MODELLING INVESTMENT DECISIONS IN INTELLIGENT SYSTEMS: A SEMANTIC APPROACH

Dr. P. Notopoulos

Department of Accounting
T.E.I. of Serres

Dr. G. Exarchos

Department of Accounting
T.E.I. of Serres

ABSTRACT

Although the interface/task separation architecture has been a preferred approach in implementing user interfaces for intelligent systems in a business context, little work has been done in application modeling, an essential component of such systems. In this paper, we propose a logic–based approach for representing application modeling; after presenting the theoretical tool for analyzing the resultant dialogue graphs, we provide a semantically adequate basis for implementing this approach, together with the various structural and functional characteristics of a such system. Finally, an example of an application modelling is presented in implementing of simple intelligent system for investment advising and we conclude that this design approach in implementing intelligent interfaces in the context of Knowledge-based Systems , is a simple but powerful one and can be adopted for any single application in that context

Key words: Application modeling, Dialogue networks, Financial modeling.

INTRODUCTION

Managers are faced today with decision-making tasks which require sophisticated computer-based Management Information Systems augmented with ´intelligent´ Decision-support (D.S.S.) and Expert systems (E.S) [16,26]. Although the application of these tools can be of enormous help to the decision-making process, the decision still rests upon the doctor and consequently DSS/ES success is heavily people-dependent.

The User-Interface component is responsible for managing the dialogue between people and application programs by providing the specialized function necessary to handle the interaction. However, in today Knowledge-based Systems , the communication between the user/manager and the various application programs is restricted, either in linguistic form or through the use of windows-based environments (pull-down menus, overlapping windows, etc.)[1,10,24]. This interactional approach is ideal when the manager knows the logical structure of his intended tasks. However, the multiplicity of unique commands provided by the designers, of such systems, in their efforts to reflect each nuance of the system functionality, puts a major obstacle to mastering tasks.

This has as implication the need for adopting a “semantic/cognitive” factors viewpoint [1,2,23], by incorporating semantic [4,21] features of the manager’s view of the task domain. This approach, since it attends to the way in which the manager thinks of the tasks he has to achieve, reflects his view of the task domain in terms of the goals he wants to accomplish. The adoption of this approach, in turn, requires a shift in the design of user interfaces in order to match more naturally with the way the manager conceives his tasks. This can be accomplished by providing, separation capabilities of the particular task to be performed, from the user-interface dialogue management systems.

One of the more interesting questions is how to integrate Intelligent Systems into an existing Management Information System, specifically into a D.S.S., in order to take advantage of the strong points of both the D.S.S. and Expert Systems in the management of business information.

INTERFACE MANAGEMENT AND DESIGN

b1) Task – Dialogue separation

Traditional methodologies for designing and specifying Information Systems have been concentrated upon the decomposition of the application domain into more manageable

sub-problems, i.e. Jackson Structured Design [19], S.S.A.D.M [13], Vienna Development Method [20], thus focusing upon implementation issues without addressing usability issues. However, the consideration of applications interfaces requirements [14] results in a different approach to structural decomposition, one that favors the separation of interface aspects from the computational aspects of the application.

It is now generally accepted that the separation of application/task aspects from the user/interface dialogue is a desired objective [14,15], because it permits the change of the system as a result of user experience and the presentation of different views of the application of user with different requirements.

Edmonds [15] proposes that the interactions between a user and a computer-based application can be considered as involving three main processors: a human being, an interface processor and a task processor; further he divides the interface processor into three sub-components:

a presentation system (I/O processes)

a dynamic processor (interaction manager)

a description of the application

This general paradigm of task-dialogue separation has been further refined by Jerrams and Smith [16], who proposed the representation of user-model as a distinct module in the interface processor and by Green [17], who incorporated the application description into what he termed application model, which contains the user's view of the semantics of the application.

The last point (application model) has caused conflicting views between researchers, as whether or not it should incorporate aspects of user-modelling and at the present little work exists on application modelling. Ball and Hays [18] represented all task-specific knowledge in a declarative data-base called "Tool Description" which is accessed via a "User Agent", who establishes what functional capability of the application is required, from the users point of view. Thus, in terms of Edmond's model the Tool Description is an application model and the User Agent is a kind of dynamic processor.

