KNOWLEDGE AND REASONING: ISSUES RAISED IN AUTOMATING THE CONCEPTUAL DESIGN OF FLUID POWER SYSTEMS

Mansur Darlington, Steve Culley and Stephen Potter

Engineering Design Centre, Department of Mechanical Engineering, University of Bath, Bath, UK
M.J.Darlington@bath.ac.uk

Abstract

Much progress has been made in the area of computer-aided designer support, but little has been made in that of design automation. Where progress has been made, it has been largely in the analytical aspects of the task (for example, simulation and stress analysis) – tasks for which computers are more suited than humans. Less tractable is automation of the early, conceptual, phase of design, heavily reliant as it is on the expert knowledge of the design practitioner. Emulating this computationally is the domain of Artificial Intelligence (AI) and requires a detailed understanding of the nature of the design process (Darlington et al, 1998).

This paper discusses some of the issues raised during an investigation in to the automation of the configuration phase of fluid power system design, and identifies some of the hurdles to be cleared before automation, supported by AI, becomes a reality. Two models, developed by the authors, are chosen to illustrate the way in which very different approaches can be taken to automating the same task with an emphasis on the knowledge that is used by designers, which must be acquired and used in automation.

Keywords: configuration design, design knowledge, automation, neural networks, case-based reasoning

1 Introduction

The design of fluid power systems is an intellectually challenging activity, one that has been described in a wide variety of texts and handbooks (e.g. Henke, 1983; Pinches and Ashby, 1989). There is also a considerable breadth of past and current research in this field (see for example, Burrows and Edge, various years).

Although generalization is always dangerous, much of this work has been associated with the 'embodiment' phases of design (Pahl and Beitz, 1996), that is, where the circuit or system has already been proposed and the tasks consist of such things as selecting specific components, sizing elements, ensuring quietness and efficiency in the system, and ensuring that the dynamic responses will be satisfactory.

The work that is dealt with in this paper is the stage prior to that described above, that is, the task of conceptualizing or configuring the basic circuit and the associated automation of that task.

This manuscript was received on 18 December 2000 and was accepted after revision for publication on 20 March 2001

Attempts at automating this process in the fluid power domain have been made, most notably by Bur ton and Sargent (1989), Kota and Lee (1993) and Zone-Ching and Chi-Chih (1993). These take a knowledge-based systems (KBS) approach, using explicit heuristic rules to drive the configuration. In other domains the KBS approach has also been taken to configuration design, for example in the domains of microprocessor design (Bowen, 1985), lift design (Marcus et al, 1988) and computer hardware (McDermott, 1982). There are a number of problems associated with these approaches, relating particularly to the acquisition, classification and maintenance of the design knowledge.

Alternative approaches to conceptual configuration, coming from other areas of Artificial Intelligence, have also been made in research over a number of years. Detailed consideration of issues relating specifically to the task of configuration design can be found in Mittal and Freyman (1989) and Wielinga and Shreiber (1997). A general overview of configuration research can be found in Franke (1998). Representative papers on some recent different approaches can be found in the same publication.

The problem with the automation of configuration design (indeed with much of the design process) is that many of the functions that might be implicated in the design process are internal and private to the designer.

However, for the purposes of automation, some insights are necessary into what these functions are and how they are effected. Whilst it is possible to provide an analysis of the externally manifested elements of the process, the important, generative elements of the design process are internal to the designer and thus less susceptible to scrutiny. Previous work has shown that analysis of the internal process is, at best, difficult, and according to some theorists may prove impossible (Dreyfus and Dreyfus, 1986). Nevertheless, to make progress this issue must be resolved.

It is necessary to establish what knowledge might drive the design process and to decide how this knowledge might be represented and applied.

This paper will, therefore, describe the issues associated with design knowledge – namely its content, representation and acquisition – and show how this knowledge has been used in two, quite different, approaches to the automatic configuration of fluid power circuits, namely a novel Neural Network system and a Case-Informed Reasoning system.

2 Knowledge Issues

2.1 Content and Representation

Knowledge content is constituted of facts (referred to as *declarative* knowledge), algorithms and procedures (*procedural* knowledge). It is not sufficient for the knowledge content to be incorporated within an intelligent system – the knowledge must be represented appropriately, that is, in such a way as to allow its successful retrieval, manipulation and application during the operation of the system. Consideration of knowledge representation issues include (Stillings et al, 1987):

- What is the knowledge involved in the performance of the task, its types, structure and organisation?
- How is this knowledge to be represented in the system?
- Does the chosen representation reflect the natural structure of the task knowledge? Is it adequate for the task? How does it bias the knowledge content?
- How is the knowledge to be acquired and/or revised?

