A Computational Model for Multi-Agent Interaction in Concurrent Engineering

Valery Adzhiev*, Meurig Beynon*, Alan Cartwright†, Yun Pui Yung*

Department of Computer Science*
Department of Engineering†
University of Warwick, Coventry CV4 7AL

ABSTRACT

Concurrent Engineering depends in a profound way on the use of computers. In this paper, we shall argue that we can best enhance the concurrent engineering process by improving the computational models that we use. We propose a multi-agent computational model to describe the parties involved in a concurrent engineering process.

Our approach involves identifying a hierarchy in which the function of a design agent at one level of abstraction is served by an agent who coordinates the activities of design agents at the next level. The concurrent engineering process is interpreted as the complementary construction of a computational object – the *virtual prototype* – that can be examined, used and altered by all parties. We explain and illustrate our methods and tools through a case-study based on the design of a machine tool drive shaft.

INTRODUCTION

The management of design is often based on specialisation – product generation tasks are grouped so that the responsibility for one function is with one particular phase of many products. In the past, each function was dealt with independently, with little interaction with other functions; consultative documents provided inputs and outputs to and from the other functions. Feedback in the form of negative comment from other functions led to a long time-span from conception to production.

Concurrent engineering is an attempt to get the best of both specialisation and integration. Design teams accept joint responsibility for a single product. This makes use of specialist skills whilst promoting feedback through iteration rather than criticism. For concurrency to operate each member of the team has to have up-to-date information about the state of the design. This necessitates many meetings and multiple copies of documents pertaining to the design. Design management in such circumstances becomes a significant problem because of the potentially damaging prospect of copies not being in step with one another. Voss (Voss 1989), in his investigation of organisational links for CAD/CAM implementation, suggested some remedial steps. It is essential to have a database that is shared by all functions. Putting different groups of people in physical proximity also helps. However, the most significant feature identified by Voss in his empirical study of a motor vehicle company was the need for a coordinator with an overall knowledge of the CAD system and with multiple functional and integrating skills.

In this paper, we introduce a agent-oriented framework for the design process. In management terms, an agent is a person whose responsibilities can be met largely independently of other tasks in the overall process. At the highest level of abstraction, the members of the design team are the agents. We introduce a design coordinator as a special member of the design team, whose responsibilities include strategic decisions concerning management of the interaction between design agents, specification of milestones for the design process, and identification of the evolving design constraints.

In our approach to concurrent engineering, the design process involves the construction of a computational object – the *virtual prototype* – which lies at the focus of the design activity of all participants. In this context, our interpretation of 'virtual' is not to be confused with its common usage in "virtual reality". The virtual prototype is synthesised from constituent views (*constituents*) associated with highly specialised roles within the design process. Each constituent is an environment within which to simulate experiments and observations appropriate to that role, and the metaphors used in simulation are not concerned with realism in the colloquial sense. For instance, a circuit diagram is a better metaphor for an electrical component than a photographic image of its wiring.

The design coordinator's brief is to oversee the synthesis of the virtual prototype from its constituents. In our framework, this activity can be interpreted as directing the design process according to a schedule by dictating the pattern of interaction between design agents and arbitrating amongst conflicting recommendations.

Constructing a virtual prototype makes exceptional demands on the computational model. The exploratory nature of the design activity demands support for *what-if* analysis. There are data representations peculiar to each specialisation and different metaphors are needed to represent them to the human interpreter. Intelligible interfaces between design agents are required, dependencies amongst views have to be maintained and incomplete specifications have to be accommodated. The design coordinator must also have the means to specify where and when design agents are privileged to make changes, and be able to provide appropriate interfaces for interaction for non-specialist computer users.

The state representation methods we have been developing over recent years specifically address these issues. They have already been successfully used to construct one constituent of the virtual prototype to be discussed in this paper (Beynon and Cartwright 1992), and to simulate a variety of physical processes, including a vehicle cruise controller, the operation of a sailboat, and the arrival and departure protocols at a railway station (Beynon et al. 1992; Ness et al. 1994, Beynon and Yung 1992). The key concepts in our research are: the use of a *definitive script* to represent the state of a complex system, and the use of agent-oriented methods of specification and animation. Details of these techniques will be described and illustrated in connection with a particular case-study in engineering design.

