Journal of the Operational Research Society (2000) 51, 1029–1040 ©2000 Operational Research Society Ltd. All rights reserved. 0160-5682/00 $15.00

www.stockton-press.co.uk/jors

Variable structure modelling of dynamic industry systems

K Christodoulou[1]* and K Vlahos[2]

[1]*City University Business School and* [2]*London Business School, London*

This paper argues for and develops a *variable structure* modelling methodology, that is one which treats individual players in an industry as agents and provides the means for agent entry and exit as well as evolution of their behaviour. The resulting simulation framework extends traditional industry dynamics approaches and object oriented simulation, unifies a number of concepts from software engineering, artificial intelligence and knowledge engineering and provides a rich, rule-based specification for describing a wide range of structural changes. The framework allows the evolution of both structure and behaviour over time and is especially suited for modelling dynamic industries and markets that can endogenously change their own structure. We demonstrate the features and power of the modelling platform with a stylised model of competition that extends earlier work in evolutionary economics and allows for firm entry and exit and outline a description of a business application to the telecommunications industry.

Keywords: agent-based simulation; evolutionary modelling; telecommunications

Introduction

Today's markets are highly dynamic and competitive and firms, in order to be successful, need to be able to adapt to the continuous changes and learn from their environment. 'The ability to learn faster than our competitors may be the only sustainable competitive advantage', noted de Geus[1] emphasising the importance of institutional learning. The trend towards enhanced competition, liberalisation and deregulation leads us to consider evolving industry structures with a multiplicity of interacting actors and objectives. Industry simulation considers business strategy models at the firm level and instead of seeking equilibrium solutions, offers insights on possible outcomes of strategic choices that companies may wish to make. Firms need to consider and be prepared for change that transforms in a fundamental way their economic and industrial environment. In the telecommunications industry for example, the rapid growth of mobile phones and the internet, and the convergence of information, media, and telephony services are transforming the industry landscape. Similarly, in the electricity industry, a wave of deregulation and privatisation, followed by vertical disintegration and a large number of cross-border mergers, is facilitating the emergence of a truly global industry.

These examples highlight the need for modelling tools than can help policy makers and management teams model a firm's choices and to explore a number of scenarios of industry evolution. modelling and simulation offers the means to study large complex systems that *evolve over time*.[2–4] Simulation forces the researcher '[...] to be precise about the relationship among entities, to make implicit assumptions explicit, and to describe in detail the mechanisms by which entities and relationships change'.[5]

System dynamics (SD),[6,7] has been successfully used for industry simulation in, for example, electricity[8] biotechnology,[9] telecommunications[10] and media.[11] Ginsberg[12] considers this type of modelling as part of what he calls 'new age' strategic planning. Futher approaches include microsimulation[13–15] that captures industry dynamics by applying a set of transition probabilities to each individual instead of mapping individual preferences, decisions and actions. In connectionist approaches,[16] the role of each individual is defined by the strength of its connections to other agents. The only structural change that is incorporated in these models is that of change in behaviour. One can also consider neural networks as a related framework, where learning consists of the revision of connection strengths between units.

One of the major premises of traditional industry simulation is that the fixed structure of the system under consideration brings about its dynamic behaviour. Once the structure and policy rules have been defined the model will describe the dynamic behaviour of the system, but system components and relations among components will not vary. The model might accurately capture the structure of a complex system, but this will be valid for as long as the current structure holds. The aim of this research is 'to close

*Correspondence: K Christodoulou, Management Systems and Information, City University Business School, Northampton Square, London ECIV 0HB, UK.
E-mail: K.Christodoulou@city.ac.uk

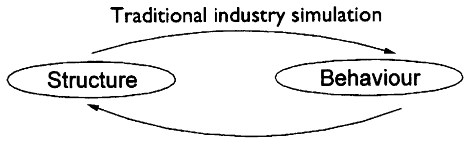


Figure 1 Closing the link: how behaviour can influence structure.

the loop' between behaviour and structure, as shown in Figure 1.

We are interested in exploring the issue of how individual players' behaviour can alter the structure of a system. To do this, we need to employ a modelling methodology that accounts for changing structure, of the type introduced by Oeren[17] and Zeigler and Oeren[18] and called *variable structure modelling*. By structure, these authors refer to the composition of the system and the interaction between its parts. Hence, they deal with models that entail in their description the possibility of changing their own structure. This is slighlty different from the approach employed in evolutionary economics, where the emphasis is on how economic agents who inhabit a fixed-structure system learn and change their *behaviour* over time.

This paper is organised as follows. The next section reviews existing approaches to variable structure simulation and poses a number of research questions. Section three presents a new simulation formalism that supports variable structure modelling in a transparent way. A description of an object-oriented implementation of the framework follows in section four and a test application in section five. Section six outlines a business application to the telecommunications industry and the final section contains concluding remarks.

Existing variable structure simulation approaches

Variable structure simulation is a comparatively new field with relatively few research contributions to the modelling and simulation of systems whose structure changes over time. We classify existing variable structure modelling approaches into system and *individual oriented*, with the former focusing more on agents that control parts of a system and the latter emphasising agent autonomy.

Zeigler's[19] Discrete Event System Specification (DEVS) introduces the notion of atomic models with input and output ports and the hierarchical fashion in which they combine to form coupled models. DEVS describes modular, hierarchical systems, in which behavioural and structural dynamics are exclusively defined at the atomic model level. Zeigler also specified 'non-trivial' variable structure systems and the system oriented approach to structural change.[20] In this approach, the knowledge of the model initiating and the area affected by a structural change are restricted by the boundary of the (sub)system of the initiating model.

The System Entity Structure (SES)[19] is a scheme that represents structural knowledge, that is knowledge about composition, coupling and taxonomy and provides a formal framework for modelling a family of possible structures. Associated with each SES is a model base and by means of a special operation called *pruning*, it is possible to generate specific system structures. SES, however, is a static construct; once a structure has been derived with the pruning operation, it does not change dynamically. A system-oriented approach is based on the idea of endomorphic intelligent agents[21] that describe an entity's awareness of its own place in the system and represent explicit models within models, a notion that Zeigler called 'endomorphy'.

Pawletta *et al*[22,23] introduce *structure events* into both the atomic and coupled model level as a means of bringing about structural change. A structure event is defined as a <*condition*, *action*> construct where the condition and action domains are limited to an event's system scope. This implementation uses the S Structure discussed earlier. *AgedDEVS*[24] employs a special class of atomic models that have an explicit internal model of themselves and their environment. Each of these agents has an internal model of the models it is controlling, that is, those that are affected by structural change. The controlling model decides on and initiates the structural change.

Individual oriented variable structure models focus on the autonomy of system components as self-organising units.[25] In the Extended Modelling System formalism (EMSY)[26] systems are understood as entities that have clear boundaries to their environment. Rules define an entity's behaviour and can change the entity's attributes, composition and environment and play the central role in describing change.

