An Agent-oriented Framework for Concurrent Engineering
(extended abstract)

V.D. Adzhiev[*], W.M. Beynon[†], A.J. Cartwright[‡], Y.P. Yung[†]

[*] Flight Research Institute, Zhukovsky, Russia
[†] Dept. of Computer Science, University of Warwick, Coventry CV4 7AL
[‡] Dept. of Engineering, University of Warwick, Coventry CV4 7AL

1. Overview

We propose a model for the concurrent engineering process based on an agent-oriented approach. Our modelling approach captures the relationships between the attributes, perceptions and actions of agents. Our model differs from traditional models in that it deals not only with the object being designed, but also with the interaction between the human design agents and its influence upon the development of an evolving prototype. Such a model can be applied even where the design process itself has a non-routine character, with incomplete task descriptions, non-deterministic solution paths and on-going human intervention. The design object can be seen as having a complex hierarchical structure, and each agent as having its own view of the object. We describe a multi-level procedure that specifies the way in which the design object is synthesised from agent views over time. In this process, the design object is developed in a bottom-up fashion through the coordination of activities of agents at every level. The complex activity of each agent is described as a sequence of elementary generic activities associated with observation and experiment. In computational terms, the effect of agent activity upon the design object is expressed through the use of scripts of definitions which can be operationally interpreted.

2. Motivation

There are many challenges for knowledge representation in the concurrent engineering process:
- representing many alternative views;
- synthesising knowledge of fundamentally different kinds;
- dealing effectively with concurrency, inconsistency and conflict;
- recording human decision-making and negotiation;
- expressing the concept of a consensus view.

Each design agent has a different perspective on the qualities of the evolving product; the source, nature and status of information about the evolving product differs from aspect to aspect and agent to agent; the simultaneous recommendations that different design agents make are not necessarily consistent; the process of arbitrating between proposals cannot be fully automated; the entire corpus of intermediate products of the design process must be interpreted in relation to the overall objectives and specification of the design task.

Conventional approaches to computer modelling are not well-suited to the needs of Concurrent Engineering. They are typically oriented towards modelling system state and behaviour within a framework of comprehensive and consistent knowledge. For this reason, they are unsuitable for representing views, and for representing insight that is acquired incrementally through a process of experiment. Such computer modelling techniques can only be effectively applied where the design process if so well-understood that there is scope for full automation. In this paper, we propose an alternative computer modelling paradigm that is well-oriented towards representing the interaction between human designers that is characteristic of Concurrent Engineering. Our approach is closer in spirit to "subject-oriented programming" [7] than to an object-oriented paradigm.

The paradigm we apply is based upon "agent-oriented modelling using systems of definitions or *definitive scripts*". (Throughout this paper, the term "definition" is used in a technical sense to refer to relationships between variables resembling definition of cells in a spreadsheet, and definitive to mean "definition-based".) The power of software tools based on this and similar paradigms is demonstrated in much previous work [3,4,5,9,10]. This paper focuses on giving a more precise specification of the Concurrent Engineering process that is associated with their use.

3. The Abstract Definitive Machine

The design activity we shall describe will be represented as a form of generalised computation, in which the design agents themselves are the processing elements of an abstract parallel machine architecture. This architecture is an elaboration of a model we have introduced previously [4]: the Abstract Definitive Machine (ADM). The ADM has found wide application in our work in connection with concurrent systems modelling. (See [4] for instance for an ADM animation of a railway station arrival and departure protocol.)

For our present purpose, the main characteristics of the ADM (cf Figure 1) are:

- computational state is represented by a definitive script – the **Definition Store D**;
- each atomic change of computational state is represented by a set of concurrently executed *actions*;
- the latent actions are recorded in an **Action Store A**;
- the contents of the D and A are determined by the set of currently instantiated *entities*, each of which is specified by a generic set of definitions and actions;
- an action consists of a guarded sequence of primitive actions, each of which redefines a variable in D or leads to the instantiation or deletion of an entity.

