MODELLING AND ANIMATING INFORMATION SYSTEMS DYNAMICS

Aphrodite TSALGATIDOU

01 PLIROFORIKI, Makrommatia 15, Athens 104 34, Greece

Communicated by M. Paul
Received 28 February 1990
Revised 11 June 1990

Keywords: Software engineering, information systems, requirements engineering, dynamic modelling, object-oriented rule-based development, Petri nets, animation

1. Introduction

The argument put forward in this paper is that the explicit modelling of application domain policy is one of the most important factors for the development of flexible information systems, i.e., systems that can react to changes of the environment. Based on this premise, this paper focusses on the modelling of the dynamic concepts (change-related) of an information system by means of a rule-based model, within an object-oriented framework. A Petri net based model augmented with signals and rules, called modified Petri net (MPN), is used to capture and express interactions of information systems objects. The validation of requirements specifications is supported by exploiting the potential of the selected model to be graphically animated.

This approach provides a means of maintaining specifications at a high level of abstraction, separating the goals of a system from the functions that realise these goals. If this separation is not achieved, then changes which reflect evolution of an information system have to be performed only at the code level, something which is very cumbersome due to the difficulty of locating the code which has to be altered. This is the case with most contemporary approaches, since the elements which determine changes in a system are specified only in programming code.

2. Modelling information systems dynamic aspects

The framework proposed here for capturing and organising information systems aspects is based on the object-oriented philosophy. Object-oriented is an approach that has gained a lot of advocates in the programming field (e.g. [2,7]) and has lately started gaining advocates in system development [1]. This approach to system development realises the modelling of a system in terms of objects and their relationships.

In the approach proposed in this paper, the capturing and definition of the static properties and the relationships between the objects identified in an information system, is achieved by employing an entity-relationship modelling technique enriched with inheritance. In the early modelling phases a hierarchy of object classes is derived where subclasses inherit the properties and relationships of the superclasses. The following will emphasize on how the behaviour of an information system can be specified after the static properties of its object classes have been defined.

An information system is considered to be consisting of a set of object classes with certain dynamic behaviour. The definition of an information system as a whole is specified by the definition of the dynamic behaviour of each object class. The definition of the behaviour of an object class is achieved by using the concept of behaviour unit.
A BU describes a function of a system or a particular situation where the object of interest plays an important part. A BU is activated by the presence of a signal and its behaviour is expressed as a set of dynamic rules.

The rationale which resulted in the use of the concepts of BUs, signals and dynamic rules, views an information system as an artifact with certain functionality and behaviour. This functionality can be described by a set of dynamic rules while the exhibited behaviour is the result of the execution of these rules. Dynamic rules are triggered by signals. The concept of a BU is used to group the rules triggered by the same signal, while BUs themselves are grouped according to the object class they belong to. Signals are external or internal messages generated either by a happening in the environment (e.g. the signal cust_order_arrived is generated by the occurrence of the corresponding happening) or by the execution of appropriate dynamic rules in the information system (e.g. the signal process_cust_order is generated as the result of the execution of a rule in the same information system). Signals are essential for the communication and the activation of the BUs.

The dynamic rules and the corresponding BUs which refer to a class provide a way to partition the behaviour of an object class. An object that suffers actions or acts upon other objects, may be associated with one or more BUs. For example, the operations of an object PRODUCT, in a stock control system, could be: check_stock, reorder, sell_product, etc. However, BUs are associated only with one object class. As it happens with static properties, the subclasses of a class may also inherit the dynamic behaviour of that class. A subclass inherits all the BUs of the parent class. Furthermore, a subclass may specify additional BUs which describe its behaviour in situations that the parent class does not react. A subclass may also specialise the behaviour of the parent class by altering the definition of an inherited BU. In this case, the functionality of the specialising BU overrides its generalised equivalent.

The syntax of a BU has as follows:

\[
\text{<dynamicrulelist>} = \text{<dynamicrulename>}
\]

\[
\text{<dynamicrulename> = 'WHEN' <signal> \{ '<param>', '<param>', \ldots \}}
\]

The syntax of a dynamic rule is:

\[
\text{<dynamicrule>} = \text{<dynamicrulename>} \text{<action>}
\]

A BU describes the response of the system to a specific signal and the definitions of the triggering signal is given in the WHEN part of that BU. Therefore, the WHEN part of a BU is compulsory because the presence of a signal is necessary for the activation of the BU. When a signal is present, the dynamic rules of the BU that matches this signal are activated. The associated rules with a BU have an IF and a THEN part. The IF part is optional whereas the THEN part is compulsory. In the case of the presence of the IF part in a rule, all the preconditions of that rule must be satisfied before the actions defined in the THEN part are executed. The preconditions of a rule may be expressions involving object attributes or time. The actions in the THEN part of a rule may be processes or primitive operations like create, modify and delete or they may generate internal signals.

When modelling the dynamics of an information system within the presented framework, the BUs of its object classes and their triggering signals have to be defined. The dynamic behaviour of each BU is expressed in a rule-based format. The next action is to construct a modified Petri net per object class.