The *MIMOSE* project[27,28] describes a framework where new entities appear as cloned objects of the initiator of creation, an approach employed in the simulation of human, animal or artificial societies.[29] As a result, an existing component hierarchy extends in its width, but not depth.

Variable structure models have been mainly studied for and applied to physical[26] and technical systems, such as adaptive computer architectures.[20,30–35] Variable structure methodology as such has not been greatly exploited for industry simulation. The only exception, perhaps, is *GLIDER*, which is a simulation language that has been specifically introduced for modelling processes of structural change.[36,37] The central issue in this approach becomes the traversal and evaluation of a tree where the nodes are the points of structural change and the arcs emerging from them are the possible structures resulting from the change, a process very similar to a decision analytic approach.

The existing approaches we saw earlier place significant emphasis on the *structural* knowledge that entities have.

However, they do not utilise the storage of generic, behavioural knowledge (a set of procedural methods that define how an entity might behave once it becomes part of a system) and therefore cannot accommodate entry of new types of entities. The limitations of earlier approaches include defining behavioural and structural dynamics exclusively at the atomic model level and restricting structural changes by the boundary of the (sub)system of the initiating model.[19,20] Similarly, using models of models *AgedDEVS*,[24] can create computing overheads. Finally, *MIMOSE*[27,28] only allows cloning of existing agents.

Moreover, most of the approaches discussed in this section originated in the field of computer science and have a very different focus from that of industry simulation, where one considers a highly interconnected network of agents who can influence any other agent in a system.

This research aims to address the limitations of existing frameworks and explore a number of questions related to variable structure modelling:

- Can we provide transparent extensions to existing approaches for industry simulation that allow for evolving model structure?
- Can we incorporate generic behavioural knowledge in agents coupled with learning mechanisms that would drive policy changes?
- How can we best strike a balance between autonomy of agents and external control?
- How do we allow entry of new kinds of agents?
- Can we provide a user friendly, interactive visual simulation platform that would allow managers to build and experiment with models of industry evolution?
- Does variable structure modelling provide additional insights over and above those provided by more 'traditional' approaches?

Variable Structure DEVS (VS/DEVS)

As mentioned earlier, existing approaches to modelling structural change do not accommodate entry of new kinds of entities in the modeled system, nor do they explicitly represent any generic behavioural knowledge that agents possess. The innovation in the approach suggested in this section is that, besides structural knowledge, we equip agents with generic behavioural knowledge, which defines its external world behaviour once they enter a system. Therefore, we define coupling (rules of interaction) at the class level, allowing, besides simple cloning, the incorporation of new, predefined entities that know how to interact in a given setting. In terms of growing the composition tree, our approach allows both horizontal and vertical growth.

This research took as its starting point the Object Oriented/DEVS (OO/DEVS) framework for industry simulation.[38,39] OO/DEVS is based on the object oriented paradigm[40] and the DEVS formalism and has been used for modelling the electricity[41,42] and mobile telecommunications industry.[43] It provides for entity based modelling and separates the entities of the modelled system (the actual business model) from the simulation engine freeing, therefore the user from the mechanics of simulation. Classes are organised in a *class hierarchy* diagram that conveys 'is kind of' relations. Specific instances of classes that make up a model are stored in the *model decomposition diagram*, which details how instances are organised and supports aggregation (is part-of) relationships. Finally, interaction among entities is defined with a series of interaction rules that are specified in the *interaction* diagram. Simulation is carried out by taking the interaction rules between agents and translating them to *events* that are executed.

In order to develop VS/DEVS we extended OO/DEVS and unified a number of different techniques and approaches. We build our models in a hierarchical fashion and have maintained OO/DEVS' convention of not distinguishing between atomic and coupled models. These 'models' are the actual agents that inhabit the system we are interested in studying and every model can have behaviour of its own.

Agents have a set of *attributes*, a set of *actions* that define an agent's internal behaviour, and a set of *interaction rules* (IR), that defines the agent's coupling (or 'external behaviour', that is, how it interacts with others). Agents also support aggregation through a *Composition Structure* (CS), which stores a list of the entity's children. Finally execution priority is determined by a *SELECT* function that handles events scheduled at the same time. Two types of priorities can be specified, that is the object and interaction rule priority.

We extended the OO/DEVS approach by introducing a set of *structure events* (SE) and *behavioural knowledge* (BK) to the agents affected. Formally, we define agents as:

$$\langle \text{Attributes, Actions, IR, CS, SELECT, SE, BK} \rangle$$

where *Attributes*, *Actions*, *IR*, *CS*, and *SELECT* are defined as before. *Structure events* (SE) are similar to Zeigler's[21] *forward models*. They affect both the composition of the entities and their behaviour and we can think of them as an agent's declarative structural knowledge, that is, how the system's composition, coupling and taxonomy is to change given that a certain condition holds. There is no distinction between structure events at the atomic and the coupled level, as is the case in the work of Pawletta *et al.*[22,23] Structure events support declarative structural knowledge and specify what happens when a certain condition is met at the object where the event is defined. A structure event is formally defined as follows:

$$\langle \text{condition, receiver, action, beforeAction, afterAction, selector, time} \rangle$$

The *condition* (if) part of a structure event can be any statement that evaluates to true or false. The receiver is the object affected by the structure event; it can be either the object for which the structure event is defined or any other object that is part of the existing system. This is how we allow for both autonomy and control of agents, as any object can influence itself and any other object. Structured events are scanned and triggered by a standard forward chaining mechanism.

In VS/DEVS there is support for six kinds of structure event *actions*, namely

1. *Entry-Add Sibling* This option grows the composition tree horizontally.
2. *Entry-Add Child* The objective here is to grow the composition tree vertically. By doing this we can introduce new kinds of entities into the modeled system.
3. *Exit* Self-explanatory; should the *condition* hold true, the *receiver* object, along with its children, is deleted from the simulation model.
4. *Combine* Under this option, the *receiver* is combined with the object where the structure event is defined. A series of generic *onMerge* actions have to be implemented in the *selector* field (discussed below) that will take care of the details in combining, say, two merged firms.
5. *Add Behaviour—Message* Should a certain condition hold, the agent 'owning' the structure event initiates a new interaction with the *receiver* agent. In this way interaction occurs only when it is needed, resulting in a more intuitive and parsimonious model representation. As structure events are evaluated at run-time, interaction between two agents can be defined dynamically.
6. *Remove Behaviour—Message* This removes an interaction between the owner of the structure event and the *receiver* agent.

The *beforeAction* and *afterAction* fields determine what is to be executed at the agent where the structure event is defined, before and/or after the *action* part is ran. After one of the actions defined above has been performed, we can define a method to be executed at the *receiver* object. This is the *selector* field and supports the view of structure events as interaction rules. Finally, *time* defines whether a structure event is to be scheduled for immediate execution or with a delay.