CONSTRUCTING THE VIRTUAL PROTOTYPE

We illustrate the principles of virtual prototyping in our framework with reference to a case-study: the design of a shaft such as the main spindle of a lathe. We have already applied our methods to the analysis of the stress/strain regimes in a stepped shaft (Beynon and Cartwright 1989). We now consider the design of a lathe spindle in a broader context.

We identify three agents in the design process: the Analyst, responsible for analysis of the shaft design, the Detailer, for the choice of standardised and available elements, and the Manufacturer, for appropriate geometry and material for ease of manufacture. Each agent has tasks within the design brief, as determined by the Design Coordinator for the shaft design. In the choice of design parameters, each agent has its own repertoire methods of making pertinent observations, as set out in Table 1. Design decisions are based on specialist knowledge and experience, much of which is directly or indirectly associated with experiment and observation. The virtual prototype is intended to serve as a computer environment within which such experiments can be simulated, and their implications recorded.

	Agent		
	Analyst	*Detailer*	*Manufacturer*
Design Parameter	*Typical observation procedure*		
min. shaft diameter	measure loads and compute	get nearest standard diameter	measure with micrometer
step sizes, positions	consult bearing, gear sizes	check avail. bearing, gear sizes	machine tool set up time and cost
static and dynamic characteristics	measure loads. mass, geometry: compute		
material	compute stress conditions	consult catalogue	heat treatment process & measure
spindle operation	simulate use. study cutting dynamics		
manufacturing cost		get quotation from contractor	investigate cost of each manuf. process
material cost		get catalogue cost	actual cost
assemblability	assign tolerances	check standard components fit	measure dimensions

Table 1

In abstracting the design of the lathe spindle from the total lathe design, we have already reduced the scope of the concurrency problem. The nature of the management function within each specialism is still too complex for us to be able to describe a suitable virtual prototype directly. For instance, the work of the Analyst involves two different kinds of observation of the shaft – one concerned with static properties of the shaft under load, the other with analysis of the shaft dynamics in suitable scenarios for operation. We address this problem through a recursive subdivision of the design task.

Each agent typically adopts many different perspectives on the design task in arriving at a judgement. These perspectives resemble different modes of observation of a physical system (Bohr 1961) – they are derived from essentially independent experiments and cannot necessarily be integrated into a unifying view. A good Analyst will be aware that the design decisions suggested by a static and dynamic analysis can conflict, and is skilled in finding appropriate compromises. In constructing the virtual prototype, we model this by treating the Analyst as coordinating work carried out under two separate roles, that of Static and Dynamic Analyst. The principle behind this decomposition of complex agents is similar to that advocated by Minsky in his Society of Mind (Minsky 1988). The hierarchical framework for the conceurrent engineering task developed in this way is represented in Figure 1.

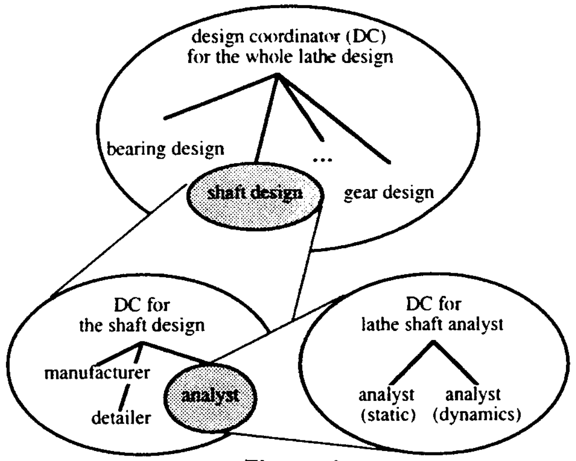

Figure 1

In this paper, we describe how we can exploit the decomposition of the design agents in the synthesis of the virtual prototype. To this end, we first review the methods we have already applied in constructing virtual prototypes to meet the needs of the primitive design agents. We then discuss how we propose to extend these methods to construct virtual prototypes for more complex design agents – for instance, to synthesise a virtual prototype for the Analyst from constituent environments appropriate for the Static and Dynamic Analyst. By a recursive application of this process, we develop a bottom-up specification for a comprehensive virtual prototype that complements the top-down decomposition of the design agents.