In discussing intelligent front-ends, Probert [20] identified three major components: a user-model, a dialogue-assistant and a system model; the last one plays the role of "road map" to the application and therefore is a kind of application model.

This task-dialogue separation architecture makes the specification user interfaces clearer. Previous user interface specifications have suffered because they lacked an acceptable language for describing the "semantic" of the interface, that is the actions that the system performs in response to the user's commands. An important element in the effort to incorporate "semantic" features in such an approach is to apply formal specifications techniques. These suggestions are shown in Fig. 1.

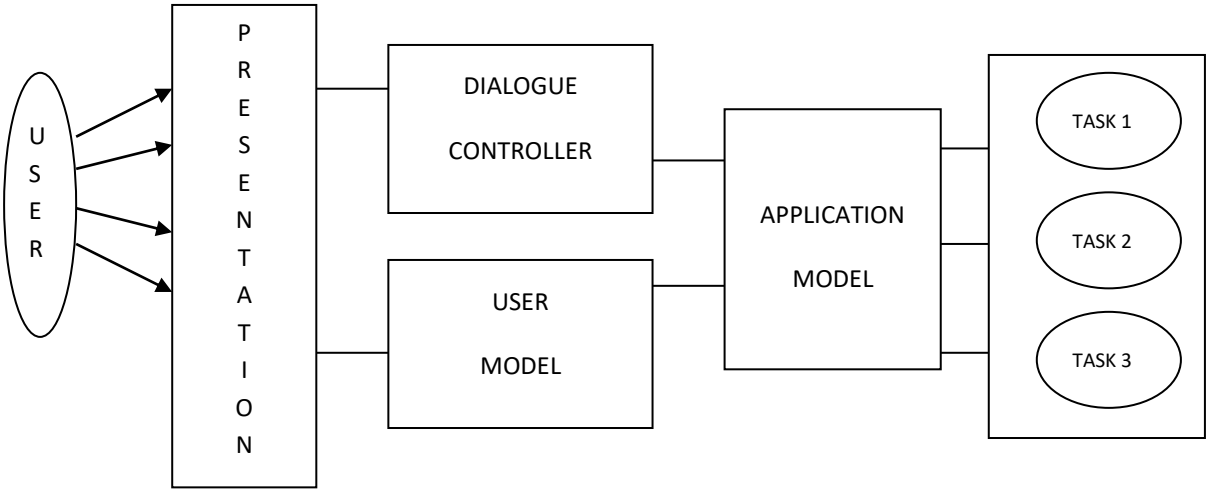

Fig. 1: User Interface Management System

b2) specification formalisms

These techniques have been applied to many aspects of software development [11], permitting the description of external behavior of the system precisely, without having the need to specify its internal implementation. However, despite the fact that the user-interface has been recognized as a critical element of DSS and ES and the need for using formal specification techniques is increasing, such techniques have only rarely been applied to the specification of user-interfaces [10, 23].

SPECIFICATION FORMALISMS

It is advantageous to be able to specify user-interfaces dialogue in an executable specification language; such languages have been based on one of two models: Backus-Naur Form (BNF) [28] and state-transition networks [29]. Each of these models provides a syntax for describing legal streams of user inputs.

The Backus-Naur Form is a well known syntax-directed specification technique which has been used in formalizing programming languages. One general problem that arises with BNF-based techniques is that it is difficult to determine exactly when something will occur. This makes it difficult to specify interactive dialogues. State-transition networks have long been used for dialogue specification. After the early work of Newman (1968), network models have been developed by Parnas (1969, 1971) and Deyert (1977).

The majority of research projects for specifying human-computer interfaces have been concerned with static rather interactive languages. In a static language, an entire text of the input language is present before any processing begins and all the outputs are then produced together in an interactive language. The input can be described as a series of brief texts, where the processing of each input depends on previous inputs. So, in static languages the previous inputs have little if at all effect on the input text, but in interactive languages a specification language must capture not only the systems actions and outputs, but also their sequences with respect to portions of the input.

