

Rule-based behaviour modelling: specification and validation of information systems dynamics

A Tsalgatidou and P Loucopoulos*

The accurate capture and representation of user requirements plays a critical role in the construction of effective and flexible information systems. One important component of a requirements specification is the definition of the behaviour of the system, i.e., those volatile concepts of the application domain that determine the dynamics of the information system. Failure to model and validate these volatile concepts results in inflexible systems that become increasingly unreliable to future changes. One way of overcoming the problem of inappropriate specification paradigms is to adopt models that focus on those procedures of the application domain which dictate organizational policy. To this end, the paper introduces the Rule-based Behaviour Model (RBM) as a means of specifying knowledge pertinent to the behaviour of a system. The paper also shows how this model can be used in carrying out graphical animation of the specification, using a Petri-net-based model.

information systems, requirements engineering, behaviour modelling, rule-based specification, validation, animation

It is generally accepted that the development of information systems (ISs) requires the use of models, techniques, and computer-assisted tools within a methodological framework to achieve the system's intended functionality.

Much has been written about the benefits and problems of contemporary approaches to software development¹⁻³. Layzell and Loucopoulos⁴ argue that a major shortcoming of these approaches is in their treatment of modelling those aspects of a system most frequent to change. Failure to do this results in inflexible systems, which become increasingly unreliable to future changes. Experience with the use of contemporary development approaches has highlighted the importance of explicitly addressing the issues of IS evolution⁵⁻⁸.

This paper argues that there is a need for a development approach that identifies areas of potential change and offers explicit representation of the volatile elements representing these changes. One way of examining these

volatile elements is in terms of organizational policy, as it is changes to this policy which are the primary cause of modifications that have to be made to the systems. ISs, being models of reality, require change only because corresponding changes occur in the domain that these systems operate. Therefore, there is a need for a clear separation between the goals of a system from the functions that realise these goals. If such a separation is not achieved then any organizational policy, which inevitably determines changes to the IS, is specified and maintained only in programming code. The implication of this is that any changes in ISs require considerable effort with unpredictable results, due to the difficulty in locating and altering the appropriate code that represents the policy to be changed.

This paper argues that the dynamic component of an IS should be specifically identified and modelled in a rule-based format to reflect explicitly strategic and operational policy. This approach provides a direct and natural way of identifying organizational policy and mapping this onto structures conducive to computation. The view advocated in this paper, and supported by others^{9,10}, is that business policy must be explicitly specified in system development and identifiably maintained throughout the development process and subsequent evolution, thus permitting an immediate and flexible response to changes in user requirements. As Sölvberg argues: '... if we cannot find tools which ensure consistency across baselines, we shall probably always have to be dependent on the maintenance programmers for evaluating proposals for organizational changes, because they will continue to be the only ones that can evaluate the impact of proposed changes on the organization's software'¹¹.

To this end, the paper introduces a model, the Rule-based Behaviour Model (RBM), and a development approach for the specification of the dynamic component of ISs. This model and its graphical counterpart are described. A key characteristic of the requirements specification area is the need to validate captured knowledge about the application domain. On the basis of the RBM, it is shown that validation of knowledge pertinent to the behaviour of a system can be carried out in the form of graphical animation of the specification. The paper uses

Department of Informatics, University of Athens, Panepistimiopolis, TVPA Buildings, 15771 Athens, Greece

*Department of Computation, UMIST, PO Box 88, Manchester M60 1QD, UK

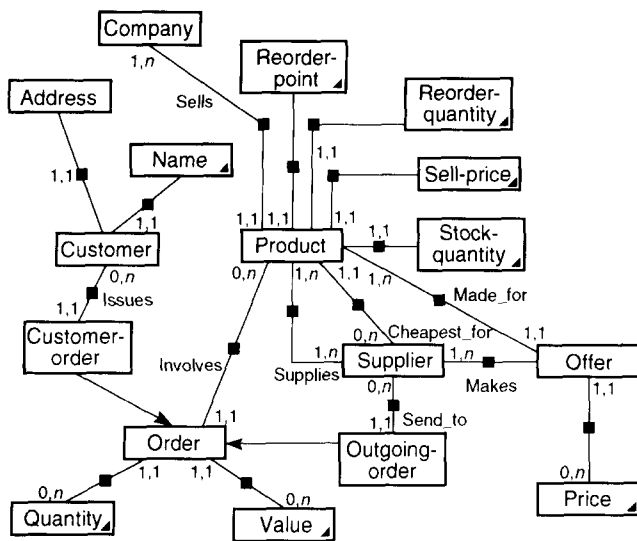


Figure 1. E-R diagram for wholesaler case study

an extensive range of examples to demonstrate the various concepts. These examples are based on the case study described in the Appendix.

MODEL FOR SPECIFICATION OF ISs

The basic concepts of the RBM, discussed in this section, fall into two broad categories, object modelling and action modelling.

