ENHANCING INTERACTION IN COMPUTER-AIDED CONCEPTUAL DESIGN

A. J. Cartwright* and W. M. Beynon†

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

ABSTRACT: Computer-aided techniques must support design as a process involving consultation between human agents and interaction with the world at the level of partial solutions. We discuss and illustrate the potential for using generalisations of spreadsheets as consultative documents in the design process. Our work involves the comprehensive adoption of systems of definitions as a representation for state in programming and design.

1. INTRODUCTION

Computer-aided design systems have led to the automation of many labour intensive tasks. Design productivity has been increased. The design process has also been enriched. Analysis is more rigorous and less error-prone, designs and product information are better presented and the design-manufacture gap is being reduced. The results are apparent in higher rates of innovation and reduced lead times in marketing new products.

These developments have put pressure on management to reappraise the role of design. Such reappraisal motivates concepts such as 'forward engineering' [1] – an activity based upon early manufacturing involvement, the management of technological uncertainty, quality function deployment, design for manufacture, and assembly and process control. Forward engineering demands cross-functional teams representing these specialisms as well as conventional engineering design. On this model, many more people have to participate in the early stages of product design.

Technical developments in CAD have reinforced the demand for wider participation. At one time, comprehension of technical drawings was a prerequisite for appreciating an engineering design. Even after the draughting process was automated, the interpretation of computer-aided drawings remained a task for the specialist. Only in recent years, when sophisticated computing resources for graphics have become more widely available and more powerful techniques for visualisation have been developed, has it become feasible to animate a design realistically at a very early stage. As a result, more people are now able to appreciate a designer's proposals; marketing managers, visual designers, service personnel, accountants – even conservators – all wish to interact with the design and express their view of the product requirement.

Enabling wide interaction in the design process poses a major technical challenge for CAD. Current CAD systems are well-adapted for constructing visual representations that the non-specialist can interpret – as when rendering techniques are used to create realistic images of objects from superficial and incomplete geometrical data. In this way, information about a design can be communicated more readily to the user. On the other hand, representing the design in the computer so as to accommodate feedback from human agents and potential conflict between alternative requirements is more problematic. Developing computer representations in a way that reflects their external meaning and relationship to real-world interpretation and function is a fundamental problem [2]. Perhaps in part for this reason, major influences on the development of CAD systems run counter to the trend towards greater human involvement in the design process. For example, much research is directed at coalescing the islands of automation downstream of the conceptual design so as to eliminate human intervention.

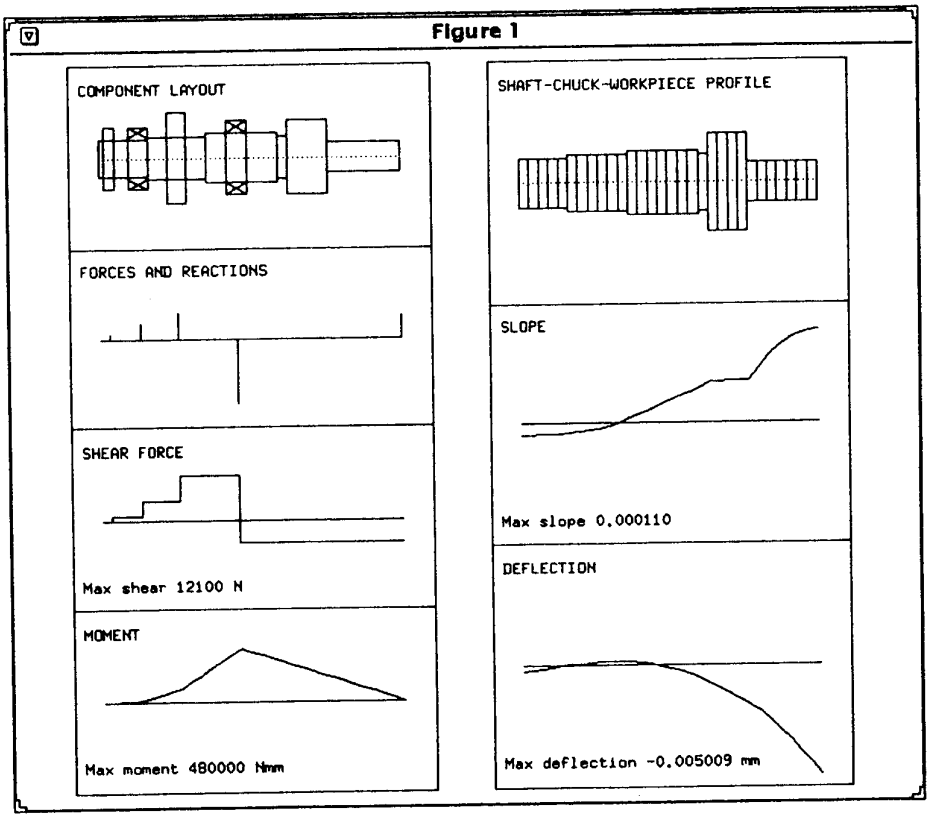
This paper sketches research aimed at enhancing the scope for interaction in conceptual design. Our objective is to provide computer support for the design and manufacturing process that enables analysis and documentation of the requirement in all its aspects. Drawing up this requirement involves consultation between many human agents representing all phases of the design and manufacture process. The primary