Network formalism is easily understood and amenable to analysis by using graph theory tools, like path algebras. However, in order to be used for dialogues specifications it has to be modified, to describe – in addition to user inputs – system's actions and their sequence with respect to the input. Each transition is associated with an action and whenever the transition occurs, the system performs the associated action.

Network formalisms have been used in such systems as SYNICS [15], STAG [30], CONNECT [31] and in various Computer-Aided Instructions (C.A.I.) systems [32]. With these projects new features were introduced, among them the most remarkable are the state transition networks hierarchies; these were introduced by Conway and Woods and instead of labeling a state transition arc with a single input token, the transition may be labeled with a term which, in turn, is represented in a separate state – transition diagram. This makes it possible to divide complex diagrams into more manageable pieces.

As an example, illustrating the use of network formalism, we implement in the following diagram the state-transition of a LOGIN command. The notation follows widely used

conventions. Each state is represented by a circle. Each transition between two states is depicted as a directed arc. It is labeled with the name of an input token, plus a footnote containing Boolean conditions, system responses and actions. A given state transition will occur if the input token is received and the conditions are satisfied; when the transition occurs, the system displays the response and performs the action.

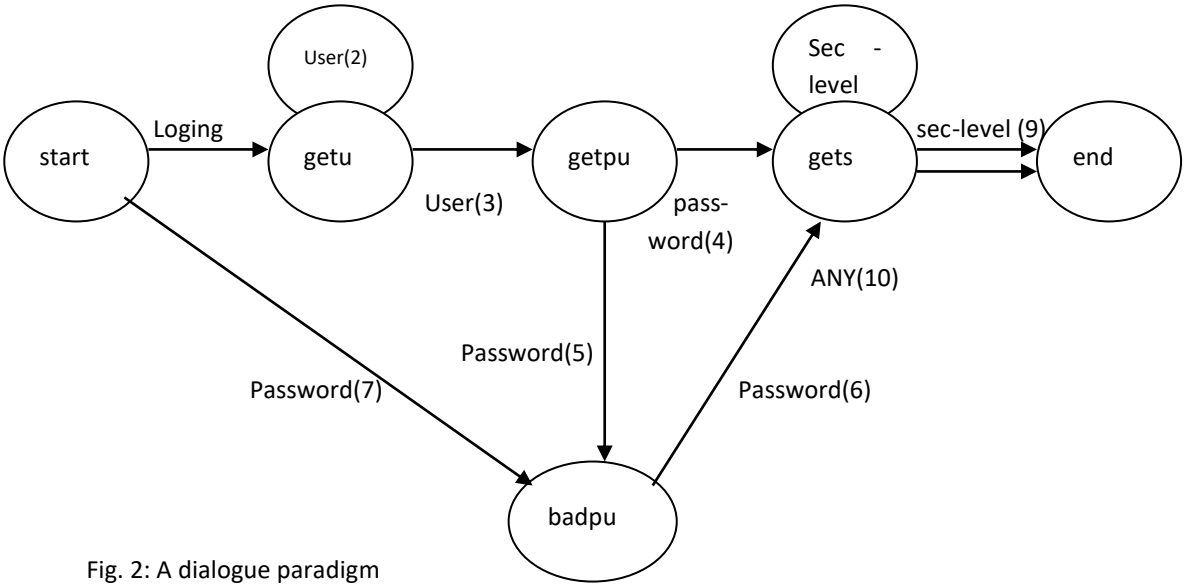

Fig. 2: A dialogue paradigm

The above specification can be represented in text form, which often is more convenient for computer input and output than the graphical diagrams, as follows

S1: INP response "Hello" ⟶ S2

denoting a transition from state S1 to state S2, which expects input token INP and displays response "Hello". Thus, the above network specification, can be written in text form, representation consisting of a list of the transitions that comprise the diagram.

THE SYSTEM

In our case, we adopted the network formalism for representation of the various cognitive actions of the manager during his interaction with the program; however, we augmented the adaptability of the network, by permitting changes in its topology in order to simulate – in addition to user inputs – the system's actions and their sequence with respect to the inputs.