Object modelling is concerned with the base and derived objects of interest to the domain of investigation. This is achieved using an extended binary entity-relationship (E-R) model^{12,13} and a set of constraints that apply on the components of the E-R schema.

Action modelling is concerned with the invocation and coordination of actions that apply on the objects of the E-R schema. This is specified in terms of signals, procedures, and rules. From a methodological perspective, it is assumed that the specification of the dynamic part of a system refers to objects (entity and relationship classes) found in a previously constructed E-R model.

Object modelling

The object model is presented in terms of an extended E-R schema that shows entity classes, relationship classes, subclasses, and value classes. An example of an E-R schema for the case study of the Appendix is shown in Figure 1.

An entity class (e.g., PRODUCT, SUPPLIER) is an aggregate of entities. A value class (e.g., NAME, PRICE) is an aggregate of values. A relationship class is an aggregate of relationships of a certain type. Relationships can be of two types: one type associates entities to values (e.g., PRODUCT has SELL_PRICE) and the other is confined to associating entities (e.g., SUPPLIER supplies PRODUCT). For each relationship, the model recognises 'sentence predicates', which make statements (e.g., 'a PRODUCT is sold at SELL_PRICE'), and 'referent functions', which are a selection mechanism for

entities or values (e.g., 'a PRODUCT sold at a SELL_PRICE').

Integrity constraints on the E-R model, or derivations on it, are represented by static rules. Integrity constraint rules define conditions that must hold between different entities or relationships. Derivation rules define entity classes, relationship classes, and value classes in terms of other entity, relationship, or value classes. An example of an integrity constraint might be 'a PRODUCT of type A should be supplied by SUPPLIER who supplies PRODUCT of type B'. An example of a derivation rule would be 'SUPPLIER is cheapest for a PRODUCT if SUPPLIER makes OFFER for the PRODUCT and the PRICE of that OFFER is the minimum of all PRICES of OFFER made for that PRODUCT'.

Action modelling

The functionality of an IS is defined by a set of actions. Actions, to be executed, have to be triggered by some agents, which may be generated by two different sources:

- externally to the IS, i.e., from its environment
- internally to the IS as a result of some changes of state in the system itself

For example, consider a stock control system. A happening in the environment, such as the arrival of new stock for a product, affects the IS by triggering the actions that perform the updating of the stock quantity of the received product. Obviously, not all the happenings in the real world affect an IS. The system responds only to certain happenings in the real world — those defined by the designer of the system.

The second aspect of action triggering is concerned with the effect that some actions executed in one part of an IS have on another part. For example, referring again to a stock control system, the result of reducing the stock quantity of a product could invoke other actions that take care of the reordering of low stock products.

Following this discussion, it is possible to view the evolutionary aspects of an IS as a collection of actions that are executed whenever an appropriate signal is received. Therefore, a 'signal' is defined as a Boolean expression which when evaluated to true invokes a set of actions that change the state of the IS. A signal may be generated in the environment of the IS (a business event or a clock condition) or within the system itself. The former type will be called external signal and the latter internal signal.

The detailed specification of actions is accommodated using the concept of a procedure. The proposed model follows an object-oriented approach in that a procedure is always attached to an entity. A 'procedure' represents a set of actions that an entity suffers or that are required of another entity.

The third component of the model is the rule that serves as the means of specifying the coordination of the execution of all procedures. A 'rule' is the means of specifying the coordination of signals and procedures,

i.e., it defines the conditions necessary for the invocation of a procedure. A rule has the syntax:

```
WHEN <signal>
[IF <precondition>]
THEN <procedure> | <signal>.
```

The **WHEN** part of a rule is compulsory. It contains the signal that can trigger the rule, in other words it initiates the consideration of a dynamic rule. The **IF** part of a rule is optional. This part contains preconditions that are evaluated when the appropriate signal is present and if found to be true, the **THEN** part of the rule is executed.

A precondition in this model may be either a state operation or a state condition (both types being expressed in terms of the E-R model). An example of a state operation might be before-modifying-the-product-price; an example of a state condition might be quantity in stock < quantity reorder point.

A small set of rules for two of the entity classes of the case study in the Appendix follows.

Rules for entity class COMPANY

```
R1: WHEN SIGNAL end-of-month
    THEN for each X in product_list, SIGNAL < check_
        stock -> X >
R2: WHEN SIGNAL end-of-week
    THEN PROCEDURE produce_stock_list
R3: WHEN SIGNAL cust_order_arrived (CUST_ORDER)
    IF ordered_PRODUCT is in product_list
    THEN SIGNAL < process cust_order (CUST_
        ORDER) > PRODUCT >
```