The next consideration is that of representation schemes. Each has its own strengths and weaknesses and is better suited to some types of task than others. In general a representation technique will consist of a set of formal methods (which may be viewed as the *syntax* of the technique) for embodying the knowledge, plus some control system that comprehends the conventions, accessing and interpreting the knowledge, so as to provide the *semantics* of the system (Potter, 2000).

2.2 Knowledge Acquisition

When attempting to model intelligence using computers, the acquisition of knowledge has traditionally involved using one of a number of types of formally structured interviews between “knowledge engineers” and human experts. Originating in psychological research these include Protocol Analysis (Ericsson and Simon, 1984), repertory grids (Kelley, 1955) and transcription methodologies. This process is laborious and time-consuming – it has been termed the “knowledge bottleneck” (Lenat, 1983) in the process of system development – and the quality of ‘knowledge’ produced through such methods is questionable.

These problems have lead to a second approach: that of the application of *machine learning* algorithms. These algorithms have been developed in the course of parallel AI research into models of human learning (well described by Carbonell, Michalski and Mitchell (1983)). Most operate in an inductive manner – given a body of examples that can be considered to implicitly express some concept, the algorithm attempts to extract a description of this concept from the examples and represent it is some useful form (usually as rules, a semantic hierarchy or a trained sub-symbolic network).

3 A Model of Knowledge in a Design System

This section summarizes a knowledge classification scheme proposed in Schreiber et al (1994) and Wielinga and Schreiber (1997) for describing the knowledge contained within a design system.

Design is an example of the kind of complex and ill-defined task which requires intelligence to perform successfully. It involves the translation of some abstract statement of need into the description of a concrete artefact or plan that meets that need. The key to performing the design task successfully is the proper *application* of correct knowledge. Execution of a particular design task requires the *organisation* of and *access* to particular knowledge as dictated by that task. The categories described below encompass this knowledge, and their relationships with one another form a hypothesis as to how these may be distributed in a configuration design system.

3.1 Domain Knowledge

This category contains knowledge of the entities which constitute the domain. For configuration design, this group includes knowledge of:

- The physical elements (and their functions) which may constitute a solution. For example, that pumps, valves and hoses are used in a fluid power system, and an understanding of their separate and related functions.
- How these elements can be combined. For example, the fact that components are connected by hoses, pipes or signal lines, and the rules that allow legitimate connection.
- How groups of related elements (up to and including the system level) behave, component parameters, and so on.
- A description of the design requirements that the system understands. This category would seem to consist primarily of declarative knowledge – these elements correspond in some way to the external

evidence of the design task.

3.2 Inference Knowledge

This knowledge is 'reasoning' knowledge that allows an abstract element of a design may be made ‘more concrete’ according to the requirements specified, the intermediate abstractions already formed, design choices made elsewhere, etc. This knowledge is essentially private to the designer, it is what underpins the designer's expertise. It is the sort of knowledge that is applied when a designer picks one or a particular combination of components over another selection to satisfy a particular aspect of the overall requirement.

3.3 Strategic Knowledge

This is knowledge of how elements of inference knowledge can be arranged and controlled so as to provide a complete strategy for producing a design. This amounts to a set of high level methodologies for controlling the search for mappings from requirements to solutions. This is an example of procedural knowledge in the design process.

3.4 Working Knowledge

This is unique for each design episode and contains the specific requirements, design choices made, knowledge of the reasons for the changes to a design, feedback from the customer about the application of the designed system, etc. This category represents a ‘pool’ of knowledge about the current design activity, from which elements may be retrieved when they are necessary for invoking or applying elements from the other categories of knowledge.

3.5 Common-Sense Knowledge

In addition to the above, there is that, which, for want of a better term, could be called ‘*common-sense*’ knowledge. This category includes knowledge which is not specific to the domain of the task in hand, but which, nevertheless, is brought to bear on the current process. For example, it contains the basic deductive, inductive and abductive reasoning ‘rules’, experiential knowledge of the world (e.g., gravitational effects) and so on. The structure and content of this type of knowledge are continuing research issues.

3.6 Knowledge Relationships

Figure 1 shows a proposal for the static relationship of the first four classes of knowledge and Fig. 2 shows the relationships between the categories that exist in a design system. These categories are still quite loosely defined, and the content will vary from domain to domain, and may even vary within a domain when, say, different design strategies are applied, so this description cannot be considered as a generative definition for a design system. However, all these categories *must* be embodied and recognizable in some form within such a system, and as such, they offer some measure of the completeness of any proposed system.

3.7 Knowledge Representations within the Design System Model

The model of knowledge shown in Fig. 2 has been developed by the authors to describe the overall process and then to construct a number of computer implementations of configuration design systems. Table 1 presents a summary of the representations chosen for each knowledge category in two very different approaches to implementing an automated fluid power configuration design tool.