The theoretical tool for analysing the various networks, which are generated, as a result of user's actions, has been borrowed from graph theory and is path algebra [38, 39]. These algebras were first introduced by Carre (1971) and then were applied to a number of path problems. Their value is that they provide a technique for handling the global properties of networks [40] and have been suggested by Alty [41, 42] as powerful tools in assisting interactive-dialogues user-interfaces.

A path algebra is defined as a set P equipped with two binary operators called "dot" and "join" and denoted by "." and ". These operators have the following properties:

the join V operation is idempotent, communitative and associative

xvx= for all xєP

xvy=yvx for all x,yєP

(xvy)vz=xv(yvz) for all x,y,zєP

the dot . operation is associative and distributive over v

(x.y).z=x.(y.z) for all x,y,zєP

x.(yvz)=(x.y)v(x.z) for all x,y,zєP

(yvz).x=(y.x)v(z.x)

the set P contains a zero element Ø for all xєP

and a unit element e such that

e.x=x=x.e for all xєP

The physical significance, when these algebras will be applied to a dialogue network, is that the set P is the set of possible labels with which the arcs can be labelled. We call this the label language for the network of interest. The join operator (v) tells us how to replace the labels on two arcs connecting the same two nodes by a single label on one arch between the nodes. The dot (.) operator tells us how to replace the labels on two sequential arcs by a single label on a single arc. All the above properties are physically reasonable when applied to dialogue networks.

Carre has described [38] a set of eight path algebras (P1-P8), some of which are particularly useful in dialogue analysis, because, they provide the designer of user interfaces with some key aspects of the networks. Furthermore, because the choice of label and operators is up to the designer this approach gives a very powerful method for analysing networks.

Adaptability of networks topology has been simulated by using the Artificial Intelligence Knowledge Representation formalism, known as “Production Systems”. These systems [9, 34, 35] consist of a set of production rules that modify an existing database, a database, the selection of appropriate rules and the resolution of conflicts that may arise when two or more productions are applicable at the same time. Every rule has the form:

IF <condition> THEN <action>

which, of course can take more sophisticated forms, like:

IF <condition_1> THEN <action_1>

IF <condition_2> THEN <action_2>

IF <condition_N> THEN <action_N>

A typical rule in our system can have a form, like:

IF <at node_13>
AND
<command=…>
AND
<variable=0>
……
……
THEN <enable switch at node_6 to network C>

An overview of the system is shown below, in Fig. 3.