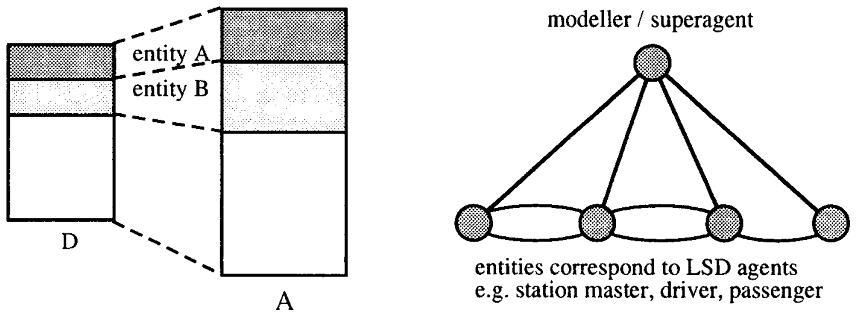
entity A
entity B
D
A
modeller / superagent
entities correspond to LSD agents
e.g. station master, driver, passenger

Figure 1: The ADM Machine Model

The use of ADM entities as models of human agents is well-represented in our previous work on concurrent systems modelling. For this purpose, we have developed a special-purpose notation LSD for specifying the observations through which the agents in a concurrent system communicate. (See the railway animation in [4] for instance, where the roles of the guard, stationmaster, driver and passengers are first specified in LSD, and then animated in the ADM.) In deriving an ADM simulation of a concurrent system from LSD specifications of its constituent agents, we impose a regime for the execution of each agent's privileges to change system state in response to changes in its environment. This in effect substitutes automatic stimulus-response reactions, typically guided by an appropriate scenario (cf [6]), for acts of free will on the part of autonomous human agents.

4. Virtues of the ADM model

The ADM is an unusual machine model that is well-suited for addressing the knowledge representation issues cited in §2. In simulating a complex system, the modeller interprets the execution of the machine with reference to the current state of the definition store D. The virtues of definitive scripts as a means of representing agent views have been demonstrated in our previous work [1,5]. The variables in such a script correspond directly to the observables that characterise the behaviour of the system being simulated. For instance, in the railway protocol animation, the observables include the current open/closed status of the train doors, the time on the stationmaster's watch, and the positions and destinations of the passengers. The definitive script reflects the way in which the values of observables are synchronised in change, so that (for instance) as the passenger steps onto the train, so the location of the passenger is indivisibly linked to the train location.

The model generated by the ADM uses metaphors to represent the current status of all observables in the system. The graphical depiction of the current positions of the train, the passengers and the guard etc. is one metaphor; an alternative visualisation might record only the current status of the interaction protocol, indicating the current status of the system with reference to critical stages in the protocol, such as whether the guard has yet raised the flag. Alternative metaphors make it possible to record the current state of the system as viewed by many different observers. Definitive scripts provide a powerful means to synchronise such alternative views.

Because the model is based upon faithful representation of the observed states of a system, rather than precise specification of its comprehensive behaviour, it is well-adapted for developing knowledge about the system by experiment. This distinguishes our method from logic-based methods of knowledge representation, and offers particular merits in conceptual design [2].

In the ADM, the modeller has exceptional privileges to intervene and reprogram dynamically. For instance, in the railway protocol animation, where the actions of two human agents in the simulation lead to conflict (as when a passenger opens a door at the same instant that the stationmaster closes it), the conflict is detected by the

ADM as simultaneous inconsistent redefinition. The modeller can act as a superagent to resolve such conflicts as they arise (see Figure 1).

The above synopsis of the ADM indicates that modelling using the ADM can readily be interpreted as explicit coordination of the interaction of a system of interacting agents. To adapt the model for this purpose, it is enough to introduce explicit instantiation of agents and authorisation of actions by the modeller, acting as superagent. In this way, the ADM is a natural context in which to express the activities of the design participants in our Concurrent Engineering process, whose task includes the coordination of agent activities through role allocation, scheduling, synchronisation and conflict resolution (cf Figure 2).