The approaches are:

- A neural-network approach, using sub-symbolic representations of design knowledge to infer the solution.
- A case-based reasoning approach. This retrieves elements of a number of previous good designs and combines them as the basis for the solution to the current problem and, as an illustration of the way that that knowledge of the types identified above can be embodied in a computational system, the CBR-derived system (referred to here as Case Informed Reasoning) and one of a number of neural network-based systems are described below. These and other approaches and their implementations are described further in Potter (2000).

4 A Neural Network-based Automatic Configuration Design System

In response to the problems referred to in the introduction, the authors initially adopted a *machine learning* approach to acquiring and expressing design knowledge. The hypothesis thus presented is that design knowledge can be inductively learned from example design cases. If the knowledge for performing the fluid power systems design task can be considered as a set of generalised associations between requirements and the domain components, then the machine learning task is one of extracting these associations from a set of example data. The data for this task would take the form of a number of ‘good’ existing design cases in this domain, each consisting of a set of requirements and the 'satisfying' circuit design.

4.1 Preparatory Tasks

Essential to any such machine learning of design knowledge, then, is a collection of examples design process. To this end, it was necessary to assemble such examples into a design archive (Darlington and Potter, 1998). Following an analysis of the domain and design task it was clear machine learning would have to proceed through the following steps:

- *Expression of requirements*: The definition of a consistent and computationally tractable method of expressing the set of requirements.
- *Expression of solutions*: the identification and description in a computationally tractable manner of the solution elements
- *Archive processing*: The processing of the archive examples so that each is described in terms of the devised requirements and solution ‘languages’.

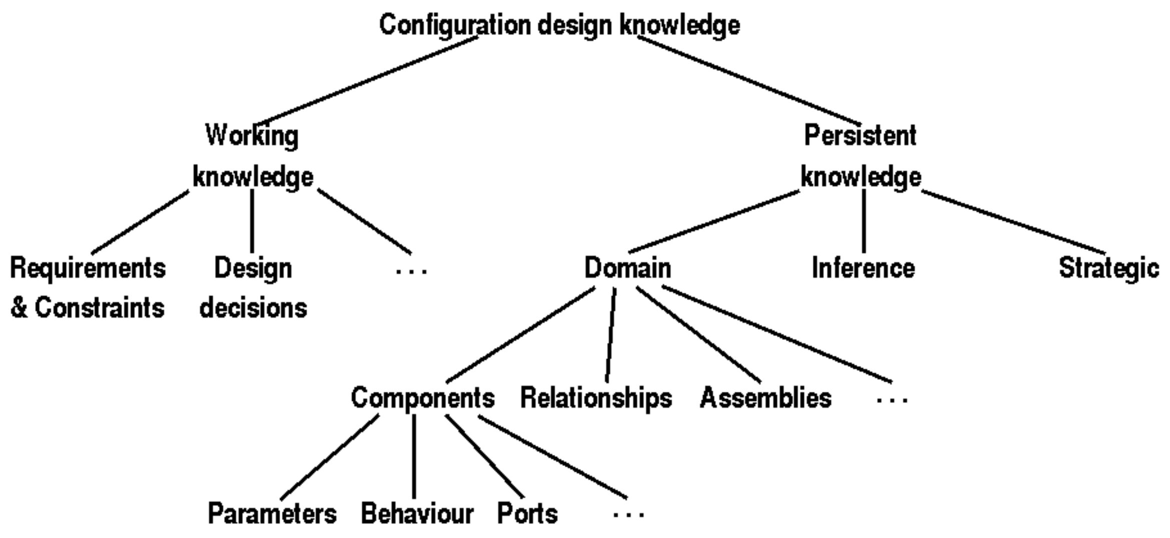

Fig. 1: *The hierarchical relationships of design knowledge (Wielinga and Schreiber, 1997)*

