Applying Agent-oriented Design to a Sail Boat Simulation

Paul Ness Meurig Beynon Yun Pui Yung

Department of Computer Science
University of Warwick CV4 7AL

Abstract

This paper describes the application of an agent-oriented modelling technique outlined in [BBY92] to a sail boat simulation. This case-study is chosen to demonstrate the principles of the modelling method, which is based on the systematic construction of definitive environments associated with different modes of observation of a real-world system. The potential for integrating several different views of physical phenomena is also illustrated.

Introduction

Computer-based modelling and simulation plays an increasingly significant role in the design of modern engineering systems. Traditional computer packages for engineering design typically provide toolkits for numerical solution of differential equations, user-interaction and visualisation. Such packages are very effective in constructing models for systems whose behaviour can be completely described analytically, but have several significant limitations. In this paper, we identify some of these limitations, and explain and illustrate how they can be overcome by adopting a new modelling paradigm.

Our chosen case-study is a Sail Boat Simulation (SBS). The physical properties of the sail, rig and hull are all described within the model. Interaction is via the sailor agent who adjusts the main-sheet (and hence sail) and rudder. Following the principles introduced in [BBY92], the state of the SBS is described using a definitive script, and an agent-oriented design method is used to determine and construct each of the sail boat components: sail, rig, hull and sailor. The resulting simulation program combines a model of the sail boat dynamics, a simple graphical animation and an interface through which the user can play the role of the sailor (see Figure 1).

The SBS case study illustrates several advantages of our design method. These include:

- a close correspondence between values of variables in the computer model and physical observations such as might be made of a real sail boat. This makes the simulation model easy to interpret.
- direct feedback through convenient animation of the model from the early stages of the design. This speeds the development process by allowing the user to identify any misconceptions at an early stage.

- a generality intrinsic to the modelling technique that means that the latent functionality of the model at every stage is far greater than that represented in the final simulation. This ensures that the model can be easily modified to meet new requirements.

These advantages will be discussed in connection with an exposition of the modelling method that describes: how the relationships between agents and observations are represented in *definitive environments*, how the agent development order is determined, and what issues arise in the interface design. The SBS case-study will be used to illustrate the principles of *situated modelling* and to motivate possible improvements to the method and supporting tools.

1. Background

1.1. Motivation: from Systems Engineering to Software Engineering

The application of computing principles and techniques to engineering modelling has a long history. The first and most enduring applications of the high-level language Fortran were to the numerical solution of equations for engineering applications. The development of computer graphics and windowing environments introduced visualisation to engineering models. More recently, object-oriented methods have been a major influence on the framework for engineering design.

At the present time, the relationship between computer science and engineering is undergoing a radical change, as the processes of modern engineering systems design and software development have more and more in common. As computer modelling plays an ever more significant part in the design of engineering systems, and computer applications increasingly concern hybrid *reactive systems* [H92] in which electronic devices, mechanical components and human agents interact, the boundaries between systems engineering and software engineering are no longer well-defined.

In this context, it is interesting to consider the scepticism of Brooks in his well-known paper *No Silver Bullet* [Br87]. Brooks argues that none of the developments of computer science that have so far been widely applied to computer modelling for engineering, such as the use of high-level languages, windowing environments and object-oriented methods, has addressed the *essence* of complex system design.

Brooks' characterises the essence of a 'software entity' in terms of four attributes: complexity, conformity, changeability and invisibility. The considerations that Brooks applies to software systems increasingly apply to the computer models of engineering systems:

- complexity

  Brooks makes an important distinction between models of physical systems that depend upon simplification through mathematical abstraction and a software system whose relation to its environment must be specified in precise detail. Mathematical models of an engineering system that describe the behaviour using differential equations are no longer appropriate when the control of processes is influenced in a complex manner by discrete events. Control of this nature is a feature of systems that are designed for "intelligent" response.

- conformity

  The environment in which a software system operates is typically specified by artificial constraints and conventions rather than natural laws. The more we try to make comprehensive computer models of engineering systems, the more we shall need to take account of arbitrary empirical constraints and data values.