5. A Computational Model for the Concurrent Engineering Process

Our conceptual framework for Concurrent Engineering centres on the collaborative construction of a computer model of the design product – an evolving *virtual prototype* [1]. Informally, the virtual prototype represents the consensus of all members of the design team in respect of the current status of the product being developed. Time is an essential element in describing this prototype, as the design process will typically involve stages after which a consensus has been reached on particular features of the final design. The virtual prototype must also be interpreted with reference to a requirements specification that informs design decisions. This framework is more general than it may at first appear; we are not necessarily concerned with a design process in which the design stages and requirements specification are preconceived – both can be dynamically created and modified in the course of the design process.

At any stage in the design process, the current status of the virtual prototype can be understood with reference to a hierarchy of abstract agents such as is depicted in Figure 2. At the top of this hierarchy, there are two agents, the Principal Manager (PM) and the Client (C) who together negotiate over proposed designs to meet an evolving specification in a particular Design Context (DC). The DC is an abstract agent that represents those factors that influence the interaction between PM and C where there is conflict between a proposed design and the current specification. Such factors include the constraints that physical laws impose, and the restrictions placed upon the environment in which the product of the design process is to operate. The PM is in overall charge of the design process, and takes responsibility for specifying the objectives and format for the concurrent engineering activity involved at each design stage. The functions of the PM include decomposition of the principal design task into stages, and scheduling and synchronisation of the work of design agents at lower level in the hierarchy. Note that not all the abstract agents in Figure 2 correspond to a single human agent – in general, several human agents may cooperate in fulfilling the role of the PM for instance. It may also be the case that a single human agent acts in many abstract roles.

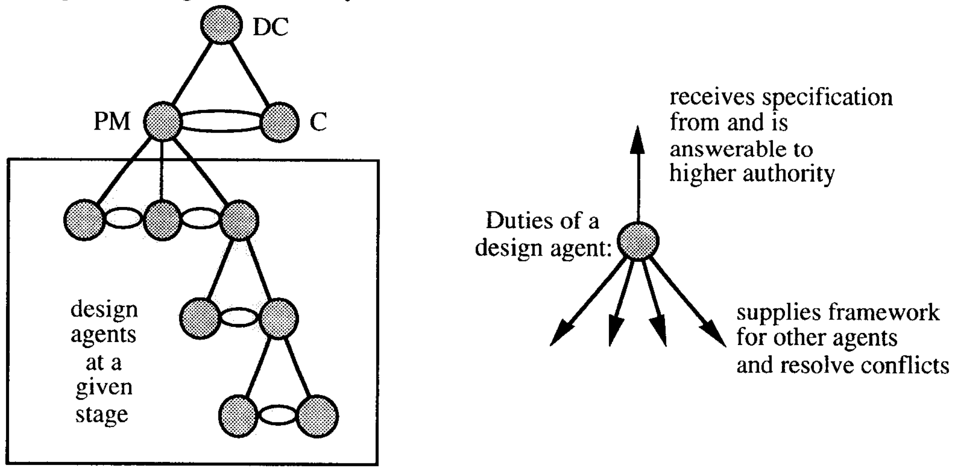
DC
PM
C
design agents at a given stage
receives specification from and is answerable to higher authority
Duties of a design agent:
supplies framework for other agents and resolve conflicts

Figure 2: The agent hierarchy and the design participant view

The virtual prototype is constructed by associating with each agent an appropriate "design frame" that represents the agent's view of the current state of the prototype. The design frame is a computer model of some aspect of the evolving product that is to be refined by the agent during the design process. The concept of design frame resembles the notion of frame introduced by Minsky, and the roles of the design agents can be conveniently expressed in terms of primitive operations on frames associated with Expectation, Elaboration, Alteration, Novelty and Learning (cf Minsky [8]). Each design frame is itself specified as an agent-oriented model within the ADM framework, and the iterative refinement of a design frame is a form of "situated modelling" characteristic of this modelling paradigm [3].