Rules for entity class PRODUCT

```
R4: WHEN SIGNAL check_stock
    IF stock_quantity <= reorder_point
    THEN SIGNAL < reorder -> SELF >
R5: WHEN SIGNAL reorder
    IF SUPPLIER supplies cheapest product
    THEN PROCEDURE order_PRODUCT from_SUP-
        PLIER
R6: WHEN SIGNAL process_cust_order (CUST_ORDER)
    IF stock_quantity <= reorder_point
    THEN PROCEDURE put_order_on_hold (CUST_
        ORDER)
R7: WHEN SIGNAL process_cust_order (CUST_ORDER)
    IF stock_quantity > reorder_point
    THEN SIGNAL < sell_product (CUST_ORDER) >
        SELF >
R8: WHEN SIGNAL process_orders_on_hold
    IF orders_on_hold ≠ 0
    THEN for each X in orders_on_hold,
        SIGNAL < process_cust_order (X) -> SELF >,
        delete X from orders_on_hold
R9: WHEN SIGNAL sell_product (CUST_ORDER)
    IF stock_quantity > CUST_ORDER.quantity
    THEN PROCEDURE sell_quantity = CUST_
        ORDER.quantity to CUSTOMER
R10: WHEN SIGNAL sell_product (CUST_ORDER)
    IF stock_quantity <= CUST_ORDER.quantity
    THEN PROCEDURE sell_quantity = stock_quantity to
        CUSTOMER
```

Examples of external signals are end_of_month, end_of_week, and cust_order_arrived (CUST_ORDER) in rules R1, R2, and R3. The internal signals check_stock,

reorder, and process_cust_order (CUST_ORDER) in rules R4, R5, and R6 have been generated by the action part of the rules R1, R4, and R3, respectively. Examples of procedures can be found on the action part of R2, R5, R6, R9, and R10. Examples of the generation of internal signals can be found on the action part of rules R1, R3, R4, R7, and R8. As an example of the use of a rule without the presence of an **IF** clause consider rule R1. In this case the action part will be executed every time the signal end_of_month is present, that is, at the end of each month the stock level of every product will be checked.

The above rules specify the business policy of the case study. For example, the policy requirement 1 of part A2 of the Appendix is specified by rules R1 and R2, while the policy requirements 2 and 3 of part A2 of the Appendix are specified by rules R5 and R2, respectively.

GRAPHICAL NOTATION OF ACTION MODEL

Because of the potentially high volume of rules and the complexity of their interdependency, a specification expressed in terms of the constructs described previously should be viewed at a high level of abstraction. It is obvious that a textual representation of rules lacks adequate abstraction facilities. Therefore, a graphical notation that permits the modelling of the interaction of these rules is a desirable property of the proposed model. If in addition this notation exhibits formal properties then the specification would lend itself to animation, which can be exploited for highlighting the behaviour of the system to potential users.

This requirement leads to the consideration of the Petri-net formalism¹⁴ and its many extensions. A Petri net is an abstract, formal model of information flow. It is a directed graph containing two types of nodes: places, represented by circles, and transitions, represented by bars. These two types of nodes are connected by directed arcs. A place is an input or an output to a transition and vice versa. If the set of places in a Petri net is P, the set of transitions is T, the input function (i.e., a function returning the set of places connected as input to each transition) is I, and the output function (i.e., a function returning the sets of places connected as output to each transition) is O, then a Petri net is formally defined as $C = (P, T, I, O)$. Places can hold tokens. A Petri net with tokens is a marked Petri-net. A marking of a Petri net is an assignment of tokens to places in that net. The vector $\mu = (\mu_1, \mu_2, \dots, \mu_n)$ gives, for each place in the Petri net, the number of tokens for that place. Thus a Petri net $C = (P, T, I, O)$ with a marking μ becomes a marked Petri net, $M = (P, T, I, O, \mu)$.

The formalism proposed here for augmenting the RBM uses places to represent signals (the **WHEN** part of rules), while transitions are inscribed with the **IF** and **THEN** parts. The graphical RBM (GRBM) is based on the augmented Petri-net model studied by Zisman¹⁵ and is formally defined as:

