2nd Advances in Hospitality and Tourism Marketing & Management Conference
ISBN: 978-960-287-139-3

A GRAPH-THEORETIC STRUCTURAL MODELLING OF COMPLEX TOURISM SYSTEMS

Panagiotis K. Notopoulos
Department of Accounting,
Technological Institute of Serres
Serres, Greece
e-mail: notisp@teiser.gr

Stamatis C. Kontsas
Department of Business Administration
Democritus University of Thrace
Komotini, Greece
e-mail: kontsas@otenet.gr

ABSTRACT

An extension of the interface/task separation architecture in implementing user-interfaces for intelligent Information Systems in a business context is presented, which focus heavily on application modelling of a complex tourism system and represents the structure of the problem domain by relating the geometric properties to certain structural properties of these systems, known as *structural modelling* (Harary, 1983).

The problem domain is depicted as a signed graph and both graphical and numeric representations are applying. Then, the adjacency matrix and matrix powering on it is used to determine paths of various lengths between variables (path algebras**)** for understanding the fundamental interrelations among the variables, to identify basic processes and pinpoint possible strategies to analyze further.

KeyWords: Social networks, structural modelling, complex tourism systems.

INTRODUCTION

Managers are faced with decision-making tasks which require sophisticated Management Information Systems (M.I.S) augmented with ´intelligent´ Decision-support Systems. Although the application of these tools can be of enormous help to the decision-making process, the decision still rests upon the manager and consequently their success is heavily people-dependent. The user-interface component is responsible for managing the dialogue between managers and application programs by providing the specialized function necessary to handle the interaction. In today MIS/DSS , 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 (Ball, 1980; Paparrizos et all 1996; Notopoulos, 2009) and although this interactional approach is ideal when the manager knows the logical structure of his intended tasks, 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.

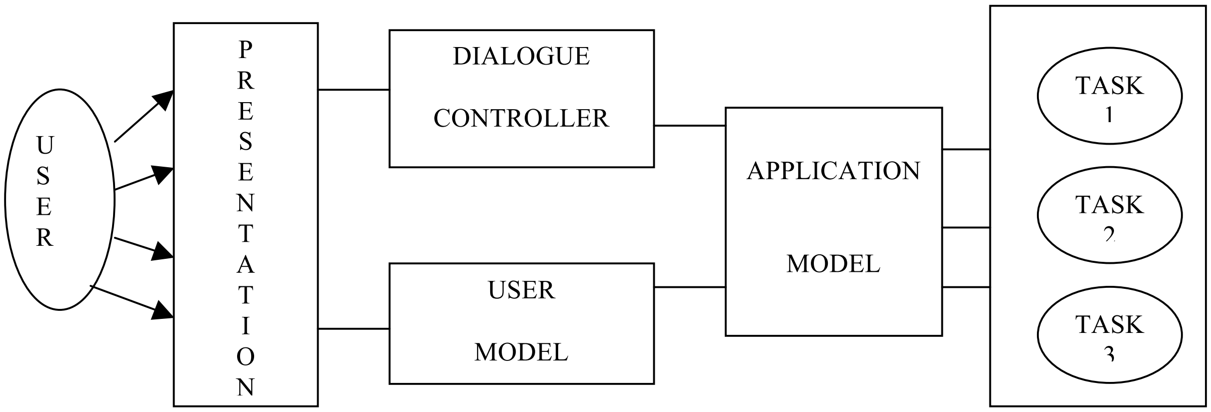
This has as implication the need for adopting a cognitive viewpoint (Allen, 1982; Marcus, A. et all, 1991) by incorporating semantic (Berners–Lee, T. et all, 2001) features of the manager's view of the task domain, which in turn, requires a shift in the design of user interfaces in order to match more naturally with the way the manager conceives and thinks of the tasks he has to achieve, reflecting his view of the task domain in terms of the goals he wants to accomplish. This can be accomplished by providing, separation capabilities of the particular task to be performed, from the user-interface dialogue management systems. .