The design frame associated with the Client is the Specification for the final product of the design process; its principal contents will typically be design constraints and monitors such as we have discussed elsewhere [5]. The design frame for the PM represents the consensus view of the current status of the design. The design frame for a design agent involved in the model-building will include a definitive script that incorporates significant variables whose values must be defined by the end of the current stage of the design process. (The term "variable" is used here in a broad sense to embrace any suitable representation for an observable – this includes elementary numerical parameters, textual annotations, visual images etc.)

The primary activity in the design process is experimentation by the design agents that is aimed at refining their design frames. At any stage of the design process, many experiments will be in progress simultaneously, and the current state of the virtual prototype will be identified with the associated collection of extant design frames. The hierarchical framework for the design process expresses the integrity of the intermediate products of these experiments, which may encompass the portfolios of many design agents.

At the upper levels of the agent hierarchy, the experimental activity has a managerial aspect that involves establishing the objectives and patterns of interaction between design agents at the next level of abstraction, coordination of their activity through schedules, milestones, maintenance of design histories, and conflict resolution. A close connection between activity of this nature and modelling in the ADM framework is evident. At the leaves of the agent hierarchy, the experimental activity is engineering-oriented, and involves specialist skills and knowledge of particular aspects of the overall design task. The support that definitive script representations – cf spreadsheets – provide for this experimental ("what if?") activity is well-documented elsewhere ([2,3,4,5]).

6. Conclusion

The qualities of the ADM as an abstract machine model for modelling and simulation in design has been informally demonstrated in much previous work. This paper indicates how this machine model can be generalised, by introducing a hierarchical control structure, and more sophisticated control strategies, so as to provide a powerful agent-oriented framework for Concurrent Engineering.

7. Acknowledgments

We are indebted to the EPSRC for support under grant number GR/J13458, and to the Royal Society for sponsorship of Valery Adzhiev under its Visiting Postdoctoral Research Felowship scheme.

8. References

1. Adzhiev, V.D., Beynon, W.M., Cartwright, A.J., Yung, Y.P., *A Computational Model for Multi-agent Interaction in Concurrent Engineering*, Proc. CEEDA'94, Univ. of Brighton, 1994, 227-232
2. Adzhiev, V.D., Beynon, W.M., Cartwright, A.J., Yung, Y.P., *A New Computer-Based Tool for Conceptual Design*, Proc. Workshop Computer Tools for Conceptual Design, Univ. of Lancaster, 1994, 171-188
3. Beynon, W.M., Ness, P.E., Yung, Y.P., *Applying Agent-oriented Design to a Sail Boat Simulation*, Proc. ESDA 1994, London July 1994, to appear
4. Beynon, W.M., Yung, Y.P., *Agent-oriented Modelling for Discrete-Event Systems*, IEE Colloquium on "Discrete-Event Dynamic Systems", Digest #1992/138, June 1992
5. Cartwright, A.J., Beynon, W.M., *Enhancing Interaction in Computer-Aided Design*, Proc. "Design and Automation" Conference, Hong Kong, 1992
6. Deutsch, M.S., *Focusing Real-Time Systems Analysis on User Operations*, IEEE Sofware 1988, 39-50
7. Harrison, W., Ossher, H., *Subject-oriented Programming (A Critique of Pure Objects)*, OOPSLA'93, 411-428
8. Minsky, M., *A Framework for Representing Knowledge* in Mind Design, Philosophy, Psychology and AI, ed. Haughland, J., MIT Press 1981, 95-128
9. Wyvill, B., *An Interactive Graphics Language*, PhD Thesis, University of Bradford, 1975
10. Zarmer, C., Nardi, B.A., Johnson, J., Miller, J. *ACE: Zen and the art of application building*, Proc. 25th Hawaii International Conference on System Sciences, Koloa, HI, Vol. 2, 687-698, 1992