$$\text{GRBM} = (P, S, T, A, M, R, R')$$

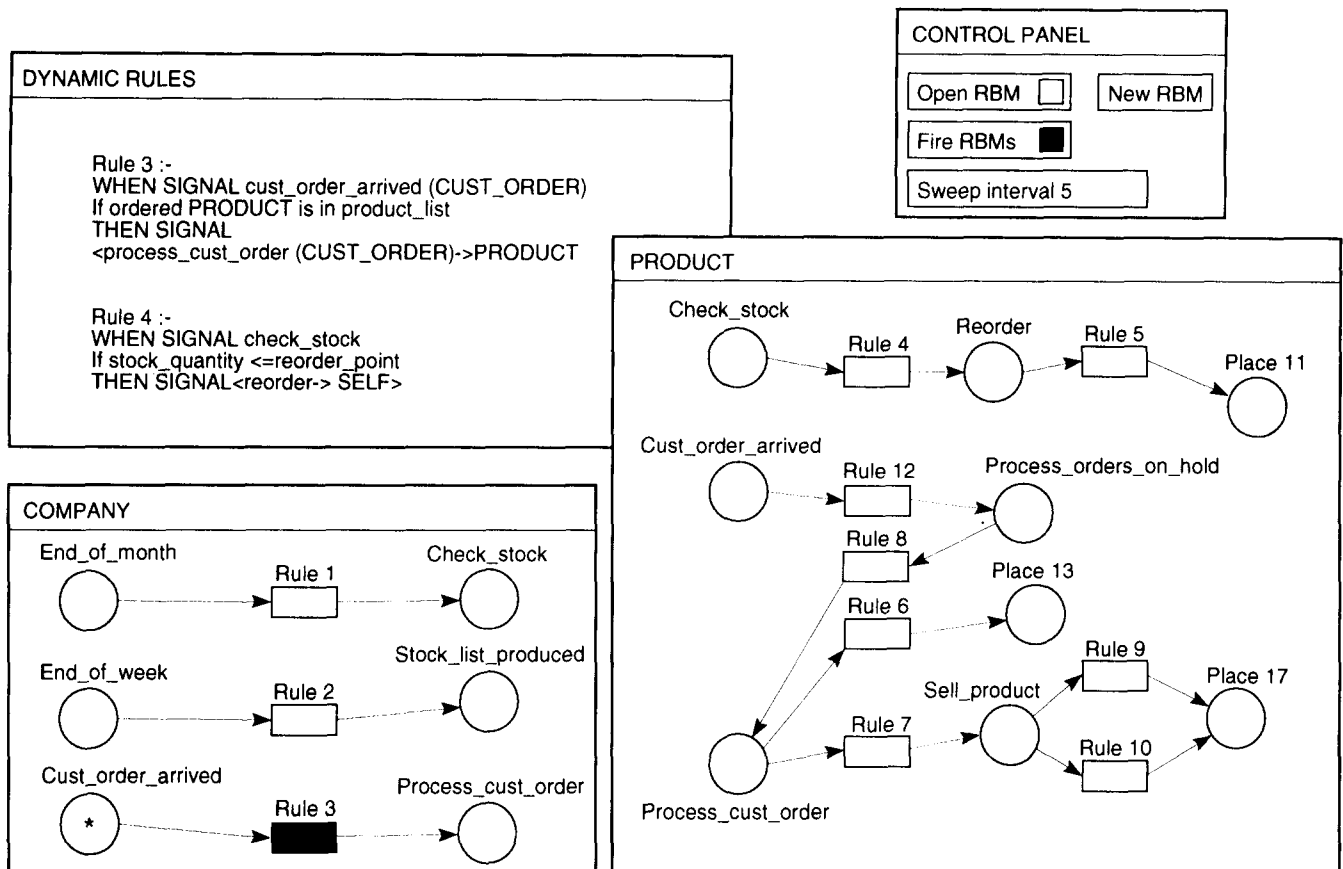


Figure 2. GRBMs for entity classes *PRODUCT* and *COMPANY*

where P is the set of places, S the set of signals, T the set of transitions, A the set of arcs connecting places to transitions, M_0 the initial marking of the net, R the set of dynamic rules, and R' the set of enabled dynamic rules.

As every rule needs a signal to be triggered, all the transitions will have at least one input place representing the triggering signal. If the actions in the **THEN** part of the rule include the generation of some signals, the transition will have output places corresponding to the generated signals. In cases where no signals are generated (e.g., rules R2 and R5 earlier), an output place may be assigned to the corresponding transition to conform to the Petri-net theory. This may also help for validation purposes as the presence of a token in such an output place denotes the termination of the execution of actions assigned to the corresponding transition.

The GRBM of a system consists of a number of small nets, each one corresponding to one entity class demonstrating its dynamic behaviour. The behaviour of a system as a whole is demonstrated by all these small nets, which communicate via signals. Figure 2 displays the GRBMs for the entity classes *COMPANY* and *PRODUCT* of the case study of the Appendix, while the textual representation of the corresponding rules is as given earlier. In Figure 2, these two nets communicate via the signals 'check_stock' and 'process_cust_order (CUST_ORDER)'. The obvious advantage of this approach is a natural decomposition of what could potentially be a large and complicated model, thus solving the problem of increasing complexity in modelling

large and complicated systems. In the chosen approach each set of rules remains manageable and understandable.

EDITING AND ANIMATING GRBM

In recent years a number of techniques, methods, and computer-assisted tools have been developed to provide help in the specification phase and generally in the process of IS development. Experience with these tools has shown that, independent of how much benefit these tools may promise, they will only become accepted if they provide 'natural' means of exploitation of their facilities with the minimum of training effort on the analysts' part¹⁶. Tools that seem to satisfy this requirement are those which deal with animation aspects.

Animation is a useful means that enables the visualization of the inner working of a process in action. It has been employed in programming for designing, developing, and debugging programs or monitoring their performance, and also for helping a programmer to understand or to remember the functionality of a program that has not been used for a long time¹⁷. Examples of the use of animation in various programming applications have been reported^{17,18}. However, programs and algorithms are not the only artefacts that can be animated. Control flow graphs or dataflow graphs can also be animated by having their nodes highlighted during execution.