problem we address is that of representing the design so that this interaction can be carried out effectively. The computer use envisaged is different in kind from "automation of labour-intensive tasks", where computation is complex, but can be entirely preconceived. Our concern is to represent the incomplete design so that it can be appreciated in conceptual terms, communicated to different parties in the design process and readily modified to reflect their concerns (cf [3]).

Our focus is not upon design as a process moving serially from need to product. A more complex pattern of interaction is appropriate, such as corresponds to passing a consultative document around between human agents. In that context, the design activity passes through phases that are product-oriented: specifications and schemes are hard-copy outputs that become inputs for the next phase. Evaluation goes on in all phases of the design, often leading to iteration. Evaluation criteria must be available on the basis of incomplete designs and possibly also on incomplete data. There is a close parallel – and connection – between our model and models in which a human designer is seen as interacting with expert systems, but our emphasis is upon interaction with human agents, each of whom can bring specialist knowledge from the real world to bear.

2. A CASE STUDY

The design of the main drive shaft of a lathe is our case study. Figure 1 is a screen from our prototype design environment. It depicts the lathe shaft, showing the disposition of drives and bearings, a possible step profile for the shaft and graphs derived by analysis. The lathe specification lays down the



primary requirements for the main drive shaft, establishing relationships or constraints that dominate the design. The most important concerns are

- maximising the stiffness of the shaft, subject to size and cost, so as to minimise deflection at the tool point
- optimising the disposition of bearings and drive elements on the shaft.

In practical terms, the disposition of components is critical, affecting not only the shaft geometry but also stiffness at the tool point and manufacturing complexity. Typically the requirement also includes accommodation for various workholding devices for automatic or manual work handling and an internal bore to allow for bar feeding and long components.

In a typical configuration, the lathe shaft is stepped and tapered, and the internal bore may also be tapered. Loading occurs at the tool point and at the driving gear or pulley and is taken by two taper roller bearings. The bending model treats the workpiece and chuck as an integral part of the shaft.

From a manufacturing perspective, it is important to monitor aspects of the design that affect the cost and complexity of production. These include the number of steps on the shaft, the weight and cost of the blank required to generate the shaft and the amount of grinding required. The quality of the tool might also be assessed in terms of the rigidity of the shaft, and the estimated fatigue life for the shaft and bearings.

These concerns dictate the characteristics of a computer model to assess a particular design. In particular, we need

- to visualize the physical characteristics of the shaft in relation to typical workpieces, cutting speed and cutting force position,
- to monitor how the choice of particular parameters for the shaft influence the quality of the product and the cost of manufacture.