- changeability

  Brooks draws attention to the unprecedented degree of maintenance and revision to which software systems are subject. As programmed electronic components play an ever greater role in engineering systems, so similar considerations will apply. What is more, the future development of complex systems will demand better methods of incremental modelling based on systematic revision of a design.

- invisibility

  The structure of a complex software entity cannot be conveniently represented visually, in contrast to conventional electro-mechanical systems that can be described by scale models, engineering drawings and circuit diagrams. Different considerations apply to the conception of mechanical and electrical systems, whose structure has a physical embodiment.

1.2. The Case Study: a Sail Boat Simulation

Our sail boat simulation is based on a simple dynamical model that is informally described in [G63]. The essential principles of sail boat motion can be explained with reference to Figure 1:

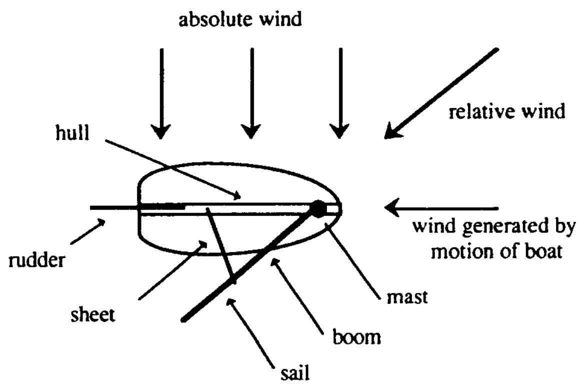

Figure 1

The motive force for the boat is generated by the action of the wind on the sail. This is due to two effects [M88]:

- a suction effect: when the wind blows across the face of the sail, it acts as an aerofoil;
- a pushing effect: when the wind blows at right-angles to the sail, it generates pressure on the sail.

The interaction between these effects is subtle, and, according to the Glénans Sailing Manual ([G63] p115), "controversy surrounds the aerodynamics of the sail". Our modelling approach gives us the flexibility to investigate different theories about sail aerodynamics, but such theories are not our primary concern in this paper. The model for the driving force on a sail we have studied in our simulation is based on empirical evidence from wind-tunnel experiments with model sails (cf. [G63] p148). A feature of our method is that we can readily run the simulation using different definitions of the driving force. For instance, we could adapt this definition interactively to take account of any influence of wind speed upon the profile of the driving force. We could also adapt the simulation to reflect the fact that the driving force depends not only upon sail orientation, but also on the manner in which the sail is manipulated into this orientation.

The forces exerted on the sail are transmitted to the boat via the rig, which consists of the boom, the sheet (the rope attached to the sail that is manipulated by the sailor), and the mast. The force upon the boat due to the wind resolves into three components:

- a force that tends to move the boat laterally in the water;
- a thrust that propels the boat along its axis of symmetry;
- a heeling force that pushes the sail sideways, and tends to capsize the boat.

The hull of the boat counteracts sideways motion of the boat in the water. The hull is ballasted to counteract heeling. The boat may also have a turning moment, depending on where the lateral thrust acts upon it.

1.3. Basic principles of the modelling method

Our modelling principles are similar to those discussed in [BBY92], where the specification of a Vehicle Cruise Control System is given. Our approach combines an agent-oriented analysis with a representation of system state by definitive scripts []. In this paper, we briefly review these techniques before examining the modelling process in more detail.