3. The MPN model: A formalism for modelling information systems dynamic behaviour

The formalism proposed here is the modified Petri net (MPN). An MPN is a Petri net [8], where places represent signals (the WHEN part of BUs) and transitions are inscribed with dynamic rules. As every rule needs to be triggered by a signal, all the transitions will have at least one input place representing the triggering signal. If the actions of
the THEN part of a rule generate some signals, the transition will have output places corresponding to the generated signals. In cases where no signals are generated, an output place may be assigned to the corresponding transition so as to conform to the Petri net theory. This may also help for validation purposes.

The MPN model of a system consists of a number of small MPNs, each one corresponding to one object class, demonstrating its dynamic behaviour. The behaviour of a system as a whole is demonstrated by all these small MPNs which communicate via signals. This approach solves the problem of increasing complexity in modelling large and complicated systems which is the case with simple Petri nets.

The graphical nature of the MPN model enables the viewing of the rules interdependency. Different levels of abstraction may be explicitly defined in different MPN views, providing a solution to the problem of the diagram complexity which results from the high volume of the dynamic rules and the interdependencies between them.

The mathematical model of the MPN has been based on the augmented Petri net model studied by Zisman [11] and is presented below.

\[ \text{MPN} = \langle P, S, T, A, M_0, BU, ABU \rangle \]

where

- \( P \) is a modified Petri net,
- \( S \) is the set of places,
- \( S \) is the set of signals,
- \( P \supset S \),
- \( T \) is the set of transitions,
- \( A \) is the set of arcs connecting places and transitions,
- \( [P \times T] \cup [T \times P] \supset A \),
- \( M_0 \) is the initial marking of the net,
- \( BU \) is the set of the behaviour units,
- \( [T \times P \times A] \supset BU \),
- \( ABU \) is the set of active BUs.

The formal basis of the MPN model 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 of the rule base, once a definition of consistency has been given.

4. Animating the MPN model

Animation is a useful means which enables the visualisation of the inner working of a process in action. The use of animation in the first stages of system development is a recently identified need and the work in [3,4,9] are among the very few examples that can be found in this area. The results of the use of animation in system specifications show that it is a very promising way of reducing the errors in the requirements phase and ensuring that the intended behaviour of a system has been properly captured and modelled. The gained feedback from animating a specification can be used for the refinement of the requirements specification as it helps to discover and weed out errors that will arise and cause serious problems at later stages of the development.

The potential of the MPN model to be graphically animated was one of the main reasons that it was selected for the representation of the dynamics of information systems. A computer-assisted tool (the Petri net editor-animator) has been implemented on a Sun workstation to support this task [10]. A very important feature of the implemented tool is that it exploits the interpretable nature of the MPN model. New definitions become immediately active and the effect of any changes performed in the model become immediately observable. In our approach, animation is mainly used for checking the firing of rules and for supporting the interactive design of the rule base. The above is illustrated in Fig. 1, where the definitions of two MPNs corresponding to the object classes COMPANY and PRODUCT can be seen. The two MPNs communicate via the signals \( \langle \text{check\_stock} \rangle \) and \( \langle \text{process\_cust\_order} \rangle \). In the same picture the \( \langle \text{process\_cust\_order} \rangle \) signal has been created as the result of the execution of rule 3 and a token has been put at the two \( \langle \text{process\_cust\_order} \rangle \) places (one in the COMPANY MPN and the other in the PRODUCT
MPN). At this stage, the three transitions inscribed with rules 6, 7 and 8 become enabled due to the generation of the previous signal.

Apart from the descriptive power of the MPN model (which is quite well demonstrated and exploited using animation), its decision power, i.e., its capacity to prove properties of the modelled system cannot be underestimated. The notion of consistency partly depends on the application at hand, since some constructs in a network of rules can be acceptable in some cases and not in others. Especially at the requirements stage, the accommodation of multiple user views is possible. This requirement may reflect some kind of redundancy or conflicts in the rule base. When the modelled system is small, the verification of the rule base consistency can be performed by the systems analyst. However for large systems the use of tools based on algorithms to automatically detect inconsistencies, redundancies or contradictions in the rule base could be of great help [6]. Thus, the use of animation combined with an algorithmic approach to the analysis of the MPN model can assist the process of validating and verifying an information system model.

5. Conclusions and future work

In this paper it is believed that an important issue in the area of requirements engineering is the explicit modelling of the dynamic aspects of information systems, in a way that enables the demonstration of their dynamic behaviour and facilitates their validation. Dynamic aspects are important in system modelling as these are the aspects that usually change and cause problems in system evolution and maintenance. Therefore, their correct capture and explicit representation is of paramount importance in the construction of information systems which are effective and flexible. Effective systems are those which satisfy the initial
user requirements. Flexible systems are those systems which are able to change whenever any changes in the real world take place and whenever user requirements change.

The use of the MPN formalism for modelling the dynamic rules is understandable to end-users and its potential to be graphically animated can assist in the validation process. A very important feature of the MPN model is that it describes the dynamics of an information system as a set of interpretable rule structures which can be interactively created by the analyst using the implemented tool. In this way, the systems analyst can interactively modify the structure of the rule base, so that the exhibited behaviour of the designed system prototype corresponds to the specifications. Current work and future development is focussed to the automation of algorithms handling the mathematical properties of the MPN model and the incorporation of the above to the existing tool.

It is believed that the successful completion of the work described here has shown that the modelling of the dynamic behaviour of information systems in terms of MPN concepts and the subsequent validation of the MPN model using animation is one step towards the development of effective and flexible information systems.

References