As we discussed earlier, the only kind of knowledge that the original DEVS framework supports is taxonomic, or structural, knowledge. Our implementation, however, goes beyond simple cloning of existing agents and allows for new types of agents to enter the system. More specifically, new instances of predefined classes can be dynamically incorporated into the modelled system. However, to go beyond simple cloning, we would need to define a series of methods that spell out how an entity might behave once it becomes part of a system. In order to do this we store 'generic behavioural knowledge' (BK); in other words, we define a series of interaction rules that define interactions at the class level. As an example, class *FIRM* may have a class interaction which states that whenever a *FIRM* sets its price it interacts with a *MARKET* that notes that price. We therefore define the properties of a class as not only its attributes and actions, but also as a set of *class interaction rules* that capture the generic behavioural knowledge. When a new instance is introduced, say *FirmABC*, the class interactions defined at the *FIRM* class level are inherited and are (computationally) defined for the actual instance and the new entity can start operating in the new system. This is a unique feature of VS/DEVS that sets it apart from other formalisms in that it allows the incorporation of new types of entities in the system being modeled by supporting the incorporation of generic behavioural knowledge.

In developing VS-DEVS, we have unified a number of different techniques and approaches as detailed in Figure 2. More specifically, the generic behavioural knowledge and structure events define an agent's knowledge base. The agents that we model are cognitive, autonomous ones that react to changes in their environment, each with declarative knowledge about how to react to these changes and, hence, *intelligent*. Structural knowledge is defined as a set of $\{if \ldots then \ldots\}$ rules, whereas generic behavioural knowledge is defined as a set of class interactions. This 'blend' of intelligent agents and rule-based simulation is the 'heart' of VS/DEVS. The framework allows a rich knowledge-driven (rule-based) specification for describing a wide range of structural changes and allows the evolution of *both* structure and behaviour over time.

Implementation

The simulation engine of VS/DEVS has been implemented in *Visual Smalltalk*.[44] There is a generic class that provides the functionality to store messages, class messages and structure events for each of the agents defined in the simulation. Each agent in the simulation is associated with

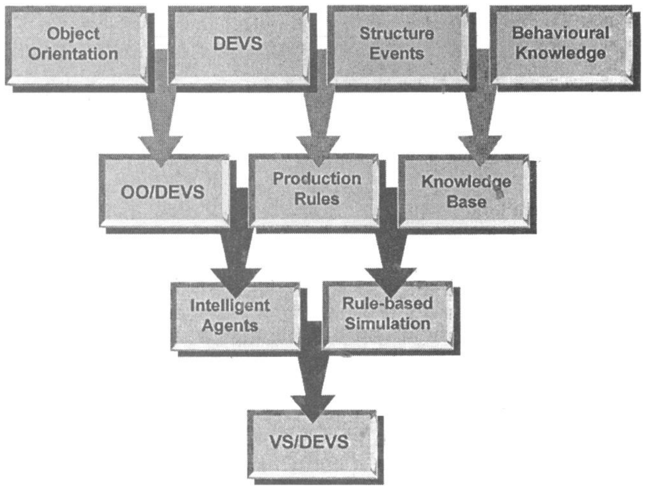


Figure 2 Roadmap to VS/DEVS.

a *simulator* that executes the simulation. The overall consistency and execution of the model is maintained by the *ObjectManager*, which keeps a list of all objects in the simulation and handles registration of new objects and deletion of others. Registration refers to checking and maintaining the consistency of the resulting model and to the updating of interactions of new or deleted entities.

The evaluation of the modular-hierarchical system is based on defining and implementing any structure events and on passing messages that result from the interaction rules between simulation objects. Before the start of a simulation, a *Simulator* object is attached to every agent in the system. The simulation of the model starts at the outermost simulator and the object manager checks for structure events. At each step the list of structure events of all instances is scanned to determine whether a condition for a structure event has been met and if so what action is to be followed. The actions generated by structure events, along with the simulation messages generated by the interaction rules are ordered by simulators, according to their execution time, and object priority. Then the action with the highest priority (the imminent action) is executed. If there is a delay specified, the simulation clock is advanced. The processing of an action can trigger further actions according to defined interaction rules. The simulation stops when the simulation time is exceeded or there are no further actions to be undertaken, in which case the simulation is blocked. Note that the definition of structure event conditions and actions as well as interaction rules, is the task of the modeller; the consistency of the overall model and registration of new entities is maintained by the *object manager*. Figure 3 shows schematically the process described above.

A graphical user interface (GUI) assists in model creation by exploiting standard Windows characteristics and by

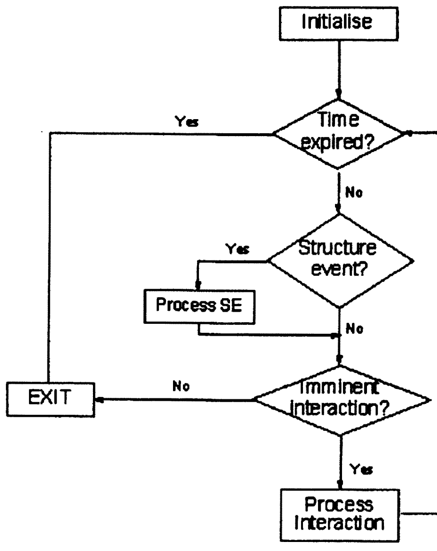


Figure 3 How simulation is executed.

making the model design phase as intuitive to the modeler as possible. The GUI supports a number of diagrams and dialogs that facilitate model development. As outlined earlier, there are three main diagrams, the class hierarchy, model decomposition and interaction diagrams, shown in Figure 4.

The class hierarchy diagram shows the contents of the current model base, whereas the model decomposition diagram shows how specific instances of classes are structured in a system. Finally, the interaction diagram shows interactions between entities. In addition to these diagrams, a number of dialogs facilitate setting up interaction rules and handling and manipulating knowledge. Figure 5 shows how to use the GUI to define class interaction.

Behavioural knowledge is stored at the more abstract (class) level with, say, a Firm (capitalised to denote it is a class) interacting with a Market. Note that the dialog that defines interaction rules between specific instances only differs in that it uses specific object instance names and these interactions are stored at the instance, instead of the class level. Finally, Figure 6 shows the GUI interface dialog to define a structure event.

Market dynamics: An extension to Nelson and Winter[45]

In the previous section we developed a framework that enables us to model systems whose underlying structure can change over time. In this section we present a simple application that establishes the technical feasibility of our framework. We extend a well-known evolutionary economics model by introducing simple structure events that produce a rich system trajectory. The aim of this section is not so much to make a contribution to economic theory, but rather to provide an example of how VS/DEVS can be used in practice to extend traditional modelling. We have modeled individual firms and have made explicit the processes by which they will enter or exit the industry. Although the approach followed in this example is by no means the only way to introduce entry and exit in this model, it can be argued that it is natural and intuitive, while at the same time adaptable to the modelling of much more complex systems.