1.3.1. Agent-oriented Analysis

In agent-oriented analysis, the relevant observations of a system are identified and classified with respect to the state-changing agents. A special purpose specification notation LSD is used for this purpose [BBY92]. An illustrated extract from the LSD specification of the SBS appears in Listing 1 below. A distinctive feature of our approach is the way in which the synchronisation between changes to observations is represented.

```
agent sailor {
oracle
   sheetlenmin, sheetlenmax
   sheet_len
handle
   heading
   sheet_len
derivate
   turn = user_input(turn_type)
   sheetdir = user_input(sheet_type)
protocol
   turn == starboard → inc(heading)
   turn == port → dec(heading)
   (sheetdir == out) && (sheet_len < sheetlenmax) → inc(sheet_len)
   (sheetdir == in) && (sheet_len > sheetlenmin) → dec(sheet_len)
}

agent sail {
const
   drivingFminK = 29.0
   drivingFmaxK = 20.0
   turbulence = π / 4          // relative angle of the sail to wind when turbulence occurs
state
   sail_dir                    // sail direction as bearing [rad]
   driving_dir                 // sail driven anti/clockwise [1/-1]
   driving_force               // driving force of sail [N]
   sheet_angle                 // angle between keel and sail [rad]
   sail_area                   // effective sail area [m²]
oracle
   rel_wind_dir                // wind direction experienced by the sail [rad]
   rel_wind_speed              // wind speed experienced by the sail [ms⁻²]
   sailAvel
derivate
   sail_area = boom_len × mast_len × cos(list)
   sail_dir = heading + π + sheet_angle
   rel_wind_speed =
   √(wind_speed² + hull_speed² − 2 × wind_speed × hull_speed × cos(wind_dir − heading))
   rel_wind_dir = heading - asin(sin(wind_dir-heading) × wind_speed / rel_wind_speed)+π
   sail_wind = rel_wind_dir - sail_dir         // angle between apparent wind and sail
   driving_dir = (sin(sail_wind) × sin(sheet_angle) ≥ 0) ? 1 : -1
   drivingFmin is drivingFminK × sail_area
   drivingFmax = drivingFmaxK × sail_area
   driving_force =
        (abs(sin(sail_wind)) ≥ abs(sin(turbulence))) ? drivingFmin:
        abs(sin(4 × sail_wind)) × drivingFmax  + drivingFmin / turbulence × abs(sail_wind)
}
```

Listing 1: LSD specifications of sailor and sail agents in a Sail Boat Simulation

In the SBS, the principal agents are identified as the sail, the rig, the hull and the sailor. The relevant observations include: the physical attributes of the boat, the current values of variable parameters associated with a particular configuration of the sail and rig and the forces and velocities associated with the boat in motion. In discussing the behaviour of agents, we tend to adopt an anthropomorphic view, similar in spirit to [BSG84].

Our primary concern is to analyse the stimulus-response patterns governing the behaviour of agents (cf [D88]). A typical agent action is conceived as a redefinition of system parameters in response to a perceived change in its environment. In LSD, the parameters to which an agent can respond are classified as **oracles** of the agent, and those parameters it can change as **handles**. The possible actions of an agent are recorded in its protocol. Each action is specified as a guarded sequence of redefinitions of variables.

Observations in a real-world system are often related in such a way that (for the purpose of a particular modelling exercise) they are indivisibly coupled when they change. For instance, in the sail boat, the angle between the sail and the wind is defined in such a way that it changes instantaneously according to the sail orientation and the wind direction. Observations whose value is determined in this way are recorded as **derivates** in the LSD specification.

We associate the existence of each observation with the presence of an agent. For instance, driving_force and sail_area are attributes whose very existence presumes the presence of a sail. In LSD, an observation that is bound to an agent in this way is designated as a **state** variable of the agent. Each observation is associated with exactly one state variable instance that can be regarded as holding its authentic value. It is to the value of this variable that other agents refer in connection with stimulus and response.

1.3.2. Simulation of Behaviour from an LSD Specification

An LSD specification cannot be interpreted directly as a prescription for system behaviour. In engineering terms, we may relate this to the Frame Problem: the effects of agent actions can only be determined with reference to assumptions about the context for action. For instance, the interaction between the sail, rig and hull of the sail boat depends upon all kinds of implicit assumptions about the environment in which they operate and the reliability of stimulus-response patterns.

In animation from an LSD specification, we model the system state using "definitive scripts" [BBY92, BY90]. A definitive script comprises a family of variables, each of which has a value defined by a formula to be interpreted in the same way as the defining formula of a spreadsheet cell. When the value of a variable in a definitive script is redefined, all the variables whose values depend upon it are updated automatically. In modelling a concurrent system, this mechanism serves to represent an atomic transition of the system involving a synchronised pattern of changes to many observations.