The use of animation in the first stages of system development is a recently identified need and there are a

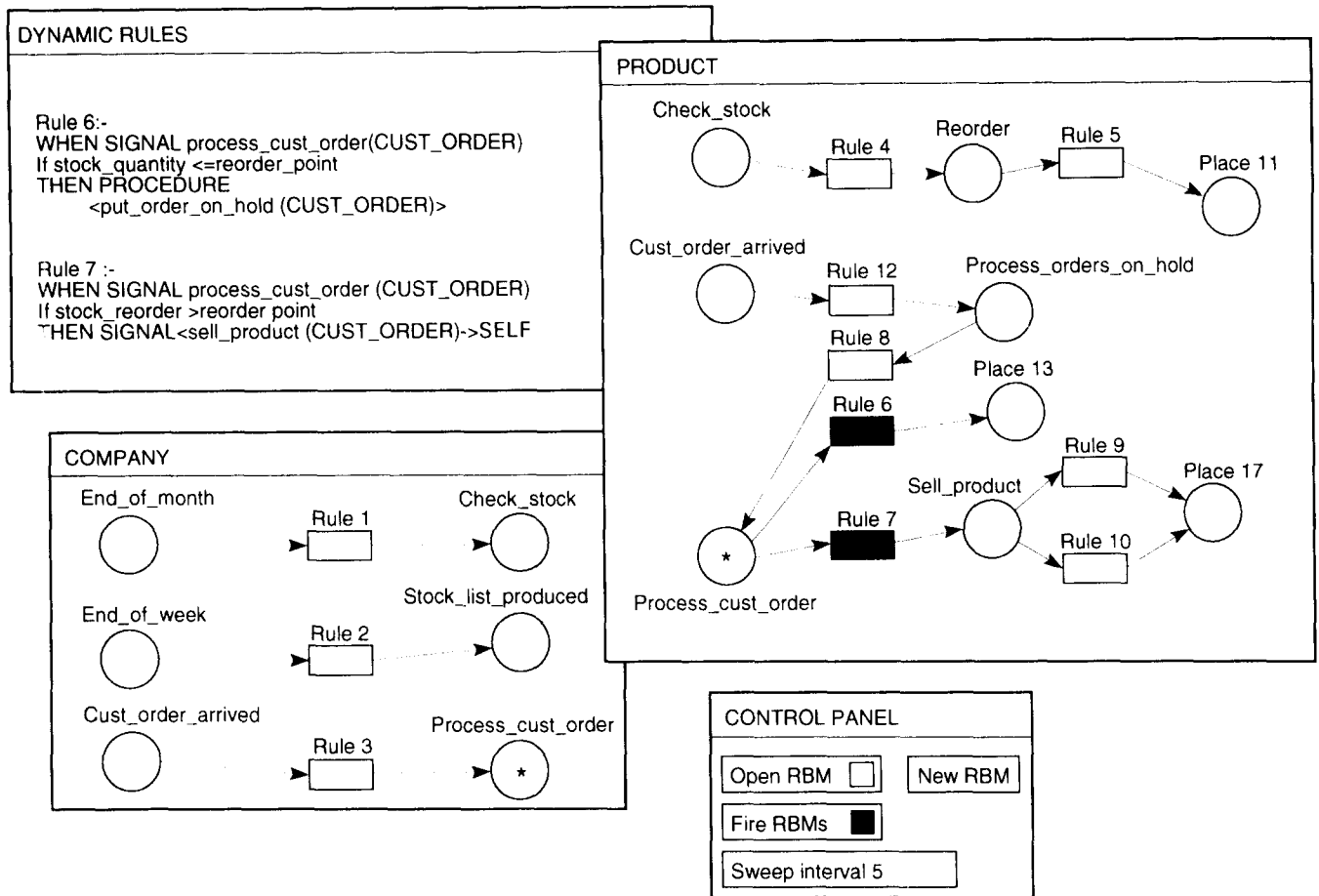


Figure 3. Animation of RBM specification

few examples of the use of animation in requirements engineering¹⁹⁻²¹. The results of the use of animation in system specifications show that it is a promising way of reducing errors in the requirements phase and ensuring that the intended behaviour of a system has been properly captured and modelled. A working model of a system at the requirements stage, especially when it is accompanied by graphics facilities, is more comprehensible to end-users and enables them to interact with an analyst more effectively. The feedback from animating a requirements specification can be used for its refinement, helping in this way the discovery and weeding out of errors that will arise and cause serious problems at later stages of development.

The RBM Petri-net model, through its graphical representation and formal semantics, lends itself to animation. Therefore, graphical animation of the model may be used for demonstrating the dynamic behaviour of the modelled system and for providing help in discovering errors in the rule-based specification, such as redundant rules, conflicting rules, circular rules, or missing rules. A Petri-net editor/ animator has been implemented on a Sun workstation²², using the Flavours object-oriented environment²³ to assist in constructing and animating a GRBM.