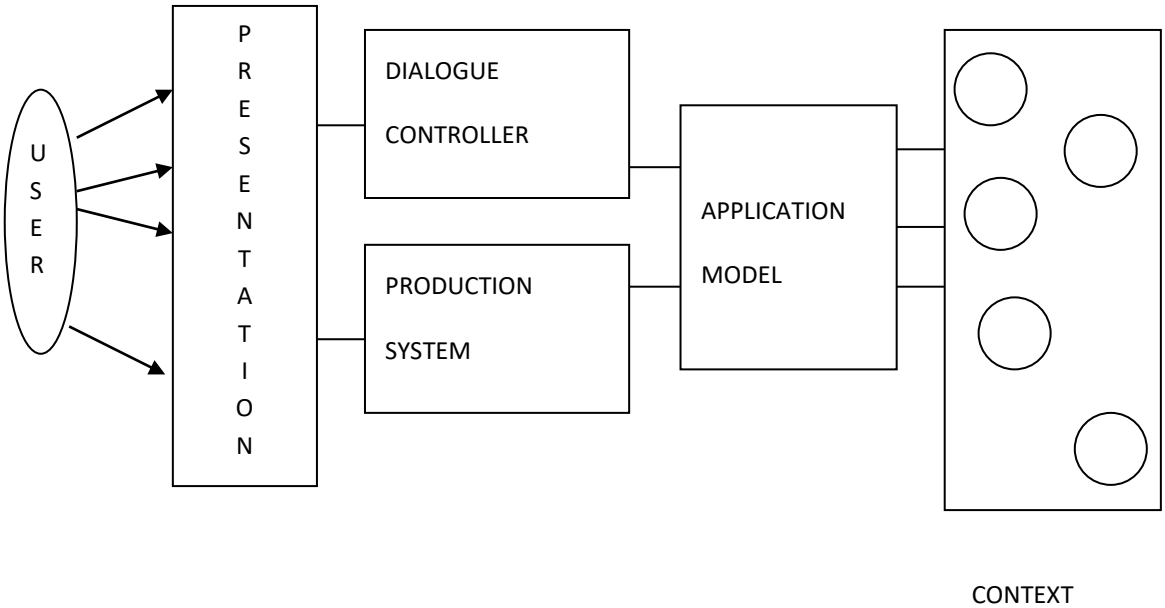

Fig. 3: An overview of the system

The system now can generate the various forms (networks) of interaction between the user/manager and the program, but, now can change their topologies with the aid of the Production System. Of course, these changes can not be ad hoc or random, but are guided by the application model, which represents the expert's view of the various tasks in the context of Investment Modeling and advised all the other components.

The program (application model) operates on the premise that Investing is not difficult if you have some money to start with, if you are willing to take a longer term view and if you avoid constant trading in and out of properties.

The dynamic behind the program's decisions is that two important cycles dominate the business scene. One is succession of expansions and recessions, which is normally called business cycle. The other cycle is marked by predictable changes in interest rates that the Federal Reserve System usually makes in response to the business cycle. This investment strategy maintains that certain sales and purchases almost always are appropriate at

various points in the business cycle, so that the only thing difficult is deciding what phase the cycle is in now.

In the following rules, other appropriate strategies are incorporated, together with some factor of probability:

IF short term interest rates peaked recently,

THEN predict peak (ct. 0.4)

IF the market has been rising for months

AND

The public mood is overwhelmingly positive,

THEN predict peak (ct. 0.6)

IF the trend in business loan demand is lower

THEN predict rising phase (ct. 0.6)

IF the IPI ratio is greater than one

AND

The IPI ratio is increasing

THEN predict rising phase (ct. 0.6)

IF short term rates higher than long term rates

THEN predict falling phase (ct.0.7)

IF predict bottom out

AND

you know a stock fund that does well in rising markets

THEN buy stock mutual funds (ct. 0.8)

IF predict falling phase

AND

Predict peak

THEN stay in T-bills and silver, if you have them (ct. 0.8)

IF predict peak

THEN buy silver certificates (ct. 0.9)

IF the dollar has fallen greatly vs. foreign currencies

AND

The dollar's fall is continuing

THEN buy stock mutual funds (ct. 0.5)

CONCLUSIONS

The need of the current generation of DSS/ES to embody a "cognitive" factors viewpoint, reflecting thus manager's view of the task domain, in terms of the goals he may accomplish, had a major impact on the design of the user-interface systems.

In this work we presented a user-interface for DSS/ES in Investment Modeling. By focusing the application context in one particular (Investment Advising) domain, the building of application models (knowledge-bases) would be easier, provided that an expert will be available for knowledge elicitation. These application models – expressing expert's view of the application – can advice all other modules of the system and simplify the interface design.

The various acts of the user/manager during his interaction with the system have been modeled by network diagrams and the changes of the network topology – as a result of user experience – have been performed by using Production Systems, a well known Knowledge Representation formalism. The theoretical tool for analysis of various networks, were Carre's path algebras, whose properties give us great help for overall manipulation of the networks.

The complexity of this graph-based approach is significantly increased when the system is required to be open, allowing continuous acceptance of various tasks in the application

domain. However, they allow the construction of better formal specifications for various networks representing interactions, facilitating thus the verification of the system.

Future directions involve, a) the implementation of various user-defined algebras, and b) the Knowledge Acquisition for implementing other application models from an expert, in various modeling tasks [28, 29], with the help of Knowledge Engineers and the use of inductive tools.

REFERENCES

Allen, R.B., (1982), "Cognitive factors in human interaction with computers", in Badre, A. and Shneiderman, B. (eds), *Directions in Human Computer Interaction*, Ablex, Norwood, N. J., p.p. 1-26.

Ball, E., and Hayes, P., (1980), "Representation of task specific knowledge in a gracefully interacting User Interface", in Proc. A.A.A.I., p.p. 116-120.

Bateman, R.F., (1983), "A translator to encourage user-modifiable man-machine dialogue," in Sime, M.E. and Coombs, M.J. (eds), *Designing for Human Computer Communication*, Academic Press, London.