Our development method exploits the unusual characteristics of definitive scripts as a medium for representing system behaviour. A definitive script can be used either to represent a system that has no autonomous behaviour, or one whose autonomous behaviour has been suspended or eliminated. The modeller can interact with such a script by redefining a variable in order to:

- observe the effect on the values of other variables (as in the *use* of a spreadsheet);
- represent a new perceived relationship between observations (as in *setting up* a spreadsheet);
- simulate the effect of agent actions (as in debugging the execution of a program).

By introducing programmed actions in the form of guarded sequences of redefinitions, the modeller can specify autonomous behaviour by agents, whilst retaining the power to modify or override their actions when required. A model of this nature will be described as a *definitive environment.*

1.3.3. The Computer Model

The overall aim of the modelling method is the development of an LSD specification and an associated computer model in the form of a definitive environment. The LSD specification is non-executable, but documents the real-world analysis involved in the construction of the computer model. The extent to which the computer model has an autonomous behaviour depends upon what assumptions we make about the stimulus-response behaviour of agents. Typically, completely autonomous behaviour of the system is only required when a particular scenario is to be analysed. In the SBS, much of the agent behaviour is concerned with communication of forces between components, and it is appropriate to model this as autonomous response. The sailor's actions cannot be predicted in this way, however, so that the role of the sailor in the simulation is played by the modeller.

Interaction between the modeller and the system is most significant during the development of a computer model. In this process, it is essential for the modeller to consider fragments of the system behaviour in isolation, and to verify that the stimulus-response behaviour of components has been appropriately captured. This activity is analogous to physical experiment on components of a system, such as an engineer might perform in the design process.

Because of the freedom with which we can interact with our computer model, it is inappropriate to regard it as *specifying the system behaviour*. In many respects, it is similar to the real-world system it represents, in that its behaviour can be explored in different scenarios, but cannot be circumscribed. It is on this basis, so as to emphasise the distinctive nature of our model, that we refer to it as a definitive *environment*. In passing from an LSD specification to a definitive environment, we introduce additional definitions that reflect reasonable generic assumptions about the scenarios of interest. For instance, it is appropriate that the image of the sail boat is indivisibly linked to its physical location, though in point of fact this presumes that it is not being viewed from absurdly far away (cf [FBY93]).

2. About the Modelling Process

The modelling process involves many applications of an iterative process, to be called a *construction*. Each construction takes place in the context of a particular *mode of observation*. For instance, in the SBS, one mode of observation is concerned with describing the geometry of the components of the boat, so as to enable simple visualisation (as in the bird's-eye and front projections of the sail boat depicted in Figure 2). Another is concerned with observing the forces acting upon the sail boat according to the speed of the boat, and the orientation of the boat and sail (as represented in the frame on the left in Figure 2, in which the arrows represent the relative wind direction, the driving force on the sail and the drag on the hull).

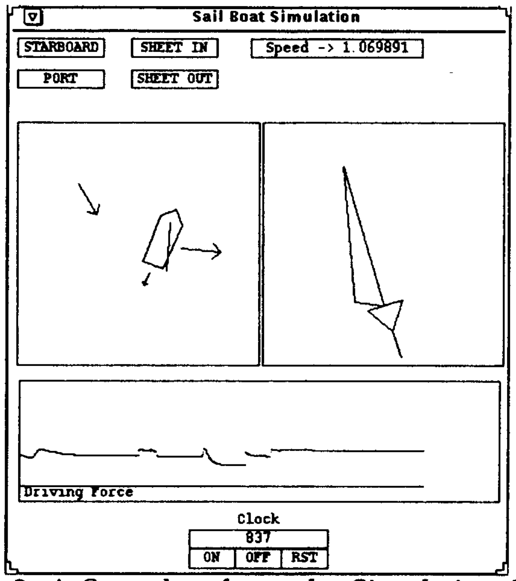

Figure 2: A Snapshot from the Simulation Screen

A construction involves iteration through three phases of activity:

- analysis of the real-world system, leading to the enhancement of an LSD specification;
- complementary synthesis of a computer model, leading to the enhancement of a definitive environment;
- evaluation of the computer model.