The editing part of the tool provides facilities for the creation, modification, deletion, and copying of an entire or parts of a model. In addition to editing facilities for the basic diagrammatic notation of the RBM Petri-net

model, textual rule editing facilities are also provided; the editor can create entries in two files, one for the dynamic rules and the other for the signals (with the constraint that the names of rules and signals must be unique). This facility permits the display of dynamic rules and signals whenever an analyst wishes to examine in more detail a rule or a condition under which a rule may fire. Examples of a dynamic rule window are shown in Figures 2 and 3.

An important feature of the tool is the interactive editing of a specification. A model can be edited while it is being animated. Thus insertion or deletion of places, transitions, rules, and so on is taken into account by the tool in the process of animation, as soon as it happens.

Animation of an individual model can be activated and deactivated through the control panel window. A user can start animation by depositing tokens at various places. When a transition is notified that one of its input places has a token, it checks all its other input places and if they all have tokens, the transition becomes enabled. Enabled transitions are highlighted by the animator to alert the user about the chosen path in the network. If the precondition part of a rule is satisfied, the rule fires and the marking of the net changes accordingly. The firing of the rules may generate other signals, which can subsequently enable other transitions. Thus, in the authors' approach, animation is mainly used for checking the firing of rules and for supporting the interactive design of the rule base.

Consider for example that a customer's order has

arrived and the environment has generated the signal 'cust_order_arrived (CUST_ORDER)'. The place associated with this signal will receive a token and the corresponding rule (rule 2) will be enabled and highlighted by the tool. This can be seen in Figure 2, which demonstrates the GRBMs for the two entity classes PRODUCT and COMPANY which communicate via the signals 'check stock' and 'process_cust_order (CUST_ORDER)'. If the IF part of rule 2 is true, the system will execute the THEN part of rule 2, which contains the generation of the signal 'process_cust_order (CUST_ORDER)'. Any other places representing this signal, no matter to which RBM Petri net they belong, will receive a token.

In Figure 3 the signal 'process_cust_order (CUST_ORDER)' has been generated and the two 'process_cust_order' places (of the PRODUCT and CUSTOMER GRBMs, respectively) hold a token. This means that the IF part of rule 2 was satisfied and its THEN part executed. Rules 6 and 7 have become enabled due to the generation of the previous signal. Thus the marking of the net has changed and the new marking is associated with the new set of enabled transitions. In this way, through the generation of an internal signal, it is possible to model the fact that some happening associated with the entity class COMPANY has triggered rules which define the behaviour of the entity class PRODUCT. The enabled rules R6 and R7 will be considered next for firing, which in turn may generate signals to other rules or terminate the sequence by effecting some change of state in the IS.

The use of animation in the context of the RBM Petri-net model is viewed by the authors as being of assistance to an analyst rather than automatically correcting any detected errors. This is because a characteristic facet of the requirements specification process is the presence of multiple user views about the modelled domain. Any tool employed during this development stage should highlight inconsistencies of specification, while permitting the presence of multiple user views. When the modelled system is small, rule base consistency can be checked by the system analyst by simply observing the rules. For large systems, however, the use of tools based on algorithms for automatic detection of inconsistencies, redundancies, or contradictions in the rule base would be of great help²⁴.

CONCLUSIONS

The emphasis of the work presented in this paper is in the area of requirements engineering and in particular in the specification of those elements of an IS that are volatile in nature. In the context of the dynamic components of a system, the model proposed adopts a rule-based orientation, thus reflecting the business policy that often determines the evolution of a system.

The main benefits of the proposed approach come from its three main characteristics, namely, the rule-based approach, the methodological approach based on

an E-R framework, and the ability of the model to be animated.

- The dynamic behaviour of a system is expressed in a rule-based manner. This makes the model easy to understand and, arguably, closer to the end-user perception of the system's behaviour. The model supports the interactions between a system and its environment and between various parts of a system. In addition, the conditions that must be met before the execution of any actions and the consequences of the execution of these actions are also encoded.
- The dynamic model is derived within an entity-centred methodological framework. In software engineering, adopting a strategy that derives a conceptual schema at the early stages of the development process is well accepted and many E-R formalisms are currently in practice.
- The model is interpreted in a Petri-net notation, which permits its animation. This feature is supported by the editor-animator tool. The 'simulation-type' approach to validate a specification should be regarded as an integral part of the task of analysing a specification.

Therefore, the model accommodates the derivation of a specification of dynamic components in ISs as well as the animation of the specification for validation purposes. Furthermore, the formal basis of the GRBM enables its behaviour to be expressed in algebraic form. It supplies the basis for automatable algorithms measuring properties of the modelled system, such as consistency once a definition of consistency has been given. Experience from other approaches with similar objectives has shown the value of developing a specification on the basis of formal models and in a well defined design discipline^{19,25,26}.