Alty, J.L. and Brooks, A., (1985), "Micro technology and user friendly systems: the CONNECT dialog executor," *Journal of Microcomputer Applications*, Vol. 8, no. 4, p.p. 333-346.

Carre, B.A., (1971), "An algebra for network routing problems," *Journal of Inst. Maths Applic*. 7, p.p. 273-294.

Backhouse, R.C. and Carre, B.A., (1975), "Regular algebra applied to path finding problems," *Journal of the Institute of Mathematics and its applications*, Vol. 15, no.2, p.p. 161-186.

Alty, J.L., (1984), "Path Algebras: a useful CAI/CAL analysis technique," *Computer Education*, Vol. 8, No 1, p.p. 5-13.

Alty, J.L., (1984), "The application of Path Algebras to Interactive Dialogue Design*," Beh. and Int. Tech*, Vol. 3, No 2, p.p. 119-132.

Carre, B.A., (1979), *Graphs and Networks*, Oxford Mathematics and Computing Science Series, Clarendon Press, Oxford, p.p. 84-93.

Edmonds, E.A., (1982), "The man-computer interface: note on concepts and design," *Int. Journal of Man-Machine Studies*, Vol. 16, p.p. 321-236.

Edmonds, E.A., (1881), "Adaptive man-computer interfaces," in Coombs, M.J. and Alty, J.L. (eds*)., Computing Skills and the User Interface*, Academic Press, London.

Feyock, S., (1977), "Transition diagram-based CAI/Help system," *Int. Journal of Man-Machine Studies*, Vol. 9, p.p. 339-413.

Gehani, N. and McGettrick, A. (eds), (1986), *Software Specification Techniques*, Addison-Wesley, Workingham, ENGLEWOOD CLIFFS, N.J.

Green, M., (1984), "Report on Dialogue Specification Tools," IFIP Working Group.

Genesereth, M.R. and Nilsson, N.J., (1988), *Logical Foundations of Artificial Intelligence*, Morgan Kaufman, Los Altos, CA.

Harmon, P. and King, D., (1998), *Expert Systems: Artificial Intelligence in Business*, John Wiley, New York.

Heitmeyer, C.L., (1981), "An intermediate Command Language (ILS) for the Family of Military Message Systems, *Technical Memorandum*, 7590-450, Naval Research Laboratory, Washington, D.C.

Jacob, R.J.K., (1983), "Using formal specifications in the design of human-computer interfaces," *Communications of the ACM*, Vol. 26, no 4, p.p. 259-264.

Jackson, M.A., (1983), System Development, Prentice-Hall.

Jones, C.B., (1986), *Systematic Software Development using VDM*, Prentice-Hall, ENGLEWOOD, CLIFFS, N.J.

Jerrams-Smith, J., (1985), "SUSI-a Smart User Interface System," in *People and Computers: designing the interface*, Proceedings of the Conference of the British Computer Society, University of East Anglia, Norwich.

Card, S., Moran, T., Newell, A., (1983), *The psychology of human-computer interaction*, Hillsdale, N.J.: Lawrence Erlbaum.

Hix, D., Hartson, H. (1993), *Developing user interfaces. Ensuring usability through products and process,* John Wiley & Sons, New York.

Shneiderman, B., (1998), *Designing the user interface*, 3rd edition, Reading, M.A., Addison-Wesley.

Nielsen, J., (1999)" User Interface Directions for the Web", *Communications of the ACM*, Vol. 42 (1), p.p. 65-72.

Partridge, D.,(1997), *Knowledge based Information Systems,* Mc Graw–Hill Book Company Europe, England.

Avison D.E., Fitzerald G., (1996), *Information Systems development: Methodologies, Techniques and Tools,* Mc Graw–Hill Book Company, England.

Maedche, A. and V. Zacharias, (2003), "Ontology learning for the semantic web", IEEE Intelligent Systems, Vol 16(2): p.p.72-79.

Karagiannis Stefanos-Kopanakis I.(2005)., Applied Research in Knowledge Management for Tourism Services, International Center for Research and Studies in Tourism –Aix en Provence France: Ser. C. Gestion Management, Vol.9, p.p. 10