Each iteration is based on the principles for system analysis and computer model synthesis outlined in §1.3. Its function is to identify a family of observations and agents in the real-world system, and to construct a computer model that represents their inter-relationship as faithfully as possible. Iteration is necessary in general because some experimentation is required to generate an appropriate computer model. To this end, during the evaluation phase of a construction, we typically simulate familiar scenarios within our definitive environment and see if the results confirm our expectations. Discrepancies encountered in evaluating our computer model may reflect errors in the analysis or synthesis aspects of the construction process. Examples of such errors that arose in developing the sail-boat simulation are:

- overlooking the need to compute sail dynamics with reference to the relative rather than the absolute wind velocity;
- incorrect formulation of the trigonometrical relationship between the length of the sheet and the orientation of the sail.

One virtue of our method is that, through animation, we are able to uncover mistakes of this nature that might easily pass unnoticed in a paper specification (cf Harel [H92]). We are also able to deal in a similar fashion with uncertain information that we can only determine empirically. For instance, in order to simulate heeling, we have to estimate the moment of inertia of a typical sailing boat. We can gain some guidance on this point by carrying out sailing simulations, and applying our knowledge of the handling characteristics of an actual sail boat.

In overall structure, the modelling process consists of constructions relative to each relevant mode of observation. The modeller has much discretion over the order in which constructions are performed, but is constrained by hierarchical relationships between modes of observation. In our development of the SBS, for example, a preliminary construction identifies the geometrical attributes of the sail, rig and hull agents, and leads to their visual representation in a definitive environment in which the developer can experiment with the geometry of the boat and the configurations of the rig. A subsequent construction introduces the dynamics of the sail-boat, relating the motion of the boat to the orientation of the sail with respect to the wind and hull.

The early stages of the modelling method, in which the principal agents and their state variables are identified, has the flavour of object-oriented design [B86, SM92], but there are significant differences. In simulation, one agent can directly manipulate a handle that is a state variable of another agent. In the modelling process, there is an even more significant distinction: each construction typically embellishes agents with new attributes, enriching the model of an object by taking account of another mode of observation. In this respect, our approach conforms to our intuition: a real-world object transcends any model circumscribed by a particular mode of observation.

An important feature of the modelling process is that the modeller has discretion over the point at which a construction process is terminated. It is possible to recover and resume previous constructions if required, in much the same way that we revisit an experimental framework in the light of new evidence or a new requirement in the process of engineering design. The power of our modelling technique stems from two complementary virtues:

- since definitive environments are readily integrated, environments for exploring complex system behaviours can be constructed from ones that represent simple experiments with components;
- when we have synthesised a definitive environment to explore a complex system behaviour, it is conceptually easy to retrieve from the environment those definitions and actions that represent experimental contexts associated with its development.

The revision of a construction within a hierarchy can be problematic. For instance, if the force of the wind on the sail is defined without reference to the area of the sail, changing the dimensions of the sail will invalidate the dynamic model. In general, derivates and definitive scripts provide a useful mechanism for adapting interfaces between agents to accommodate potential changes [BY90].

Constructions are not confined to modes of observation that embrace the whole system. When the essential characteristics of the agents have been constructed and visualised, there is a role for detailed experiments directed at particular components. In the SBS, the agent specifications were refined in the sequence: sail, rig, hull. This reflects the chain of causation in the system: the wind acts upon the sail, the force upon the sail is transmitted to the boat via the rig, and reactions to these forces are generated by the ballasted hull.

Figure 3 shows a test rig that was constructed to check the plausibility of different models of the sail dynamics. In practice such a test rig could be constructed by securing a boat so that it cannot move and measuring the forces acting upon it under different test conditions. The graphs in Figure 3 depict the way in which the driving force on the sail, and the propulsion and heeling forces on the boat vary with the sheet angle when other parameters are set to constant values. The projection of the boat is depicted with the sheet angle set to zero and the wind direction relative to the boat indicated by an arrow. The test rig is associated with a definitive environment in which the significant parameters that affect the sail dynamics in animation can be redefined. These include the sail area, the velocity of the boat and the velocity of the wind.