In the last twenty years, industrial economics[46–49] has produced a number of models to explain competition and to account for a number of empirical regularities, such as profit rates, entry and exit rates, and firm size distribution. In these models, which were introduced with the seminal work of Jovanovic,[50] one finds an (almost) complete description of the firm's decisions and market processes; for instance, exit in the industry occurs when a firm's estimated efficiency falls below a threshold value.

In general, one can distinguish between passive and active learning models. In both models, firms decide whether to remain in the industry or not. In passive learning

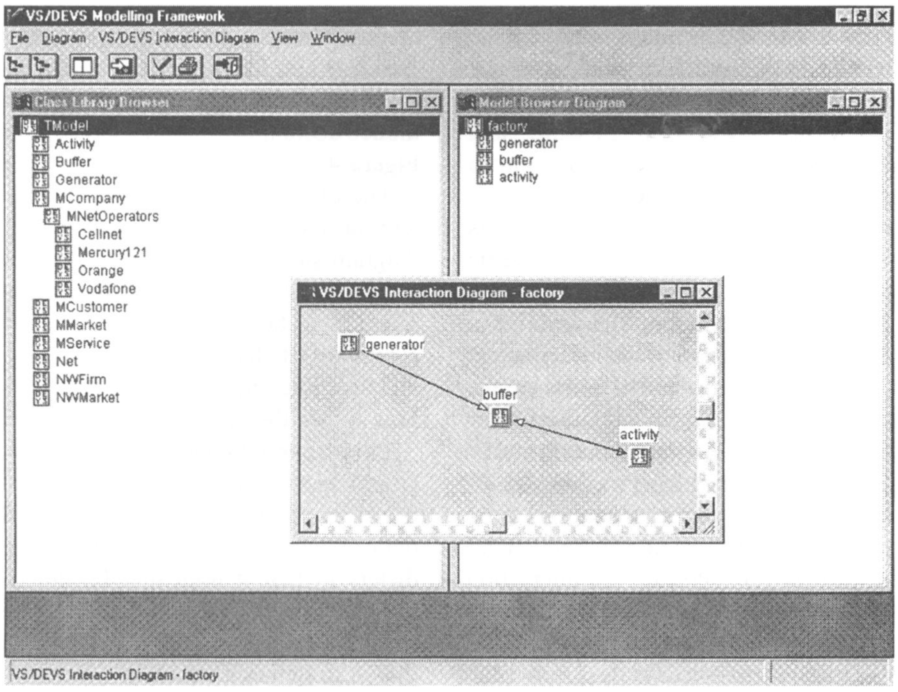
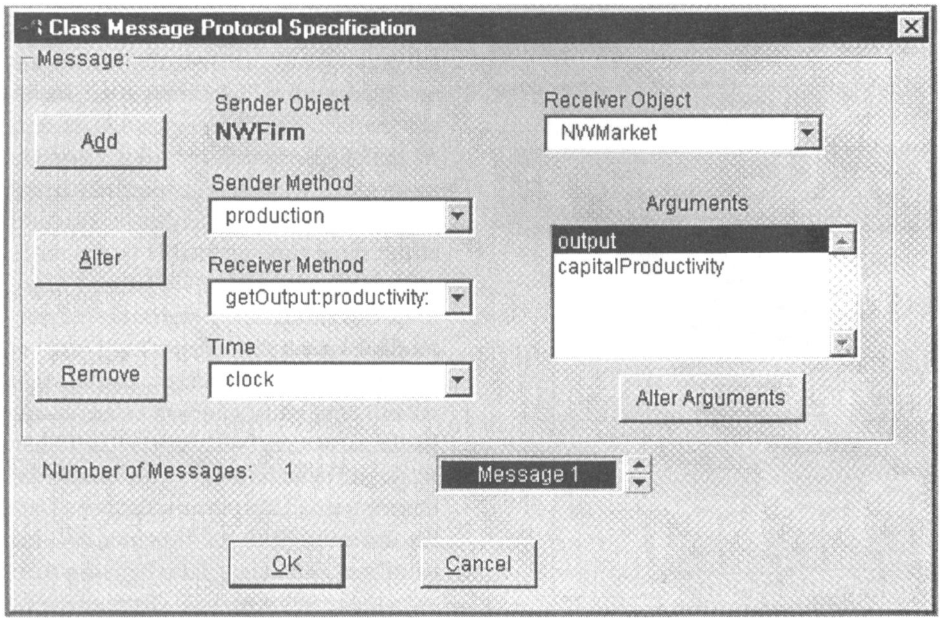


Figure 4 The hierarchy, decomposition and interaction diagrams.

models,[50,51] firms do not engage in any activity that can improve their efficiency/productivity. In active learning models[45,52] firms try to increase their productivity as a result of investment in R&D. Active learning models make market shares endogenous to the process of innovation and thus differ significantly from the structure–conduct–performance paradigm, which is the 'traditional' industrial organisation framework.

In their seminal work on evolutionary economics, Nelson and Winter[45] describe a model of technological change as competition between innovating and/or imitating firms. We concentrate on the model of 'Schumpeterian competition'


Class Message Protocol Specification

Message:

Sender Object: NWFirm

Receiver Object: NWMarket

Sender Method: production

Arguments: output, capitalProductivity

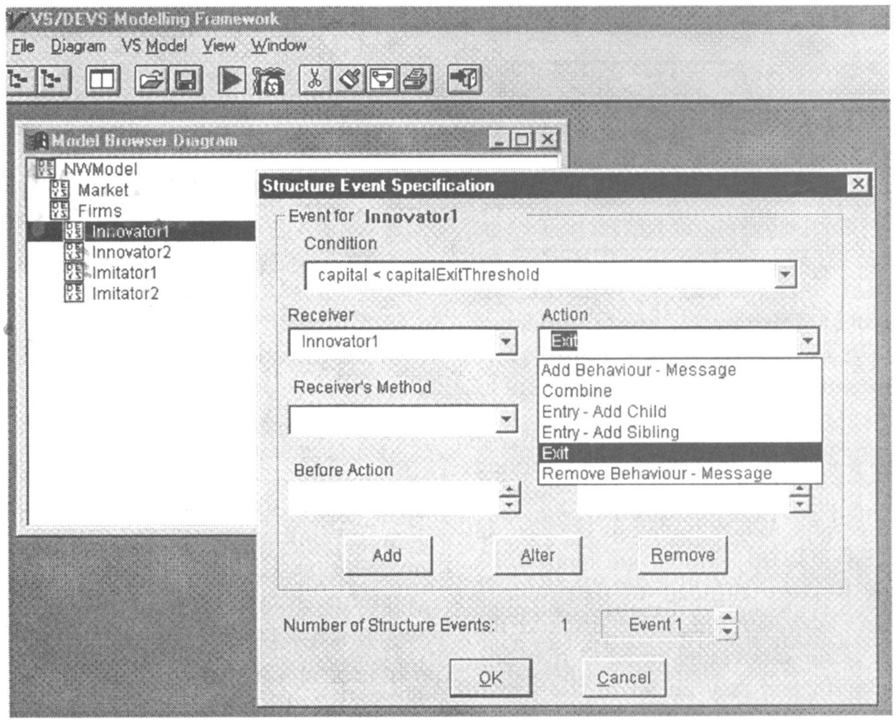
Receiver Method: getOutput:productivity:

Time: clock

Number of Messages: 1

Message 1

Figure 5 Using the GUI to define a class message.