CONSTITUENTS OF THE VIRTUAL PROTOTYPE

An agent-oriented model for the virtual prototype is constructed from relatively simple constituents whose form is abstractly represented in Figure 2. Each constituent is associated with a primitive design agent – represented by a leaf in the agent hierarchy depicted in Figure 1, and by the human agent in Figure 2 – whose role is defined by a highly specialised mode of observation and experimentation. The environment for static analysis of the lathe shaft described in (Beynon & Cartwright 1989) is an example of one of these constituents. Other papers (Beynon et al. 1992; Ness et al. 1994) illustrate how the same principles can be used to construct the simulation of the lathe shaft in operation that is required in dynamic analysis.

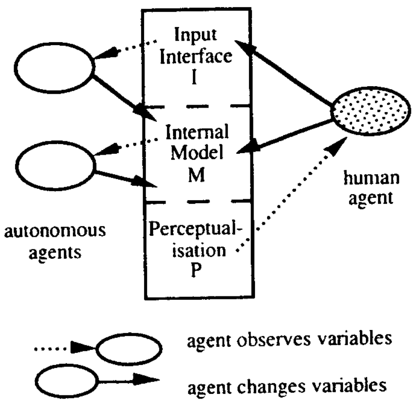

Figure 2

In Figure 2, the central rectangle represents a definitive script that has conceptually three components, denoted by P, M and I respectively: variables concerned with visualisation (or in general *perceptualisation*), those concerned with maintaining a faithful internal model of external values such as would be involved in experiments on a physical prototype, and those whose values can be assigned through direct manipulation of the interface. Note that changes in values may propagate between variables in different categories via definitions. For instance, the visualisation of a geometric element will typically be indivisibly related to its coordinates as stored in the internal model.

The different categories of variables P, M and I can be illustrated with reference to the Vehicle Cruise Control simulation described in (Beynon et al. 1992). A typical variable in M represents a parameter of the dynamic model, such as the acceleration of the vehicle, a variable in P represents a display element, such as the line depicting the magnitude and direction of the wind force, and a typical variables in I would record the status and location of the mouse, as required for menu button selection.

The circle to the right represents an archetypal human agent interacting with the computer by redefining variables in the script. Such an agent can respond to the perceived state of the model by directly redefining a variable in M or by perturbing a value in I. The circles to the left represent archetypal autonomous agents, that are typically pre-programmed by the user (or by a design agent at a higher-level in the hierarchy) and can serve a variety of functions. One such agent might redefine a variable in M in response to a change of an interface variable – as in a programmed response to the selection of a button. Another might react to changes in the value of variables in one part of the internal model by redefining variables elsewhere in the model, as is required for instance when undoing definitions that violate constraints, and when using analogue variables to simulate dynamic processes (Beynon et al. 1992).

The values of the perceptualisation variables in the script faithfully reflect the current state of the shaft as viewed from the perspective of the design agent. The relationship between the values of variables in the definitive script and these real-world observations is defined by a different metaphor according to which agent is involved, and what kind of experimental context is being simulated. Using our present tools, the state of a real-world system can be represented – in a way that respects collections of observations that change indivisibly in combination – by displays that comprise text, line-drawings and geometric models. For this purpose, we have developed suitable *definitive notations* within which to formulate appropriate scripts. By means of such notations, the technical issues involved in specifying complex displays are hidden from the user, who is free to concentrate on the analysis of observations in the real-world interpretation. In this respect, the formulation of scripts is well-suited for use by a specialist who has a thorough understanding of the real-world application and the metaphor used in the model, but will not typically have expert programming skills. As our system matures, we expect to be able to extend the range of metaphors accessible to the naive user in this manner. For instance, we propose to introduce definitive notations that hide the implementation details presently involved in formulating relationships between analogue variables, and in describing the graphs of real-valued functions.