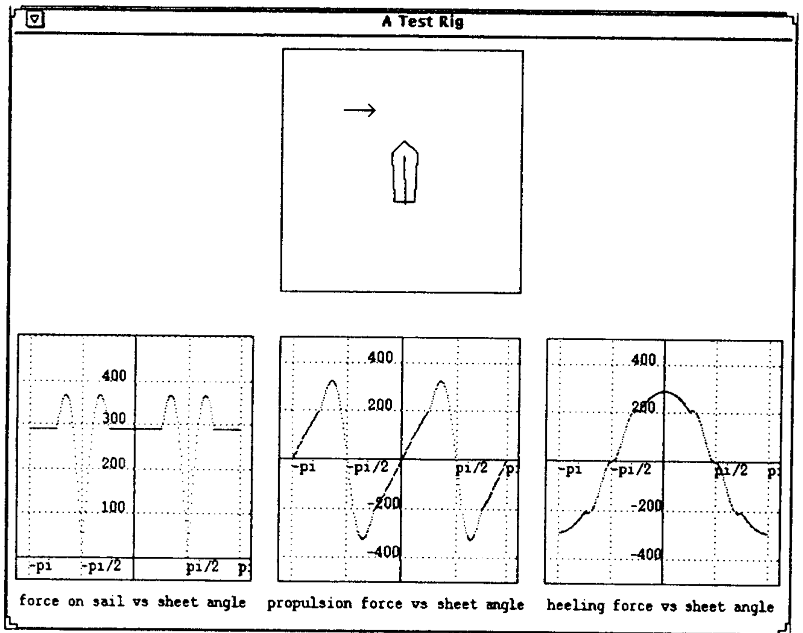

Figure 3: A Test Rig for analysing forces generated by the wind

By using the test rig, we have explored the possibility of explaining the driving force in Figure 3, which is based on empirical evidence, in terms of an interaction between simple pushing and suction effects. The visualisation illustrated in Figure 3, where the functional relationships between significant parameters are depicted in graphical form, is a more traditional application of spreadsheet principles that is oriented towards an engineer's rather than a sailor's viewpoint. Both Figure 2 and Figure 3 are examples of constructions within the modelling process, based on the applying the same principles, but illustrating different modes of observation. This illustrates how our method can integrate the views of many different human interpreters. In this respect, it performs a useful educational function.

In this connection, it is of particular interest that the model of the driving force in Figure 3 was originally devised by Ness on the basis of his sailing experience – only subsequently did we find independent evidence in [G63] for its validity. This was one reason for our interest in finding a satisfactory theoretical explanation for Ness's model, and indeed for first seeking a more plausible theoretically-inspired model for the driving force. This illustrates that, in developing insight into sail boat dynamics, there is a significant interplay between theoretical knowledge of physical laws, experimental evidence from constrained environments, and experience gained from practical sailing. Our modelling methods help to address the problems of integrating these perspectives, with a view to holistic understanding.

Simulating the role of the sailor agent in the SBS is a difficult issue. In our model, it is possible for the user to simulate manipulation of the sheet and rudder through a menu-driven interface. In adding this interface, we place an enormous restriction upon the functionality of the underlying definitive environment, confining the user to one or two simple modes of redefinition of variables. The effect of the rudder is simulated very crudely by directly redefining the orientation of the boat. This feature is included to illustrate how the models of components can co-exist within the SBS at quite different levels of abstraction. Though the simulation of sheet manipulation is more subtle, it is a poor analogue of the actual experience of manipulating a rope under tension.

3. The Nature of the Modelling Method

3.1. Computer Models of System Behaviour

A typical application of computer modelling in engineering involves the simulation of the behaviour of a system in a particular scenario. Many simulations are based on a theory about the system behaviour that supplies a mathematical model. In using the model, we extract information about the initial state of the system, then compute the expected behaviour. Two kinds of observation of real-world systems are implicitly involved in this process: the observations that informed the theory itself, and the particular observations that establish the initial conditions for its application.