Condition: capital < capitalExitThreshold
Receiver: Innovator1
Action: Exit
Number of Structure Events: 1 Event 1

Figure 6 Using the GUI to define a structure event.

which deals with the evolution of production techniques of a stylised industry, which produces a homogeneous, undifferentiated product. We start with n firms, all of which are explicitly created and at time t firm i is characterised by a capital stock K_{it} and productivity of capital A_{it}. Each of these firms decides on a production level and the output of the industry is found by simple aggregation. An exogenously defined demand curve determines the market price per unit of output which, in turn, determines turnover and net profit for each firm. Firms undertake, at a cost, imitative and/or innovative activities in order to improve their capital productivity. Technology improves with time and if an innovating firm succeeds in its R&D, it 'gains' this new technology and its efficiency increases. Once firms have determined their productivity, they determine the amount of capital they will have available in the next period by considering their desired investment and the financial constraints they have. Note that this is model determines *probabilistically* what happens in each period.

The extension implemented here models the entry and exit of firms in a marketplace, a feature not present at Nelson and Winter's original work.[45] Besides using the same production and innovation/imitation methods described in the original, we model entry and exit have implemented these processes using a set of two *structure events*. More specifically,

1. *Entry* in the market occurs when average profits per unit capital exceed some threshold value x. We assume that there are no barriers to entry and there are no entry costs for the new firm. The new firm is similar to existing ones (as they all produce a homogeneous good) and what is determined randomly is whether the new firm will be an innovator or imitator. Even though the assumptions appear to be unrealistic and despite the stylised nature of the model, we could find parallels with, say, the recent development of subscription free internet providers in the UK who essentially outsource everything and enter the industry with minimal costs.
2. The condition for *exit* is met when a firm's capital falls below some threshold value x.

In running the model we consider three variants; one with no entry or exit (the 'standard' Nelson and Winter model (Model A), one with entry only (Model B) and one with both entry and exit (Model C). In all models, we start off with four firms, two innovators and two imitators (see Figure 7), each having 25% of the market.

The arrival in the market of new firms suppresses the average market price which slowly recovers (see Figure 8). One can actually note the interesting cyclical behaviour that results from entry and exit in relation to the basic N&W model. The same also applies for profits; the higher the entry rate, the further do profits fall. When only entry is allowed (model B) prices remain consistently below those of the base case model, but with the introduction of exit (model C) price spikes occur when exit takes place. In all models, a steady state is reached after about 45 periods.

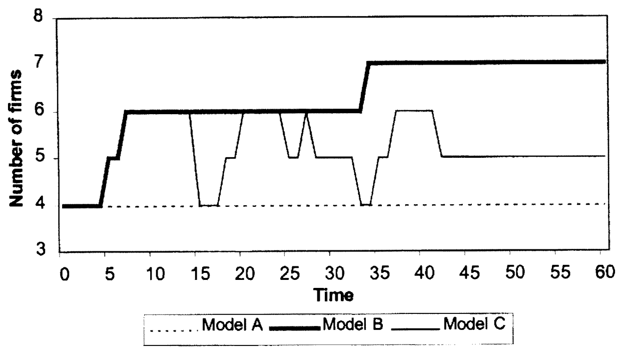


Figure 7 Number of firms in models A, B, C.

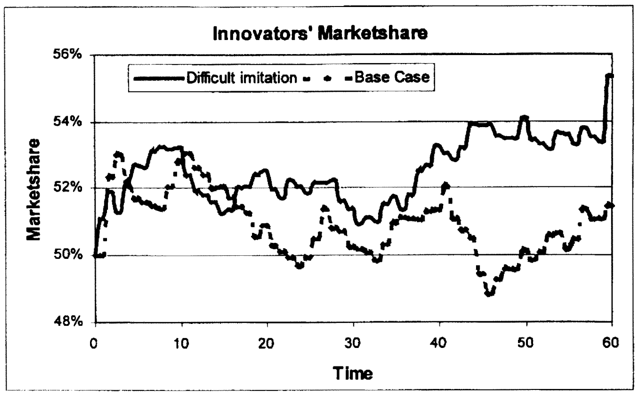


Figure 9 How innovators' market shares evolve.

We also considered a number of scenarios, by varying the degree of difficulty of imitation activities, and obtained a series of interesting results. When imitation is easy, an imitator might well be able to exploit an innovator's advantage and it will be very difficult to determine a 'winner' in the market. However, the more difficult the imitation, the easier it is for innovating firms to maintain their lead in the long term and stay ahead of the game. Figure 9 shows the total market share of innovators in the two scenarios described above.

VS/DEVS in practice

In this section, having developed an extension to the Nelson and Winter model[45] that validates the VS/DEVS framework, we describe how we used the approach for a business application that addresses policy issues in telecommunications markets. A full description of the model and its results appear in Reference 53.

A number of countries are trying to promote competition in telecommunications markets and specifically at the local loop as the 'internet revolution' continues and demand for higher bandwidth to the home increases. The issue for regulators is how best to promote entry, investment and competition. Regulators can essentially approach competition by infrastructure (or facilities-based) and service competition. Under infrastructure competition, a new entrant must invest in its own network and cannot use the incumbent's facilities to convey calls. An alternative to infrastructure entry is service, or resale, entry. Besides these two 'extreme' cases we also consider a 'mix' of service and infrastructure competition, comprising equal access/carrier pre-selection and local loop unbundling (LLU). Under these settings, an entrant builds part of a network and can lease the local loop or any other unbundled network element off the incumbent.

There is no empirical research that compares facilities with service competition[54] and the question whether allowing facilities-only entry is the best way to promote competition in a market is still an open one. Traditional analyses have been relatively unsuccessful in determining which kind of competition delivers better results with strong arguments on either side.