At the same time the use of animation in various kinds of domains has shown the usefulness of using interactive graphics as a communication medium. Animation has been successfully used in algorithm design and development^{17,18,27}, in education²⁸, and in program debugging²⁹. In the context of rule-based systems, the tracing of rules in a rule base and reasoning about reached results are important activities. Most existing rule-based systems have extensive trace and explanation text-based facilities. However, graphics-based facilities provide more help because they can be understood and used by less sophisticated users³⁰.

The graphical animation presented in this paper can clearly be employed not only in the validation of a specification of an IS, but also in systems that contain rules expressed in a forward-chaining representation.

The nature of the implemented tool in its current state requires that the tool is operated by systems analysts rather than non-computing experts. In this respect the developed system must be regarded as an assistant to an analyst. As argued elsewhere, the task of systems analysis is such that the role of end-users is supportive in providing knowledge about the modelled system and in validating this knowledge^{31,32}. Contemporary computer-aided software engineering tools and research in this area rec-

ognise this aspect, and this is reflected in state-of-the-art products and research prototypes^{33 35}. Possibly, future advances in the area of knowledge-based-systems might result in tools that would involve end-users more directly in the process of requirements analysis, but it is debatable whether the participation of expert systems analysts in the task of requirements specification will be completely eliminated. In the meantime, there is a need to develop new paradigms that more accurately model an application domain, provide features for the validation of the model, and ultimately enable the maintenance of an IS in terms of its specification rather than its programming code.

The Rule-based Behaviour Model presented in this paper is a contribution to the current debate and research for paradigms that more closely reflect the activities in requirements elicitation, specification, and validation. The rule formalism specifies the conditions under which certain activities may take place. The development of the RBM is based on the premise that such a formalism expresses business policies in understandable terms and can result in an executable specification that can be used for prototyping purposes.

ACKNOWLEDGEMENTS

The work reported in this paper has been partly funded by the Commission of the European Communities under the Esprit R&D Programme and the Greek PTT.

REFERENCES

- 1 Maddison, R *Information system methodologies* Wiley-Heyden (1983)
- 2 Olle, T W *et al. (eds) CRISI — information system design methodologies: a comparative review* North-Holland (1983)
- 3 Olle, T W *et al. (eds) CRIS3 — improving the practice* North-Holland (1986)
- 4 Layzell, P J and Loucopoulos, P 'A rule based approach to the construction and evolution of business information systems' in *IEEE Conf. Proc. Software Maintenance* Phoenix, AZ, USA (24–27 October 1988) pp 258–264
- 5 Talbot, D and Witty, R W *Alvey programme software engineering strategy* Alvey Directorate, London, UK (1983)
- 6 Lehman, M 'Programs, life cycles and laws of software evolution' *Proc. IEEE* Vol 68 No 9 (September 1980) pp 1060–1076
- 7 Brown, P 'Why does software die?' in Wallis, P *Life-cycle management* (State of the Art Report) Vol 8 No 7 (1980) pp 31–45
- 8 Yue, K 'What does it mean to say that a specification is complete?' in *Proc. 4th Int. Workshop Software Specification and Design* Monterey, CA, USA (1987) pp 42–49
- 9 Fjeldstad, R K *et al.* 'Application program maintenance' in Parikh and Zvegintzov, N (eds) *Tutorial on software maintenance* IEEE (1983) pp 13–27
- 10 Mathur, R N 'Methodology for business system development' *IEEE Trans. Soft. Eng.* Vol 13 No 5 (May 1987) pp 593–601
- 11 Solvberg, A 'Introduction to the workshop session on methods and models for evolutionary information systems' in *Proc. IFIP TC 8 Working Conf. Evolutionary Information Systems* Budapest, Hungary (1–3 September 1981) North-Holland (1981)
- 12 Chen, P P 'The entity-relationship model — towards a unified view of data', *ACM Trans. Database Syst.* Vol 1 No 1 (1976) pp 9–36
- 13 Nijssen, G M, Duke, D J and Twine, S M 'The entity-relationship data model considered harmful' in *Proc. 6th Symp. Empirical Foundations of Information and Software Sciences* Atlanta, GA, USA (October 1988)
- 14 Petri, C A 'Communication with automata' *Supplement 1 to Technical report RAD C-TR-65-337, Vol 1* Griffiss Air Force Base, NY, USA (1966) translated from 'Kommunikation mit Automaten' University of Bonn, Germany (1962)
- 15 Zisman, M D 'A representation of office processes' *WP 76-1-03* Dept of Decision Sciences, University of Pennsylvania, PA, USA (1976)
- 16 Redwine, S T and Riddle, W E 'Software technology maturation' in *Proc. 8th Int. Conf. Software Engineering* (August 1985) pp 189–200
- 17 London, R L and Duisberg, R A 'Animating programs using Smalltalk' *Computer* Vol 18 No 8 (August 1985) pp 61–71
- 18 Brown, M H and Sedgewick, R 'Techniques for algorithm animation' *IEEE Software* (January 1985) pp 28–38
- 19 Dahler, J, Gerber, P, Gisiger, H-P and Kundig, A 'A graphical tool for the design and prototyping of distributed systems' *Soft. Eng. Notes* Vol 12 No 3 (July 1987) pp 25–36
- 20 Diaz-Gonzalez, J P and Urban, J E 'ENVISAGER: a visual object-oriented specification environment for real-time systems' in *Proc. 4th Int. Workshop on Software Specification and Design* London, UK (1987) pp 13–20
- 21 Shand, J, Adhami, E and McNeile, A 'An environment for the execution and graphical animation of JSD specifications' in *Proc. Workshop Knowledge-Based Systems in Software Engineering* UMIST, Manchester, UK (March 1988)
- 22 Tsalgaidou, A 'Dynamics of information systems: modelling and verification' *PhD thesis* UMIST, Manchester, UK (June 1988)
- 23 Rubinstein, M *Flavours implementation and users manual in Poplog environment* University of Sussex, UK (1986)
- 24 Giordana, A and Saitta, L 'Modelling production rules by means of predicate transition nets' *Inf. Sci.* Vol 35 (1985) pp 1–41
- 25 Kung, C H and Solvberg, A 'Activity modelling and behaviour modelling' in Olle, T W *et al. (eds) CRIS3 — improving the practice* North-Holland (1986) pp 145–172
- 26 Hanssen, H C and Lund-Hanssen, H 'Executable specifications based on Kung and Solvbergs behaviour net model' *Technical report No 24/87* University of Trondheim, Norwegian Institute of Technology, Trondheim, Norway (10 April 1987)
- 27 Duisberg, R 'Animated graphical interfaces using temporal constraints' in Mantei, M and Orbeton, P (eds) *Human Factors in Computing Systems, CHI '86 Conf. Proc.* Boston, MA, USA (1986) pp 131–136
- 28 Brown, M H and Sedgewick, R 'A system for algorithm animation' *Comput. Graph.* Vol 18 No 3 (July 1984) pp 177–186
- 29 Myers, B A 'Visual programming, programming by example and program visualisation: a taxonomy' in Mantei, M and Orbeton, P (eds) *Human Factors in Computing Systems, CHI '86 Conf. Proc.* Boston, MA, USA (1986) pp 59–66
- 30 Lewis, J W and Lynch, F S 'An effective graphics user interface for rules and inference mechanisms' in *Proc. Technical Conf. Theory and Practice of KBS* Surrey, UK (1982)
- 31 Loucopoulos, P and Champion, R E M 'Knowledge-based support for requirements engineering' *Inf. Soft. Technol.* Vol 31 No 3 (April 1989) pp 123–135
- 32 Loucopoulos, P and Karakostas, V 'Validating conceptual models of office information systems' *Soft. Eng. J.* Vol 4 No 2 (March 1989) pp 87–94
- 33 MacDonald, I 'Information engineering — an improved, automatable methodology for designing data sharing systems' in Olle, T W *et al. CRIS3 — Improving the practice* North-Holland (1986) pp 173–224
- 34 Rich, C, Waters, R C and Reubenstein, H B 'Toward a