The usefulness of computer models based on these principles depends crucially on the context in which they are invoked. How far the actual system behaviour conforms to the predictions of the computer model will be determined by how well the *frame of reference* presumed by the theory applies. This frame of reference is defined, largely implicitly, by the conventions for observation and interpretation of the system, and assumptions about their reliability. This relationship between the system behaviour and the computer model can be shown schematically thus:

within frame of reference:

$$\begin{tikzcd}
\text{system behaviour} \arrow[r, leftrightarrow] \arrow[d, dash] & \text{computer model} \arrow[d, dash] \\
\text{simulation of system behaviour} \arrow[r, leftrightarrow] & \text{execution of computer model}
\end{tikzcd}$$

In computer modelling of this nature, the frame of reference puts a straitjacket around the relationship between the computer model and the real-world system. The execution of the computer model can only be related to those observations of the system that are pre-determined by the frame of reference. The only knowledge about the system behaviour that informs the execution of the computer model is preconceived.

A computer model of a system behaviour resembles a conventional computer program. The primary focus of research on improved computer modelling techniques has been on making it easier to interpret an execution of a computer program as a system behaviour. Our approach addresses the issue in another way, constructing a computer model for a system viewed as an environment, rather than as a behaviour.

3.2. Computer Models of the System as Environment

Our common experience of real-world systems has an entirely different character from the computer simulation of a system behaviour. We are not restricted to a particular frame of reference; the range of observations connected with patterns of behaviour is not confined. All our actions are liable to lead to unexpected outcomes; the nature of the agents acting around us and their likely interaction is only partly known.

The mere *concept* of a system behaviour makes deep presumptions about the mode of observation and interpretation, and the assumptions needed to guarantee a particular system behaviour are unfathomable. When we set out to construct a computer model of a system behaviour we commit ourselves to all these implicit assumptions. The observations and agents we consider are circumscribed by our frame of reference, and we are in no position to adapt our model incrementally if observations of the actual system take us by surprise.

In the construction of definitive environments, we are modelling modest and provisional assumptions about a real-world system that are consistent with many different behaviours. This is consonant with experience; we cannot easily predict the behaviour of complex systems in the large, but have almost limitless faith in the reliability of commonplace observations. Many of the indivisible relationships represented in a definitive environment are patently beyond disbelief in just this way, and others are believable in a recognisable context. The sail boat always appears to be where its position can be independently reckoned to be; the angle of incidence of the wind on the sail is always defined by the difference in orientation between the wind and the sail; the force of the wind on the sail is an empirically determined function of the strength of the wind and the angle of incidence with the sail.

We can account for the behaviour of the sail boat far better in terms of a hierarchy of assumptions about reliable relationships between observations than by invoking a global theory. By exploring the grounds for our assumptions through experiment, we are also better able to appreciate under what circumstances we are *unable* to account for it.

3.3. Situated Modelling

The construction of definitive environments is a process that presumes the presence of an independent real-world system, albeit one that sometimes exists only in the modeller's imagination. The term *situated modelling* is introduced to convey the idea that the modelling process takes place in relation to a system as an environment. To appreciate this concept, consider what possible significance could be attached to a definitive script in which the variable names and definitions were chosen at random with no concern for possible interpretation: by what criterion could we possibly decide what mode of redefinition was appropriate?

It is for this reason that definitive environments are meaningful only in relation to an external system, and that the variables in definitive environments must correspond to observations. This context for modelling is quite different from the frame of reference required for mathematical modelling. Experience of a system as an environment does not supply a fixed frame of reference. To appreciate this, consider a definitive script that has a real-world interpretation, and consider by what criterion we possibly decide what mode of redefinition was *inappropriate.*

In situated modelling, the relationship between a system behaviour and the computer model can be shown schematically thus:

$$\begin{tikzcd} \text{system as environment} \arrow[r, leftrightarrow] & \text{computer model} \end{tikzcd}$$

within frame of reference:

$$\begin{tikzcd} \text{simulation of system behaviour} \arrow[r, leftrightarrow] & \text{execution of computer model} \end{tikzcd}$$

Only within a frame of reference is it appropriate to associate a behaviour with a definitive environment. The role of our modelling process is to provide the framework for experiments that can identify the frames of reference required for a theory about behaviour.