3. CHARACTERISTICS OF OUR APPROACH

One way to construct a computer model is to write an interactive program with appropriate functionality. The form of such a program will reflect assumptions about how different agents in the design process need to interact with the model. In practice, it will probably be biased towards specifying a coherent sequence of interactions for articulating the engineering design. An alternative approach is to describe a model declaratively, building a knowledge base of information about the intended design and developing automatic tools for reasoning with this knowledge. In principle, this allows a more flexible design strategy, but at the cost of making specification of the interaction harder. Our method combines the best features of both these approaches.

Whatever the nature of the computer model or the design process, certain information about the design object has to be represented and certain transformations performed. For instance, in our case study, it is necessary

- to devise a representation of the lathe shaft configuration from a set of parameters,
- to display engineering data for different configurations,
- to experiment with the geometry and disposition of the components on the shaft,
- to monitor the deflection as a function of estimated manufacturing cost,
- to simulate performance of the lathe in use.

We can simplify these tasks greatly by applying a method of representing knowledge that is characteristically associated with a "what if?" mode of analysis. A "what if?" scenario is defined by a state and a set of latent transformations of that state. The spreadsheet is a familiar model of this nature: it describes a particular state of a system in such a way that the relationships that persist between values in change are modelled as indivisible.

The "what if?" concept of knowledge involves state intrinsically. Spreadsheets use procedural variables that are assigned different values to reflect independent observations of the same quantity (cf a logical model of knowledge, which uses static mathematical variables [2]). The "what if?" activities upon which are models are based centre around data dependencies expressing our expectations about how the various ingredients of the lathe shaft model are interrelated. For instance, we model the fact that if the dimensions or location of the gears on the lathe are changed then this will affect the distribution of load and alter the engineering data being displayed. Other dependencies describe the intended relation between the screen display and the state of the engineering model: a textual string is a warning message when the deflection exceeds a critical value.

Modelling fundamental data dependencies enables us to represent information about the design object so that it can be appreciated through experiment. These dependencies reflect the essential nature of the object as we choose to observe it. Their representation does not commit us to a particular strategy for transforming or interacting with the design object; it models those aspects of the observed behaviour of the design object that we wish to take for granted. This makes it much easier to describe procedural activities associated with the design process, such as enhancing the design model, developing the design environment or simulating the design object.

The design process itself involves investigating ever more refined "what if?" scenarios. As we refine the model of the design object, we establish data dependencies that reflect more subjective analysis. Each "what if?" scenario entails

- determining the parameters to be varied,
- recording significant values of parameters for later re-use and comparison,
- monitoring critical features of the design to avoid the violation of constraints.

In developing a system for analysing the lathe shaft, "what if?" analysis can be applied in many different ways: to design the interface layout; to study the effects of load distribution on a uniform shaft; to assess the impact of introducing steps to the shaft. In our approach, refinement of the design model proceeds in parallel with the development of a design environment. As the environment develops, the scope for easy extension of the system is increased, so that the design process becomes progressively less constrained.

4. TECHNICAL ISSUES: AN OVERVIEW

Our approach to design is based on techniques for modelling and visualisation that have been used in a variety of applications. These techniques rely on using sets of definitions ("definitive scripts") to relate values of variables that specify various aspects of the current state of a computer model and the user environment. The principle is similar to that used in the spreadsheet, where the value of a cell may be implicitly defined in terms of the values of other cells, and these values have an external interpretation that determines the state of a model. The power of such representations derives from the close correspondence between values displayed on the computer screen, and meaningful quantities that the user can imagine manipulating and observing independent of the computer model.

Definitive scripts can be used to represent many aspects of the state of a model. For instance, the display in Figure 1 is described by a script that includes variables denoting the locations and characteristics of windows, the relation between points and lines in diagrams, textual annotations and scalar quantities. A language for formulating definitions over a particular underlying algebra of values and operators is a *definitive notation*. Several definitive notations are used in Figure 1; they include SCOUT (for SCreen layOUT) and DoNaLD (for Line Drawing). Both SCOUT and DoNaLD are translated into an intermediate language EDEN that incorporates a definitive notation, but also includes procedural features that are exploited in the interpretation of scripts. For more technical details, see [4,5,6]. (Similar principles were first applied to interactive graphics by Wyvill [7] and have been widely used as ingredient of a design system [8].)