A) Interface Management and Design

Traditional methodologies for designing and specifying M.I.S. have been concentrated upon the decomposition of the application domain into more manageable sub-problems, i.e. Jackson Structured Design Jackson, 1983), S.S.A.D.M (Avison, et all 1996), Vienna Development Method (Jones, 1986) thus focusing upon implementation issues without addressing usability issues. However, the consideration of applications interfaces requirements results in a different approach to structural decomposition, one that favors the separation of interface aspects from the computational aspects of the application as a desired objective (Card et all, 1983) 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 proposes (Edmonds, 1981, 1982) that the interactions between a user and an 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) and a description of the application. This general paradigm of task-dialogue separation has been further refined by Jerrams (Jerrams-Smith,1985). who proposed the representation of user-model as a distinct module in the interface processor and by Green (Green, 1984) who incorporated the application description into what he termed application model, which contains the user's view of the semantics of the application. These suggestions are shown below.

Figure 1:
User Interface Management System



The last point (application model) has caused conflicting views between researchers, as whether or not it should incorporate aspects of user-modeling and at the present little work exists on application modeling (Ball et all,1980; Probert, 1989; Bergeron, 2003). 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 techniques have been applied to many aspects of software development (Gehani, et all (eds), 1986), 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 MIS and the need for using formal specification techniques is increasing, such techniques have only rarely been applied to the specification of user-interfaces (Tsouros, 1994; Paparrizos et all, 1996; Notopoulos, 2009)

B) Specification Formalisms

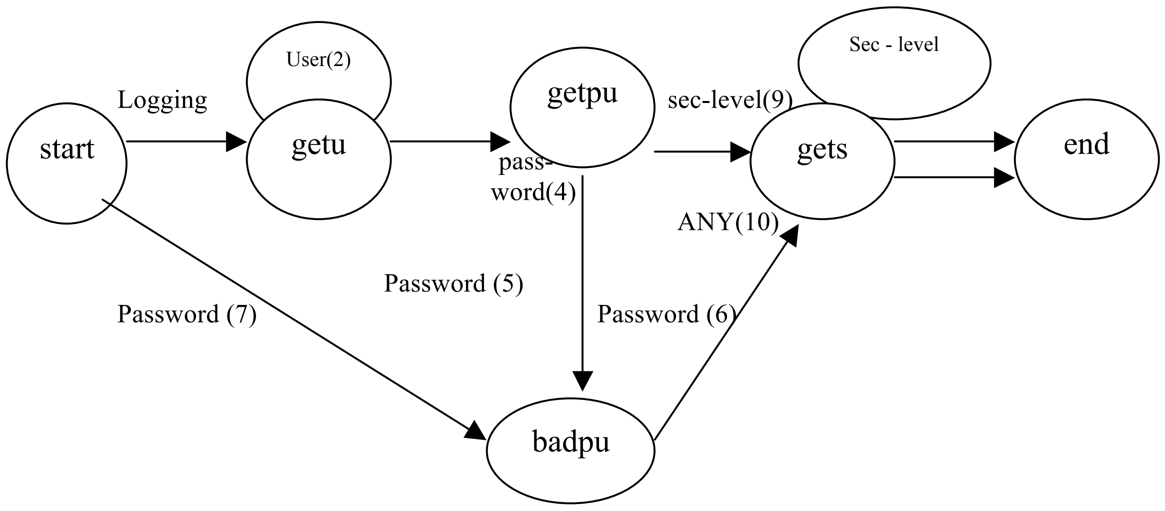
It is advantageous to be able to specify user-interfaces dialogue in an executable specification language; such languages have been based on one of two models: Backus-Naur Form (BNF) and state-transition networks. The Backus-Naur Form is a well known syntax-directed specification technique which has been used in formalizing programming languages but it is difficult to determine exactly when something will occur and this makes it difficult to specify interactive dialogues. State-transition networks have long been used for dialogue specification. After the early work (Newman, 1968), network models have been developed (Parnas 1969, 1971; Deyert ,1977) and network formalisms have been used in such systems as SYNICS, STAG (Jerrams-Smith, 1985), CONNECT (Alty, J.L. et all, 1985) and in various C.A.I. systems (Alty, 1984).

Network formalism is easily understood and amenable to analysis; 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,

ISBN: 978-960-287-139-3

the system performs the associated action. As an example, illustrating the use of network formalism, we implement in the following diagram the state-transition of a LOGIN command in a M.I.S. The notation follows widely used conventions; each state is represented by a circle and 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.