4. Concluding remarks

4.1. Background to the Paper

The chosen case-study and the general theme of this paper are based upon an assignment carried out by Ness in connection with a short postgraduate course on concurrent systems modelling given by Beynon. The first prototype was developed by Ness after a short period of familiarisation and practical experience with the relevant ideas and software tools. All subsequent development of the simulation is due to Yung, and the text of this paper is due to Beynon. A second paper developed in a similar fashion is also included in these proceedings [FBY93].

5.2. Issues for the Research

The features of our modelling method can be summarised as follows:

- states and views of real-world systems can be interpreted in our computer model;
- observation of the system can inform the model at every step of the development;
- we can develop our model incrementally and interactively;
- it is possible to simulate system behaviour in such a way that the suspension of animation permits us to simulate interaction within the current system state.

These qualities are significant in addressing Brooks' concern for ways of dealing with the problems of complexity, conformity, changeability and invisibility associated with complex system design.

A degree of idealisation and imagination is needed to infer these qualities from our present practical perspective. There is a clear need to refine our support tools and techniques.

With respect to constructions in the modelling method, further work is needed to clarify the identification and classification of observations and agents. Though we do not believe that LSD can be a formal notation in the conventional sense (cf. [S87]), it has a central role in the development method and some guidelines for principled use are essential. Deficiencies in the LSD notation encourage the dangerous practice of hacking definitive environments without developing an associated LSD specification. More formal and efficient ways of storing and retrieving the constructions involved in the development of a model are required.

At a more mundane level, our software tools use a variety of different syntactic conventions, are not all developed to the same degree of reliability and sophistication, execute slowly, and generate only simple graphics. Many concepts that we can now specify satisfactorily in principle (such as analog variables, events and loci) as yet have no proper high-level support.

Acknowledgements

We are grateful to Valery Adzhiev, Alan Cartwright, Mike Joy and Steve Russ for useful discussions relating to this paper. We are pleased to acknowledge the financial support of the SERC under grant GR/J13458.

References

[BSG84] Booth, K.S., Schaeffer, J., Gentleman, W.M., *Anthropomorphic Programming*, Computer Science Dept Report CS-82-47, Univ. of Waterloo, Ontario, Feb. 1984

[B86] Booch, G., *Object-Oriented Development*, IEEE Trans Software Engineering, SE-12(2), 1986, 211-221

[Br87] Brooks, F.P., *No Silver Bullet: Essence and Accidents of Software Engineering*, Computer April 1987, 10-19

[BY90] Beynon, W.M., Yung. Y.P.,*Definitive Interfaces as a Visualisation Mechanism*, Proc. Graphics Interface '90, Canadian Inf. Proc. Soc. 1990, 285-292

[BBY92] Beynon, W.M., Bridge, I., Yung. Y.P., *Agent-oriented Modelling for a Vehicle Cruise Control System*, Proc ASME Conf ESDA '92, Istanbul, Turkey 1992, 159-165

[C86] Cameron, J. A., *An Overview of JSD*, IEEE Trans. on SE, 12(2), Feb. 1986, 222-240

[D88] Deutsch, M.S., *Focusing Real-Time Systems Analysis on User Operations*, IEEE Software, Sept 1988, 39-50

[FBY93] Farkas, M., Beynon, W.M., Yung, Y.P., *Agent-oriented Modelling for a Billiards Simulation* , ibid

[G63] *The Glénans Sailing Manual*, Adlard Coles Ltd, 1963

[H92] Harel, D., *Biting the silver bullet: towards a brighter future for system development*, IEEE Computer, Jan 1992

[KR66] Kilmister, C.W., and Reeve, J.E., *Rational Mechanics*, Longmans 1966

[M88] Macaulay, David, *The Way Things Work*, Dorling Kindersley, London 1988

[R83] Roberts, Nancy et al, *Introduction to Computer Simulation: A Systems Dynamics Modelling Approach*, Addison-Wesley 1983

[S87] Smith, B.C., *Two Lessons in Logic*, Comput. Intell. 3. 214-218 (1987)

[SM92] Schlaer, S., Mellor, S.J., *Object Lifecycles: Modeling the World in States*, Yourdon Press, Prentice Hall 1992

[T88] Tomiyama, T., *Object Oriented Programming Paradigm for Intelligent CAD Systems*, Proc Intelligent CAD Systems II, 1988, 3-16