Our model attempts to test different regulatory approaches and firm behaviour in a competitive market without assuming full rationality or equilibrium in the market. Therefore, unlike a traditional economics approach

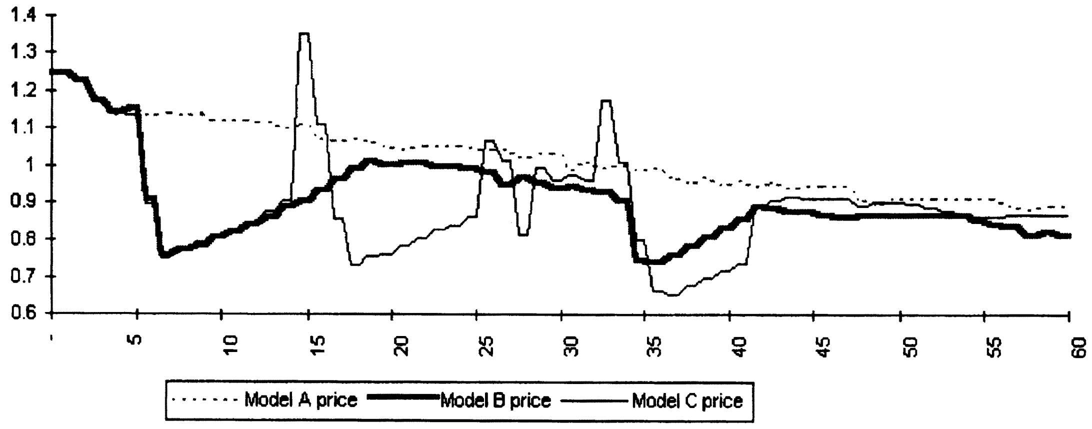


Figure 8 Price evolution in models A, B, C.

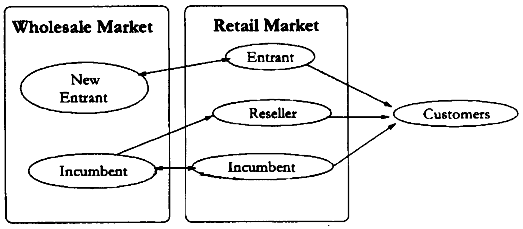


Figure 10 A simplifed way telecommunications markets work.

that would consider the long run industry equilibrium before/after firm entry, we use our model to represent how the market is designed to operate and to examine the strategic behaviour of the incumbent and new entrants.

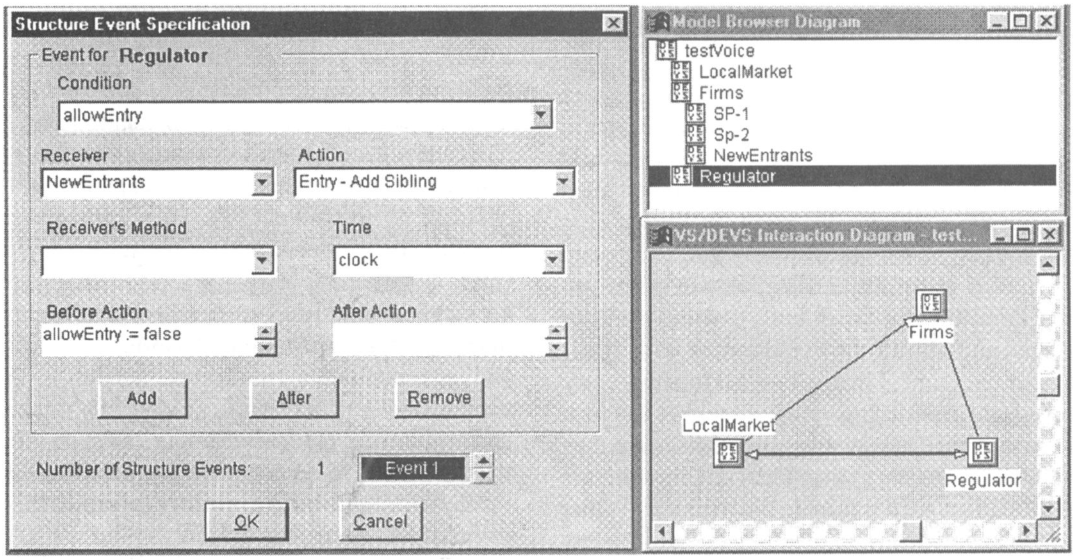
We represent the regulator and each firm as autonomous adaptive agents; this means that our model explicitly maps agents of interest (regulator, incumbent, new entrants) with their attributes and behaviour and therefore incorporates both dynamic and detailed complexity. VS/DEVS allows us to consider not only how the structure of a system influences the behaviour of its participants, but also how participants' behaviour can change the structure of the system. In addition, the framework allows a rich knowledge-driven (rule-based) specification for describing a wide range of structural changes and allows the evolution of both structure (firm entry) and behaviour (investment and pricing) over time.

A simplified way to consider the way the market works appears in Figure 10. Customers buy services in a retail market from service providers who, in turn, either own their own infrastructure (incumbent or new entrant) or resell the incumbent's capacity. To simplify our model, we consider only residential voice services and model a retail market where customers buy voice services.

The model describes the way the retail market operates. We consider the evolution of the market on a quarterly basis and thus every simulation time step corresponds to a quarter. Firms decide on the retail prices they will charge and based on that firm's price relative, the perceived quality of service (QoS) and quality of network (QoN) and the overall level of demand, the market 'determines' the number of new customers that would join firm *i*.

To exploit VS/DEVS features, we define structure events at the regulator level, as regulators 'allow' different kinds of entry (service, infrastructure). In our model we compare three different kinds of entry and competition; infrastructure only entry, service only entry, where new entrants are under no requirements whatsoever to build any of their own network. Finally, we consider 'service and infrastructure entry' where entrants are allowed to enter with limited or no network of their own, but are subsequently required to invest in their own facilities. Figure 11 below shows how we use VS/DEVS to define structure events.

The regulator allows entry if concentration in the market, measured by the Herfindahl (H) index, is above the level of 0.4. H is calculated as the sum of the squared market shares for all firms in an industry. We use $H=0.4$ as that value works as a good proxy for a regulator who wants to promote competition and innovation. Results from this model (Figure 12) suggest that a 'mix' of infrastructure and service competition stimulates investment by both incumbents and entrants and offers better consumer benefits.


Structure Event Specification
Event for Regulator
Condition: allowEntry
Receiver: NewEntrants
Action: Entry - Add Sibling
Receiver's Method:
Time: clock
Before Action: allowEntry := false
After Action:
Number of Structure Events: 1 Event 1

Model Browser Diagram
testVoice
LocalMarket
Firms
SP-1
Sp-2
NewEntrants
Regulator

Figure 11 Using the GUI to define structure events.