AGENT INTERACTION WITHIN CONSTITUENTS

The directed edges in Figure 2 represent the stimulus-response patterns that underlie the behaviour of agents. Those on the right-hand side reflect the manner in which a human agent redefines parameters in the model in response to the perceived state of the virtual prototype. The scope for redefinition is restricted by the external semantics (e.g. the deflection of the shaft under load is determined by well-tested predictions of theory, and cannot be arbitrarily redefined), by intrinsic constraints (e.g. the choice of Young's modulus is restricted to a range determined by known materials) and by the context for the design agent's actions (e.g. the agent is carrying out static analysis of a shaft of specified dimensions). As the design process progresses, the privileges to redefine parameters may change. For instance, there may be a stage after which the choice of material is no longer in doubt.

In our approach, the agent privileges are specified in a special notation LSD. In an LSD specification, a variable that can be observed by an agent is identified as an **oracle** for the agent, and a variable whose value can be conditionally redefined is a **handle**. Each agent has certain privileges to change state: each privilege is represented by a guarded sequence of redefinitions of handle variables. The set of privileges of an agent specifies its **protocol**. In describing the behaviour of an agent, it is often necessary to represent values that are derived from others by fixed rules of interpretation or computation. A value of this nature is specified as a **derivate** for the agent. For instance, a derivate such as:

```
detailer.shaft_len_in_inches = sa.shaft_len_in_cm / 2.54
```

may represent the detailer's interpretation of the shaft analyst's measurement of the lathe drive shaft length.

The LSD specification of the agent's protocols places restrictions upon how changes of state within a constituent can occur, but does not of itself determine the behaviour of a constituent. Only when there are autonomous agents with a known pattern of behaviour that can be preconceived is it appropriate to animate them through systematic execution of their privileges. For instance, in simulating the operation of the lathe during dynamic analysis, we would follow the pattern of animation previously applied to other systems (cf (Beynon et al. 1992; Ness et al. 1994)), introducing a clock agent and computing the values of analogue variables, such as force, acceleration and velocity, with reference to it. Such agents are represented in Figure 2 by the directed edges on the left of the diagram. In general, it is impossible to predict the precise pattern by which other agents – such as the design agents – exercise their privileges to modify the virtual prototype.

An operational interpretation of an LSD specification can be developed using the Abstract Definitive Machine (ADM) (Beynon & Yung 1992). The key concept of the ADM is that the current state of a computation is represented by a definitive script, the latent transitions by a set of actions, and the pre-programmed agents by a set of entities. These are respectively associated with a definition store D, an action store A and an entity store E. An entity is a group of definitions and actions, and an action is a guarded sequences of primitive actions, where a primitive action is either a redefinition, or the instantiation or deletion of an entity instance. On each execution cycle of the ADM, the actions with true guards are executed in parallel.

As a computational model, the ADM has some useful characteristics:

- parallel redefinition. Several agent actions can be performed simultaneously.
- automatic detection of conflicts. When two agents simultaneously redefine the same variable, or redefine variables in a way that introduces cyclic dependency, the ADM will prompt the user to resolve the situation.
- exceptional privilege of the user. The ADM runs in an interactive mode: the user is able to perform actions as if in the role of an invisible and previously unspecified ADM entity. This makes it possible to redirect an ADM computation interactively.

These features are seen to their best advantage in the concurrent development of constituent views of the virtual prototype that is carried out under the guidance of the design coordinator.

COMBINING THE CONSTITUENTS

When combining constituent views, we can distinguish two different mechanisms for dealing with observations that refer to a common entity. Where there is consistency between such observations in two or more views, the relationship between them can be expressed by formulating suitable bridging definitions. In general, the consistency of observations is related to the level of coordination of agent activity, and can be most conveniently modelled as the inheritance of information from an agent at a higher level of abstraction in the design hierarchy. Consistency that is guaranteed by absolute constraints on the design process, such as compliance to physical laws, is a special case of such inheritance. Where there is a possibility of inconsistency beween observations common to two views, there is a design issue to be resolved, and interaction amongst agents has to be considered.