Figure 2:
A dialogue paradigm



C. **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 analyzing the various networks, which are generated, as a result of user's actions, has been borrowed from Graph-Theory and is Path Algebra (Backhouse 1975, Carre 1979). These algebras were first introduced by Carre (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 and have been suggested by Alty (Alty, 1984a, 1984b) as powerful tools in assisting dialogue 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, communicative and associative

xvx=x	for all xϵP
xvy=yvx	for all x,yϵP
(xvy)vz=xv(yvz)	for all x,y,zϵP

- the dot **.** operation is associative and distributive over v

(x.y).z=x.(y.z)	for all x,y,zϵP
x.(yvz)=(x.y)v(x.z)	for all x,y,zϵP
(yvz).x=(y.x)v(z.x)	

- the set P contains a zero element Ø — for all xϵP
  and a unit element e such that e.x = x = x.e — for all xϵP

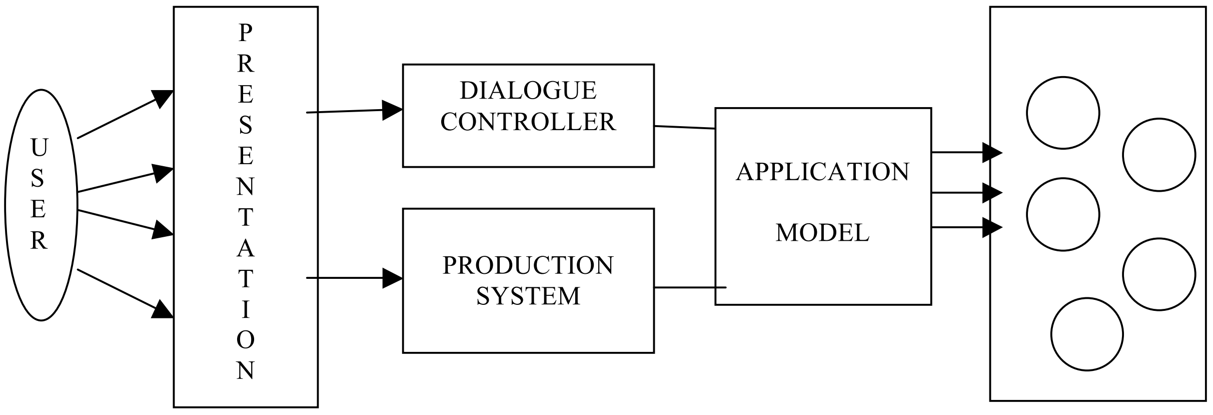
ISBN: 978-960-287-139-3

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 labeled. 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 as can be seen by the following 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 dialogue-networks**.**

	P	xvy	x•y	∅	e
P1:	$\{0,1\}$	max(x,y)	min{x,y}	0	1
P2:	$R\cup\{\infty\}$	min(x,y)	x+y	∞	0
P3:	$R\cup\{-\infty\}$	max(x,y)	x+y	-∞	0
P4:	$\{x\in R \mid 0\leq x\leq 1\}$	max(x,y)	xxy	0	1
P5:	$\{x\in R \mid x\geq 0\}\cup\{\infty\}$	max(x,y)	min{x,y}	0	∞
P6:	$\wp(Ó^*)$	$x\cup y$	$\{x\bullet\psi \mid x\in X, \psi\in Y\}$	∅	^
P7:	$\wp(S)$	$x\cup y$	$\{x\bullet\psi\in S \mid x\in X, \psi\in Y\}$	∅	^
P8:	B	$b(x\cup\psi)$	$\{x\bullet\psi \mid x\in X, \psi\in Y\}$	∅	^

Adaptability of networks topology has been simulated by using the Knowledge Representation formalism, of Production Systems. These systems (Ginsberg, 1993, Vlahavas et all, 2010) 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 the 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 in the following figure.

Figure 3:
An overview of the system



2nd Advances in Hospitality and Tourism Marketing & Management Conference
ISBN: 978-960-287-139-3

ISBN: 978-960-287-139-3

The system 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' module. 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 Tourism Systems and advised all the other components.

The program (application model) operates on the premise that Investing in Tourism 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)

D. Conclusions

The need of the current generation of Intelligent Systems 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 in Tourism Information Systems. By focusing the application context in one particular 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 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. The theoretical tool for analysis of various dialogue networks, were Carre's Path Algebras, whose properties give us great help for overall manipulation of these and allow the construction of better formal specifications for various tourism networks representing interactions, facilitating thus the verification of the system.