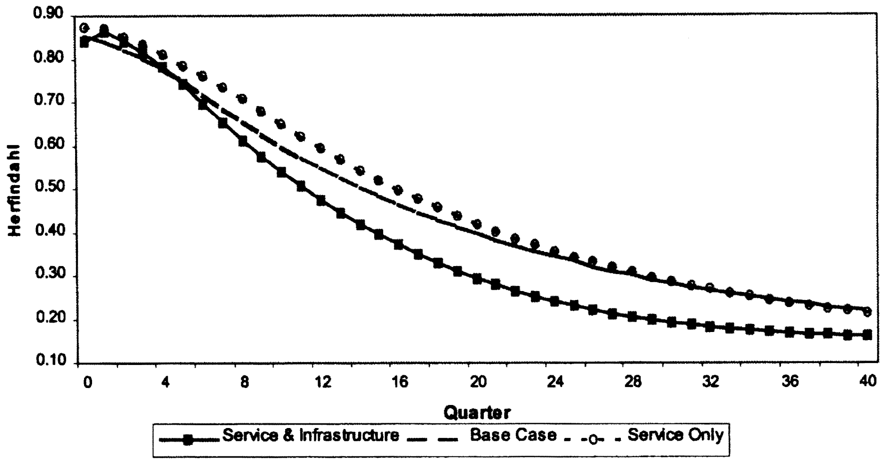


Figure 12 Sample results of telecommunications policy simulation.

Conclusions

The paper explains the motivation for using variable structure modelling in industry simulation. It provides an original and thorough review of developments in industry simulation and variable structure modelling and discusses its links to agent-based and knowledge-based simulation. The review of existing approaches identified the potential, but also some gaps in earlier work. The VS/DEVS formalism proposed is based on a synthesis of concepts from industry simulation, object orientation, and rule-based simulation and can provide the basis for simulation of industries and markets that undergo rapid structural change.

VS/DEVS constitutes a significant advancement in variable structure simulation technologies. Unlike earlier formalisms, it does not make a distinction between controlling and controlled entities, as this would be a limitation for an industry simulation platform. By defining structure events that can modify any part of a system, it strikes a balance between autonomy of agents and control. In a market setting there are agents that directly control others and implicitly or explicitly define the structure of an industry, for example, a regulator. At the same time, many structural changes within industries and organisations are not dictated by anyone, but are the result of adaptation, as agents, be it individuals, firms or organisations, change their beliefs and strategies. VS/DEVS provides a rich knowledge-driven (rule-based) specification for describing a wide range of structural changes and allows the evolution of both structure and behaviour over time. Finally, in order to allow entry of new kinds of agents generic knowledge should be stored at the class level so that once new agents enter the system they are able to interact with existing ones. The variable structure aspects of the formalism are a transparent extension to other industry simulation approaches, in that they do not interfere with traditional systems modelling and can be left aside, if the modelling context does not demand them.

The paper describes the implementation of the framework in object-oriented environment. This implementation allows the high level specification of structural events using a graphical user interface. The paper demonstrated the modelling platform using a theoretical model of competition that extends the original model of Nelson and Winter[45] by accommodating entry and exit to the market by means of structural events. In addition, we describe how we use VS/DEVS for a business application to the telecommunications industry. The framework proved to be particularly useful in capturing the fluid and rapidly changing structure of telecommunications markets, and has helped us to generate and analyse alternative futures.

Variable structure simulation is particularly suited to the modelling of industries whose structure evolves over time, such as the converging infocom industries. The framework allows managers and policy makers to build and experiment with models of industry evolution by considering individual actors and their choices. Building models in such a way allows us to explore *individual* firm strategies, a regulator's policy, determinants of customer choice and how they affect each other in a dynamic setting. Therefore, management and policy decision variables could become endogenous to the model and affect all industry participants.

Building models in VS/DEVS involves mapping a mulitplicity of decision-making levels for many agents (firms, customers, regulator). Modelling validation and verification can become a more challenging task when many agents are involved. A shortcoming of this approach and any simulation model in general is that it cannot contain any mathematical proofs, but rather illustrations and insights of model behaviour; as such, significant care must be taken to demonstrate a model's validity.

VS/DEVS allows us to consider what will happen once the environment has changed. How can individual firms respond to this change? How can firms make better and quicker use of change in their environment? Variable structure simulation allows us to ask more questions and consider a number of scenarios, where processes of structural change are produced by actors operating on the same systems, but with differing information, goals, and learning abilities.

Acknowledgement—In carrying out his research, the first author was supported by a grant from British Telecommunications plc.

References

1 de Geus A (1988). Planning as learning. *Harvard Bus Rev* **66**: 70–74.

2 Campbell A *et al* (1985). Experimental mathematics: the role of computation in nonlinear science. *Commun of the ACM* **28**: 374–384.

3 Casti J (1997). *Would-be Worlds: How Simulation is Changing the Frontiers of Science*. John Wiley & Sons: New York.

4 Allen PM and McGlade JM (1987). Modelling complex human systems: a fisheries example. *Eur J Opl Res* **30**: 147–167.

5 Carley K and Prietula MJ (1994). *Computational Organization Theory*. Lawrence Elbaum Associates: Hillsdale, New Jersey.

6 Forrester JW (1961). *Industrial Dynamics*. The MIT Press: Cambridge, Massachusetts.

7 Legasto AA, Forrester JW and Lyneis JM (1980). *System Dynamics*, Volume 2 in TIMS Studies in Management Science. North Holland: Amsterdam.

8 Ford A (1997). System dynamics and the electric power industry. *Sys Dynam Rev* **13**: 57–85.

9 Morecroft JDW, Lane DC and Viita PS (1991). Modelling growth strategy in a biotechnology startup firm. *Sys Dynam Rev* **7**: 93–116.

10 Lyneis JM (1994). *Tactical vs Strategic Approaches to Competitive Positioning—An Example from the Telecommunications Industry*. Proceedings of the 1994 International System Dynamics Conference. (Available from University of Stirling, FK9 4LA, Scotland).

11 Hall RI (1976). A system pathology of an organization: the rise and fall of the old Saturday Evening Post. *Admin Sci Q* **21**: 185–211.

12 Ginsberg A (1997). 'New age' strategic planning: bridging theory and practice. *Long Range Plan* **30**: 125–128.

13 Orcutt G (1957). A new type of socio-economic system. *Rev Econ and Stat* **39**: 116–123.

14 Orcutt G (1976). Policy Exploration Through Microanalytic Simulation, The Urban Institute: Washington.

15 Bergmann BR (1990). A microsimulated model of inventories in interfirm competition. *J Econ Behav and Organiz* **14**: 65–77.

16 Kirman A (1997). The economy as an evolving network. *J Evol Econ* **7**: 339–353.

17 Oeren TI (1975). Simulation of time-varying systems. In: Rose J (ed). *Advances in Cybernetics and Systems*. Gordon and Breach Science Publishers: UK, pp 1229–1238.

18 Zeigler BP and Oeren TI (1986). *Multifacetted, Multiparadigm Modelling Perspectives—tools for the 90s*. Proceedings of the Society for Computer Simulation Winter Simulation Conference, ACM, USA, pp 708–712.