There are many patterns of interaction for the parties in the concurrent engineering process. In exploring a new design, the Analyst will typically investigate the static and dynamic characteristics of many shaft design variants. Design variants may be incomplete (e.g. a candidate design produced during static analysis may make no reference to the moment of inertia of the shaft) and may have observations in common (e.g. both static and dynamic analysis presumes knowledge of the gear locations on the shaft). When observations overlap, the choice of design parameters for static and dynamic analysis will not necessarily be consistent. In effect, the Analyst is switching between two roles until a suitable compromise between optimal parameters for the static and dynamic models is found. At any intermediate stage, there may be several candidate designs with good qualities in respect of one or other static or dynamic criteria.

The design activity of the Analyst, viewed as coordination of work done in the two roles of Static and Dynamic Analyst, resembles the whole design process in microcosm. It is characteristic of the design process that many variants are developed by independent parties, that some of these remain incompletely specified and that the coherence of independent design proposals can only be guaranteed through a coordination process.

The LSD specification below expresses the way in which a design coordinator can exercise control over the patterns of interaction between agents. In the specification, the agents own_work and listen_to_detailer represents two modes in which the static_analyst can operate: in the one case, exercising independent control over the choice of material, in the other, working with the material chosen by the detailer. The **state** variables in the specification designate observations that are bound to an agent, so that maxdefl_sa refers to the value of maximum deflection that the static_analyser has in mind. The variable pattern is a handle for the design_coordinator, by means of which a new pattern of communication can be imposed upon the detailer and the static-analyst.

```
agent design_coordinator {
state pattern
oracle material_sa, maxdefl_sa
oracle material_d, maxdefl_d
handle pattern, role_sa, role_d
protocol
    true -> pattern = FUNC(material_sa,
        maxdefl_sa, material_d,
        maxdefl_d, time_constraint, ... )
    ...
}
```

```
agent static_analyst {
state maxdefl_sa, material_sa
state role_sa
oracle pattern
derivate
     role_sa = (pattern == "independent") ?
               "independent"
          : (pattern == "detailer_determines_material") ?
               "listen_to_detailer"
          : ...
...
     agent own_work {
          handle material_sa
          derivate
               ACTIVE = role_sa == "independent"
               maxdefl_sa = f(material_sa)
          protocol
               true -> material_sa = FUNC_choose()
     }
     agent listen_to_detailer {
          oracle material_d, maxdefl_d
          derivate
               ACTIVE = role_sa == "listen_to_detailer"
               material_sa = material_d
               maxdefl_sa = maxdefl_d
     }
     ...
protocol
     suspend_work ^ role_sa == "independent" ->
          save(static_analyst, storename)
     role_sa == "independent" ->
          retrieve(static_analyst, storename)
}

agent detailer {
state maxdefl_d, material_d
state role_d
oracle pattern
derivate
     role_d = (pattern == "independent") ?
               "independent"
          : (pattern == "detailer_determines_material") ?
               "independent"
          : ...
...
}
```

Listing 1

When interpreting the LSD specification, it should be noted that the precise nature of the communication between agents is determined in detail only through animation in the ADM. In this process, there are many different possible conventions for implementing oracles, for instance. By a way of illustration, suppose that the detailer and the static analyst collaborate in such a way that the static analyst works with the material selected by the detailer. In one scenario, the analyst adopts the detailer's choice of material and proceeds to explore its implications through independent experiment. In another scenario, the detailer acts in the experimental environment of the static analyst, and can change the choice of material intercatively.

Listing 1 is a fragment that merely indicates the kind of mechanism that can be used to set up communication patterns in the design process. In general, there are many ways in which such mechanisms can operate within a broader framework for design management (cf (Sonnenwald 1994)). In developing the LSD specification in Listing 1, it is simplistic to treat the shaft analyst as if there were but one instance of the Static Analyst agent,whose activity is at all times directed towards a corporate goal under the direction of the design coordinator. In practice, the real design agents may be developing many tentative design fragments in parallel, and in some instances the design coordinator may be in a position to stipulate interaction between two specific instances of design activity on the part of the detailer and shaft analyst. A proper exploration of the issues raised by multiple versions of design and multiple instances of design agents is beyond the scope of this paper; we instead conclude our discussion by considering how the goal-directed public interaction between an official Shaft Analyst and Detailer guided by the Design Coordinator can fit into the framework of a formal design project.