REFERENCES

1. Allen, R.B. (1982). Cognitive factors in human interaction with computers. in Badre, A. and Shneiderman, B. (eds), *Directions in Human Computer Interaction*, (pp. 1-26). Ablex, Norwood, N.
2. J., Alty, J.L. and Brooks, A. (1985) Micro technology and user friendly systems: the CONNECT dialog executor. *Journal of Microcomputer Applications*. 8(4): 333-346.
3. Alty, J.L. (1984). Path Algebras: a useful CAI/CAL analysis technique. *Computer Education*. 8(1): 5-13.

4. Alty, J.L., (1984). The application of Path Algebras to Interactive Dialogue Design. *Beh. and Int. Tech*. 3(2):119-132.
5. Avison D.E. and Fitzerald G. (1996). *Information Systems development: Methodologies, Techniques and Tools.,* Mc Graw–Hill Book Company, England.
6. Backhouse, R.C. and Carre, B.A. (1975). Regular algebra applied to path finding problems. *Journal of the Institute of Mathematics and its applications*. 15(2):161-186.
7. Ball, E. and Hayes, P. (1980). Representation of task specific knowledge in a gracefully interacting User Interface. in Proc. A.A.A.I.: 116-120.
8. 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*. London: Academic Press.
9. Berners–Lee, T. and Hendler, J. (2001). Publishing on the Semantic Web. Nature: 410(1023–1024).
10. Bergeron B. (2003). *Essentials of Knowledge Management*. John Wiley & Sons.
11. Card, S., Moran, T., Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, N.J.: Lawrence Erlbaum.
12. Carre, B.A. (1971). An algebra for network routing problems. *Journal of Inst. Maths Applic*. 7: 273-294.
13. Carre, B.A. (1979). *Graphs and Networks*. Oxford Mathematics and Computing Science Series. Oxford: Clarendon Press.
14. Edmonds, E.A. (1982). The man-computer interface: note on concepts and design. *Int. Journal of Man-Machine Studies*. 16:321-236.
15. Edmonds, E.A. (1981). Adaptive man-computer interfaces. in Coombs, M.J. and Alty, J.L. (eds*).,* *Computing Skills and the User Interface*. London: Academic Press.
16. Feyock, S. (1977). Transition diagram-based CAI/Help system. *Int. Journal of Man-Machine Studies*. 9: 339-413.
17. Gehani, N. and McGettrick, A. (eds), (1986). *Software Specification Techniques*. Addison-Wesley, Workingham, ENGLEWOOD CLIFFS, N.J.
18. Ginsberg, M. (1993). *Essentials of Artificial Intelligence*, Morgan Kaufmann.
19. Green, M., (1984). Report on Dialogue Specification Tools. IFIP Working Group.
20. Genesereth, M.R. and Nilsson, N.J.. (1988). *Logical Foundations of Artificial Intelligence*. Morgan Kaufman, Los Altos, CA.
21. Harmon, P. and King, D. (1998). *Expert Systems: Artificial Intelligence in Business.* New York: John Wiley.
22. 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.
23. Hix, D., Hartson, H. (1993). *Developing user interfaces. Ensuring usability through products and process.,* New York: John Wiley & Sons.
24. Jacob, R.J.K. (1983). Using formal specifications in the design of human-computer interfaces. *Communications of the ACM*, Vol. 26(4): 259-264.
25. Jackson, M.A., (1983), System Development, Prentice-Hall.
26. Jones, C.B. (1986). *Systematic Software Development using VDM*. Prentice-Hall, ENGLEWOOD, CLIFFS, N.J.
27. 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.
28. Maedche, A. and Zacharias, V. (2003). Ontology learning for the semantic web. IEEE Intelligent Systems. Vol 16(2):72-79.
29. Marcus, A. and van Oam, A. (1991). User interface developments for the nineties. IEEE Computing. 24(9).
30. Nielsen, J., (1999). User Interface Directions for the Web. *Communications of the ACM*, Vol. 42 (1): 65-72.
31. Partridge, D. (1997). *Knowledge based Information Systems,* Mc Graw–Hill Book Company Europe, England.
32. Shneiderman, B. (1998). *Designing the user interface*. 3rd edition, Reading: M.A: Addison-Wesley.
33. Witt C.A. and Muhleman, A.P.(2004). The Implementation of Total Quality Management in Tourism. *Vol 15 (6), pp416-424*

ISBN: 978-960-287-139-3