Definitive scripts are a powerful way of representing data dependency. When interpreting Figure 1, we imagine the relationships that hold between the picture elements. For instance, the disposition of forces on the shaft correspond to the positions of the gears on the lathe. The graphs that depict the shear force, the moments and the deflection are related to each other and to the physical characteristics of the shaft, such as the stepping profile and the stiffness of the material. A script defines an environment in which it is possible to explore the relations between picture elements by changing parameters and observing the effects (cf a document, where this exploration is imagined by the viewer).

As the spreadsheet illustrates, scripts are well-suited to "what if?" activity. A script is a rich source of parameters that can usefully represent latent changes of state. It is easy to record and recover states of particular interest by storing and recalling extracts from scripts. Values of key parameters can be monitored using textual annotations that are functionally dependent on the system state, such as the string displaying the maximum deflection in Figure 1.

Different views of the design object can be related to different privileges to examine and redefine variables. An expert assessing the marketability of a lathe design would need privileges to explore rather than to change the design. Agents may also have different perceptions of data dependency. For instance, an engineer may redistribute the load on the lathe shaft without considering the appearance of the lathe, or cost of manufacture. For this task, the qualities of the lathe configuration may be regarded as dependent on the choice of loading parameters. To explore aesthetic designs, it would be more appropriate to consider loading as dependent on the disposition and form of the components. These two perspectives may lead to conflicting proposals, as is common in consultation, but alternative scripts can provide an appropriate basis for compromise.

Definitive scripts model instantaneous propagation of state change. In this respect, they resemble levers that transmit forces between components in a mechanical system [4]. Scripts can be used as linking mechanisms to combine components of a design that have been developed independently and use different interface conventions. This makes it possible to build "what if?" environments in an incremental fashion and to combine them hierarchically. Our lathe shaft model subsumes scripts that can be used interactively to redesign the screen layout and commentary, to inspect engineering data in tabular and graphical form and to maintain estimates of manufacturing cost based on the dimensions of components. A procedural beam analysis program [9] is at present integrated into the system as a special-purpose operator in the underlying algebra. Further integration is possible, so that the states of the program itself (e.g. the values of the transfer matrices) become part of the information dynamically maintained in the script. This will enable us to carry out deeper analysis of how fundamental engineering assumptions influence the model.

5. ILLUSTRATING THE USE OF DEFINITIVE SCRIPTS

Figure 1 is described in its entirety by a script of definitions specifying the relationships between windows, geometric components and textual annotations. Many kinds of interaction with the computer model correspond to simple redefinitions or extensions to the script. An important feature of our method is that each interaction with the system typically leads to an incremental enhancement of the script. This can act in two ways: to enrich the design model or to add functionality to the design environment (cf interaction with a conventional design package, where the scope of the system is entirely preconceived).

Some simple examples illustrate the versatility of the script:

- to relocate or change the dimensions of components on the shaft, redefine a parameter; the corresponding distribution of load will be recomputed automatically and the engineering data updated.
- to rearrange the display, or to relate window locations, redefine the opposite corners of appropriate windows.
- to take account of a (uniform) bore in a solid shaft, introduce a definition for the inner radius and redefine the function computing the second moment of area.
- to monitor how bore size affects the maximum deflection, set up a textual window that contains a warning message when the deflection exceeds a critical value, and is otherwise empty. Then redefine the radius of the bore.
- to switch from a manual to an automatic mode of constraining the deflection graph to lie in the display window, redefine the planar region displayed in the window so that the maximum y-coordinate is the maximum deflection.
- to introduce a sweep line, or a pointer to the diagram, define a line or shape whose location is determined by a sweep parameter or pointer identifier.