A typical interaction between Static Analyst and Detailer concerned with the choice of material for the shaft. The Static Analyst seeks a material that best conforms to a particular strength and stiffness pattern; the Detailer the cheapest and most readily available standard material within the specified property range. Each agent has a valid justification for selecting a particular material, but their choices have different implications for the design. Within their constituent environments, the agents are free to explore variants of the current prototype, which in turn generate numerous scenarios for future development of the design. When a decision has to be made, the task of the design coordinator is to adjudicate on what constitutes the "current prototype" and what are the local variants. These choice-points can serve as the milestones of the design.

When a milestone is reached, the design_coordinator may restrict the patterns of work that are currently allowed, so placing constraints on the design interaction. At the same time, certain parameters may be fixed or confined to a limited range of values. Where relevant, such constraints on the values of variables can be imposed in constituent environments through the introduction of enforcement agents, as discussed above. Such specification of constituent environments is part of a broader aspect of the design coordinator's work – that of identifying modes of interaction that are most effective in achieving specific design goals, and customising the interfaces supplied to the design agents to suit their needs. For instance, an effective way to restrict access to variables in a definitive script is to introduce direct manipulation interfaces (such as the control panel for the vehicle cruise control simulation in (Beynon et al. 1992)) through which a circumscribed family of redefinitions can be performed.

As Figure 1 indicates, there are two perspectives on the work of the design coordinator. One is concerned with the management of subagents, the other with the generation of an acceptable design. The full LSD specification of the role of the shaft design coordinator must include a specification of the shaft itself as the subject of the concurrent engineering process. An outline LSD specification for the shaft has the form:

```
agent shaft
    state
        material, maxdefl.
        t_start. t_finish.
        [(stage_1, deadline_1), (stage_2. deadline_2). ... ]
        design_pattern,
        design_state
    ...
    agent shaft_sa {
    state
        material_sa.
        maxdefl_sa
    ...
    }
    ...
}
```

Listing 2

The state variables in this specification represent those attributes that are most significant for the current phase of the design – here presumed to include maxdefl and material. Other state variables reflect the stages and deadlines in the design timetable associated with the choice of these parameters. The entire phase is presumed to extend from time t_start to t_finish – within this period of time, there are intermediate stages. The variable design_state reflects the general status of the project, to be assessed by the shaft design coordinator at the end of each substage. The state variables in the shaft are handles for the shaft design coordinator, who has sole responsibility for specifying their values. In this context, the subagent shaft_sa within shaft represents the shaft analyst's official view of the current status of the shaft design.

CONCLUSION

Design is rapidly becoming computer-based rather than computer-aided, as the demands for greater complexity, optimisation and concurrency in the design process make computers indispensable. The work described in this paper is part of a long-term research programme aimed at developing a new mode of using computer systems in design that in itself encourages a proper understanding of the design process.

ACKNOWLEDGMENTS

We are grateful to the SERC for support under grant GR/J13458. We are also indebted to the Royal Society for supporting Valery Adzhiev under its Postdoctoral Research Fellowship Scheme.

REFERENCES

Bohr, N.. *Atomic Physics and Human Knowledge*, Science Editions Inc., New York 1961

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

Beynon, W.M.. Cartwright, A.J., *A definitive programming approach to the implementation of CAD software*, Intell. CAD Systems II: Implementation Issues, Springer Verlag 1989. 456-468

Beynon, W.M.. Cartwright. A.J. *Enhancing Interaction in Computer-Aided Design* Proc "Design and Automation" Conf., Hong Kong. August 1992

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

Hewitt, C. *Viewing Control Structures as Patterns of Passing Messages* Artificial Intelligence, 8. 323-364. 1977

Minsky, M. *The Society of Mind* Picador, London 1988

Ness, P., Beynon, W.M.. Yung, Y.P. *Agent-oriented Modelling for a Sail Boat Simulation.* Proc. ASME ESDA '94. London. *to appear*

Sonnenwald, D.. *A Descriptive Model of Communication among Developers*, Proc. PPIG'94 Workshop, The Open University, UK, January 1994. 37-48

Voss, C. *Integrating mechanisms: Empirical Observations in a UK Car Company*. Seminar on the Management of CAD. University of Warwick. October 1989