19 Zeigler BP (1984). *Multifacetted Modelling and Discrete Event Simulation*. Academic Press: London.

20 Zeigler BP (1986). Towards a simulation methodology for variable structure modelling. In: Elzas MS, Oren TI, Zeigler BP (eds). *Modelling and Simulation Methodology in the Artificial Intelligence Era*. North Holland: Amsterdam, pp 195–210.

21 Zeigler BP (1990). *Object-Oriented Simulation with Hierarchical, Modular Models—Intelligent Agents and Endomorphic Systems*. Academic Press: London.

22 Pawletta T and Pawletta S (1995). Design of an object-oriented simulator for structure variable systems, IMACS, systems analysis. *Modelling & Simulation,* **18/19**: 471–474.

23 Pawletta T *et al* (1996). An Object Oriented Framework for Modelling and Simulation of Variable Structure Systems. In: Ingalls VW, Cynamon J and Saylor JV (eds). *Soc for Computer Simulation International*, Portland OR, USA pp 8–13.

24 Uhrmacher A (1996). *Variable Structure Modelling—Discrete Events in Simulation*. Proceedings of the Fifth Conference on Artificial Intelligence, Simulation and Planning in High Autonomy Systems, IEEE Press, USA pp 133–140.

25 Hogeweg P and Hesper B (1989). An adaptive, self-modifying, non-goal directed modelling methodology. In: Elgas MS, Oeren Ti, Zeigler BP (eds). *Modelling and Simulation methodology: Knowledge Systems' Paradigms*. Elsevier Science, pp 77–92.

26 Uhrmacher A (1992). EMSY: An Extended Modelling System. In: Houstis EN and Rice JR (eds). *Expert Systems and Symbolic Computing for Scientific Computation*. North Holland: Amsterdam, pp 323–332.

27 Mohring M (1992). MIMOSE: A functional language for modelling and simulation of individual behaviour in interacting populations. In: Faulbaum, F (ed). *Stoft-Stat '91: Advances in Statistical Software 1992*. Fisher: Stuttgart, pp 327–334.

28 Klee A, Mohring M and Strotmann V (1994). The modelling and simulation of the dynamic change of structure in birth/death processes. In: Faulbaum F (ed). *Advances in Statistical Software* 4, 1994. Gustav Fischer: Germany, pp 171–178.

29 Gilbert N and Doran J (1994). *Simulating Societies*. UCL Press: London.

30 Zeigler BP, Kim TG and Lee C (1991). Variable structure modelling methodology: an adaptive computer architecture example. *Trans Society for Comput Simul* **7**: 291–319.

31 Zeigler BP and Reynolds RG (1984). Modelling alternative structures for time critical corporate adaptation. In: Singh MD (ed). *Real Time Control of Large Scale Systems* 1984. Springer Verlag: Heidelberg.

32 Zeigler BP and Reynolds RG (1985). *Towards a Theory of Adaptive Computer Architectures*. Proceedings of the Fifth International Conference on Distributed Computing Systems, Computer Society Press, USA pp 468–475.

33 F Barros, M Mendes and B Zeigler (1994). *Variable DEVS-Variable Structure Modelling Formalism: An Adaptive Computer Architecture Application*. Proceedings of the Fifth Annual Conference on Artificial Intelligence, Simulation and Planning in High Autonomy Systems, IEEE Computer Society Press, USA pp 185–191.

34 Zeigler BP and Louri A (1993). A simulation environment for intelligent machine architecture. *J Parallel and Distrib Comput* **18**: 77–88.

35 Chean M and Fortes LAB (1990). A taxonomy of reconfigurable techniques for fault-tolerant processor arrays. *IEEE Comp* **23**: 55–69.

36 Domingo C, Tonella G and Teran O (1996). *Generating Scenarios by Structural Simulation*. Proceedings of the Sixth Annual Conference on Artificial Intelligence, Simulation and Planning in High Autonomy Systems. The University of Arizona, IEEE Press, USA pp 331–335.

37 Domingo C *et al* (1996). *Simulation of Structural Change*. Proceedings of the Eighth European Simulation Symposium: Simulation in industry, Genoa, 1996. Society for Computer Simulation International, Portland, USA pp 112–116.

38 Ninios P (1994). *An object Oriented-DEVS Framework for Strategic Management and Industry Simulation*, PhD Thesis. London Business School: University of London.

39 Ninios P, Vlahos K and Bunn DW (1995). OO/DEVS: A platform for industry simulation and strategic modelling. *Decis Support Sys* **15**: 229–245.

40 Booch G (1994). *Object-Oriented Analysis and Design*. The Benjamin/Cummings Publishing Company: Redwood City, CA.

41 Ninios P, Vlahos K and Bunn DW (1995). Industry simulation: system modelling with an object oriented/DEVS technology. *Eur J Opl Res* **81**: 521–534.

42 Vlahos K, Ninios P and Bunn DW (1998). An integrative modelling approach for understanding competitive electricity markets. *J Opl Res Soc* **49**: 187–199.

43 Christodoulou K, Jensen K and Vlahos K (1999). Using object oriented simulation to explore substitution between technologies: an application to the UK mobile telecommunication industry. *Technol Forecast and Social Change* **62**: 203–217.

44 ParcPlace-Digitalk Inc. (1995). Visual Smalltalk Tutorial and Language Reference. ParcPlace, Sunnyvale, CA.

45 Nelson R and Winter S (1982). *An Evolutionary Theory of Economic Change*. Harvard University Press: Cambridge, MA.

46 Tirole J (1988). *The Theory of Industrial Organization*. The MIT Press: Cambridge, MA.

47 Scherer FM (1980). *Industrial Market Structure and Economic Performance*. Rand McNally: Chicago, IL.

48 Kamien MI and Schwartz N (1982). *Market Structure and Innovation*. Cambridge University Press: Cambridge.

49 Laffont JJ and Moreaux M (1991). *Dynamics, Incomplete Information and Industrial Economics*. Basil Blackwell Ltd: Oxford.

50 Novanovic B (1982). Selection and the evolution of industry. *Econometrica* **50**: 649–670.

51 Hopenhayn (1992). Entry, exit and firm dynamics in long run equilibrium. *Econometrica* **60**: 1127–1150.

52 Ericson R and Pakes A (1995). Markov-perfect industry dynamics: a framework for empirical work. *Rev Econ Stud* **62**: 53–82.

53 Christodoulou K and Vlahos K (2000). *A model-based Comparison of Infrastructure vs Service Competition at the Local Loop*, Presented at the XIII Biennial Conference of the International Telecommunications Society, Buenos Aires. (Available from author).

54 Woroch G (1998). *Facilities Competition and Local Network Investment: Theory, Evidence and Policy Implications*. University of Berkeley, Consortium for Research on Telecommunications Policy Working Paper CRTP-47.

Received November 1998;
accepted March 2000 after three revisions