study

Dizziness is a common complaint and can be a symptom of numerous disease processes. These can vary from psychogenic disorder in origin to presentations of serious intra cranial pathology. Wright describes Dizziness as follows:

"Dizziness is a difficult condition, and having to diagnose and manage the dizzy patient may seem like being thrown in at the deep end when you can only just swim." [Wright, 1988]

Unfortunately dizziness is one of the most difficult conditions to diagnose as the symptoms are vague requiring time consuming consultations [Chen, X. et al., 1995]. Dizziness, in any of its many different forms is not a disease, as such, but the result of some underlying disorder. It is often very difficult for ‘dizzy’ patients with problems to put into words their exact feelings but it is important to establish what it is they feel, so that time, patience and the right questions are often needed to help them express themselves. Taking a proper history is really the key to diagnosing dizziness but to do this requires the knowledge of the causes and associated symptoms so that a realistic set of questions can
be posed. Examinations and investigations are often used to observe the effects of diseases. What examinations or investigations should be applied, and what effects impact on the diagnosis, are also not easy to identify [Wright, 1988].

The system developed is a hybrid medical information system that incorporates the traditional medical ISs and medical KBSs. It allows patients or clerks to input medical records and symptoms as with a traditional medical record system. The system suggests the most probable disease diagnosis and gives suggestions as to the most suitable investigations. The system prints a final diagnostic report that is sent to the patient’s general practitioner. Results suggest that this system will increase diagnostic accuracy, reduce unnecessary investigations, save time, improve clinical research and support teaching.

The five major components of the system are described below.

The medical record system (MRS): is a system that manages and organises patients’ medical records including profile, past history, family history, social history and medical history.

The clinical decision support system (CDSS): is a medical expert system. It assists doctors in diagnosing diseases that cause dizziness, and proposes appropriate medical examinations and tests.

The pharmacy system (PS): contains a drug database that stores drug information, such as indications, cautions, contra-indications, side-effects, interactions and dosage.

The laboratory information system (LIS): manages and handles medical examinations and investigations, such as names, types, indications, possible results, technical descriptions.

The clinical research system (CRS): allows doctors to access the patient database and perform medical statistical analysis.

HMISD also includes several major databases and knowledge bases.

4.2 System Evaluation

Ninety three patient records were used to test and examine HMISD performance. During testing, computer clinic and consultants clinics were kept separate so that their diagnoses were independent of each other. HMISD diagnoses were evaluated by comparing with diagnoses made by consultants and registrars. From analyzing ninety three dizzy patients, there are six major diseases that cover above 85% cases. These major diseases include: Menieres Disease, Psychogenic Disorder, Migraine, Vertebro-Basilar Ischaemia, B.P.P.V. and Postural Hypotension. Other dizzy diseases include: Brainstem Stroke, Vestibular Neuronitis, Epilepsy and Alcoholism.

Table 1 illustrates the statistical results regarding ninety three patient cases diagnosed by HMISD and consultants, and indicates the rates of correctness. As indicated, the average accuracy rate of HMISD for dizziness diseases is about 85%.

<table>
<thead>
<tr>
<th>Disease</th>
<th>No of Patients</th>
<th>Accuracy rate %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psychogenic Disorder</td>
<td>20</td>
<td>95.0</td>
</tr>
<tr>
<td>Migraine</td>
<td>12</td>
<td>83.3</td>
</tr>
<tr>
<td>V.B.I.</td>
<td>11</td>
<td>72.7</td>
</tr>
<tr>
<td>B.P.P.V.</td>
<td>14</td>
<td>57.1</td>
</tr>
<tr>
<td>Menieres Disease</td>
<td>17</td>
<td>100.0</td>
</tr>
<tr>
<td>Postural Hypotension</td>
<td>7</td>
<td>100.0</td>
</tr>
<tr>
<td>Other</td>
<td>12</td>
<td>83.3</td>
</tr>
<tr>
<td>Total 93</td>
<td>Ave 84.5</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Statistical data of ninety three patient cases for HMISD diagnoses verification

From the investigation of twelve medical professionals and twenty-five patients, the user’s view of the system is as follows:

80% patients thought that HMISD was helpful or very helpful when describing their symptoms.

100% medical staff thought that HMISD could assist doctors for dizziness diagnosis.

100% medical staff thought that HMISD can increase diagnostic accuracy, reduce unnecessary investigations, search drug information, manage medical records, and save time.

5. CONCLUSION

In this paper a hybrid methodology (HyM) for the development of HISs is proposed. It integrates four existing methods using intra-process and inter-process approaches during the requirements analysis and system design phases of software lifecycle. Main benefits using this method include:

HyM covers all phases of the software lifecycle. It is applicable to the development of traditional information systems, knowledge-based systems and HISs. It takes advantage of four existing methods and removes many of the limitations imposed by using any individual method. Because it is based on existing methods many software developers should be able to understand and apply this method easily.

In the intra-process approach, consistency rules are provided to realise SA/IM/KA integration so that existing methods can be safely and effectively used together. In the inter-process approach, a transformation algorithm is also proposed to realise SA/IM/KA-OO integration.

The software development life cycle proposed promotes a smooth transition from analysis to design, this is particularly important for large complex systems. It has a tight, highly iterative, quality control loop to ensure the consistency and completeness of KBS compo-
ments while allowing more straightforward procedural components to be developed using a traditional waterfall approach.

HyM, as described in this paper, only focuses on the integration of selected methods in the requirements analysis phase and the transformation from analysis to design. There are still some areas that require development, e.g., considerations for model testing and evaluation criteria, and real-time system development.

In conjunction with the NRI a hybrid medical information system for dizziness was developed. Results were very satisfactory [Chen, et al. 1996].

REFERENCES