The important feature of these examples is that such diverse changes of state are represented simply by patterns of redefinition. Many nuances can be added to the model by making minor modifications to the script (cf adding features to a conventional design package, where the original program must be edited). These include enabling changes of state that were not preconceived when devising the script. For instance, pointers may be introduced ephemerally in the context of a single consultation, then discarded.

The power to reprogram on the fly is exploited in building up a design environment. In designing the screen layout for Figure 1, we first develop a script to define a generic figure that will either depict the

lathe or the stepped shaft, depending on the choice of parameters. When the dimension and location of components has been determined they can be recorded in an initialisation file. A redefinition in this file may be used to modify the default labelling on a figure, for instance, to substitute the label "SHEAR FORCE" for "FORCE". Such syntactic changes can improve communication between agents in the design process, allowing annotations to be translated from one natural or technical language to another.

A definitive script provides convenient ways to modify the state of the computer model. By entering redefinitions in sequence, a designer can simulate activities that would normally be specified automatically in a design package. These might involve analysis of a design, as in automatically computing deflection with differing bore radii to find the largest acceptable value, or simulation of the design object, as in monitoring deflection during a cutting process. Activities of this kind can be programmed using the action mechanism in EDEN, whereby a particular sequence of redefinitions is triggered by changes in values.

Actions can also be used to program "design assistants". As a simple example, all the graphs in Figure 1 are defined generically by a polyline. By default, point loads are depicted by V-shapes rather than vertical spikes. The designer can interactively introduce spikes in the reaction and shear force graphs by redefining the default polyline in the neighbourhood of point loads. Such redefinition is hard to specify generically, since it uses information particular to the configuration being displayed, but can be performed automatically using EDEN actions.

6. CONCLUSIONS

Definitive scripts have great potential as a means of communicating information about designs. To realise these benefits fully, we need to exploit techniques for interaction (e.g. menus, buttons and direct manipulation) that hide information from the non-specialist user. Features of this nature are already within the scope of the SCOUT system (cf [5]) and can readily be added to the model.

A more significant difficulty in exploiting definitive scripts fully in design concerns the generation and management of scripts. Our lathe shaft model contains many hundreds of definitions: most of these are generated initially by an independent program, but they are complemented by definitions describing specific choices of parameters that have been derived from "what if?" experiments in the course of developing the design. As the design progresses, the core of stable definitions increases, but fragments of scripts representing alternative designs are also accumulated, and new definitions are introduced. Future research will combine further development of our case study with consideration of better methods for specifying and managing scripts and interaction.

7. REFERENCES

1. Wilson PM, Greaves J "Forward Engineering - A Strategic Link between Design and Profit". Proc. Mechatronics Conference, Lancaster, Sept 1989
2. Smith, BC "Two lessons of logic". Comput Intell, **3**, 214-218, 1987
3. Mäntylä, M "A Modelling System for Top-Down Design of Assembled Products". IBM J Res Develop. **34**(5), Sept 1990
4. Beynon, WM, Yung, YP, "Definitive Interfaces as a Visualisation Mechanism". Proc GI'90, Canadian Inf Proc Soc, 285-292, 1990
5. Beynon WM, Bridge I, Yung YP, "Agent-oriented Modelling for a Vehicle Cruise Control System". Proc ASME Conf ESDA'92, Istanbul, Turkey, 1992 *to appear*
6. Beynon WM, Cartwright AJ "A definitive programming approach to the implementation of CAD software". Intell CAD Syst 2, Springer-Verlag, 126-145, 1989
7. Wyvill B "An Interactive Graphics Language". PhD Thesis, Univ of Bradford, 1975
8. Chmilar M, Wyvill B "A Software Architecture for Integrated Modelling and Animation". New Advances in Computer Graphics, Proc. CGI'89, 257–276, 1989
9. Dimarogonas AD "Computer-aided Machine Design". Chap. 4, Prentice-Hall, London 1989