requirements apprentice' in *Proc. 4th Int. Workshop Software Specification and Design* Monterey, CA, USA (1987)

- 35 Peitri, F *et al.* 'ASPIIS: a knowledge-based environment for software development' in *ESPRIT '87: achievements and impact* North Holland (1987) pp 375–391

APPENDIX: DESCRIPTION OF WHOLESALE COMPANY

Company B & S is a wholesale firm that runs its business in a simple and straightforward way.

A1: general domain description

- (1) From the supplier and customer organization, the name, address, and contact person are known.
- (2) For a list of products, there is information about the sales price, quantity of stock in hand, reorder point, reorder quantity, and suppliers that can supply the product and at what price.
- (3) An order to a supplier (outgoing orders) consists of a list of order lines, each specifying the ordered products and quantity (price not mentioned).
- (4) Shipments from suppliers are received that match a set of order lines of the same supplier.
- (5) An order from a customer (incoming order) consists of a list of order lines, each specifying the ordered products and quantity.
- (6) A shipment to a customer matches a set of order lines of that customer.

A2: Business policies, decisions, and rules

At the time of implementation, the following policies were established:

- (1) At the end of the month, check the stock level of each product and if it is below the reorder point, order a quantity equal to the reorder quantity.
- (2) Buy the products from the supplier who sells it cheapest.
- (3) Every week a stock list must be produced.
- (4) Prices can only change overnight.

After problems appeared because product A, bought from supplier X, was not completely compatible with product B, bought from supplier Y, the managing director decided that:

- (5) Product B can only be bought from suppliers that also sell product A and vice versa.

A year later the company was selling a lot of A-, B-, and C-type products, so that the company ran out of stock often for these and only these products. As sales had unpredictable peak sales, the managing director decided that:

- (6) For only products A, B, and C, now reorder whenever the reorder point is reached and do not wait until the end of month.
- (7) Whenever the stock is below the reorder point for these products (A,B,C only), good customers will have their order immediately processed. The other orders are put on hold until new stock arrives. Then, first come, first served applies.