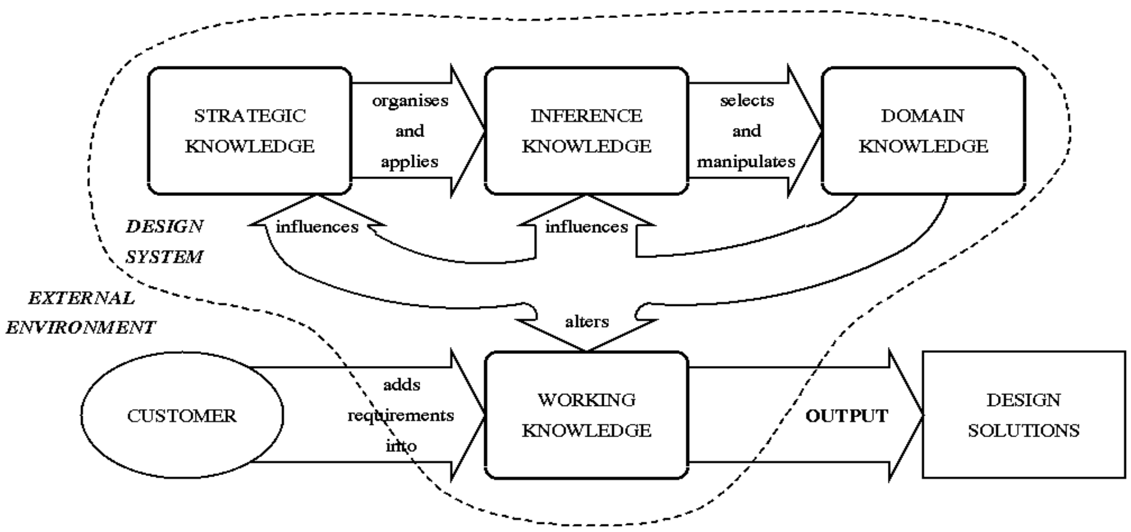

Fig. 2: *The relationship between the knowledge categories during the design process*

Table 1: A summary of knowledge representations chosen to embody the design model in neural-network-based and a CRB-based systems

	Strategic Knowledge	Inference Knowledge	Domain Knowledge	Working Knowledge
Neural network-based system	top-level design methodology, represented by sequence and control of inference knowledge in imperative program code	trained artificial neural networks	inputs and outputs of neural networks (designating requirements and components)	initial requirements, intermediate network outputs
CBR-derived system	design by retrieval of elements of a number of previous good designs and combining them into a single solution – represented by imperative program code, and meta-rules and inference engine	retrieval mechanism (solution elements of previous designs indexed on the requirements that they satisfy).	parameters designating design requirements, complete solution designs, and 'explanation rules' that locate solution elements within an existing solution case.	initial requirements, retrieved design elements, working facts in inference engine during modification phase

- *Configuration design methodology*: The derivation of some design methodology for fluid power systems. Within this, the design knowledge required to apply the methodology can be recognised. Identifying the required knowledge is tantamount to identifying the learning tasks that must be accomplished. (The methodology is itself design knowledge, although postulated rather than machine-learned.)
- *Machine learning*: The choice and application of a machine learning algorithm to the identified learning tasks.

4.2 Expression of Requirements

Design requirements in this domain can be extremely complex. However, by focusing on capturing the system functionality at the point of actuation, and ignoring most non-functional design influences, it was possible to postulate a constrained computationally tractable method of expressing requirements. Representation is limited here to steady-state requirements. Responding to the difficulties of design requirement capture and representation, and extending the expressiveness to capture more dynamic elements, is the subject of continuing research. The requirements for a linear drive are specified in terms of one of a limited set of possible values for each of 14 descriptive elements, as follows:

- maximum magnitude of load [low ($\leq 1{\cdot}10^3$ kg) /medium ($1{\cdot}10^4$ kg - $1{\cdot}10^5$ kg) /high ($\geq 1{\cdot}10^6$ kg)]
- maximum magnitude of speed [low ($\leq$ 0.01 m/s) /medium (0.01 m/s - 1.0 m/s) /high ($\geq$ 1.0 m/s)]
- plane of motion [horizontal/non-horizontal]
- speed range to be continuously variable [no/yes]
- accuracy of control [low/high]
- load to be held stationary at any position during operation [no/yes]
- smooth accelerations required [no/yes]
- load to be held stationary in the event of system failure [no/yes]
- speed to be independent of the magnitude of the load [no/yes]
- speed of actuator extension to be controlled [no/yes]
- speed of actuator retraction to be controlled [no/yes]
- the solution requires a rotary motor [no/yes]
- inertial effects to be controlled [no/yes]
- energy efficiency to be of paramount importance [no/yes]

4.3 Expression of Solutions

Design archive analysis suggested that each solution circuit is based upon one of a number of common skeletal frameworks, or *templates.* Each template provides the basic functionality essential to any valid solution. A typical template might contain hydraulic pump, actuator, and pipe-work and a control valve. The precise functionality is provided by the insertion of particular components, or the replacement of a template component, in one of a set of predefined positions, or *slots*, in the template (see Fig. 3).

4.4 Archive Processing

Given an agreed method for expressing design requirements and circuit solutions, each of the archive examples can be described in a consistent, formal manner, conditions necessary for successful machine learning.

4.5 Configuration Design Methodology

To perform this design task, some methodology must be followed which will translate the set of requirements into the circuit solution. Based on evidence from practising engineers and an analysis of the archive material, such amethodology has been devised (Darlington, 1998) which decomposes the task into several distinct, sequential stages of reasoning. This decomposition is intended to reflect how a human designer might approach the design task, indicating the areas that can be dealt with separately, and the order in which they should be addressed. In addition to providing a context for performing the design task, this methodology also provides a context for machine learning about performing the task – the design system must know how to perform each stage.

Components and Slot Positions
pressure relief valve 1 (template slot A)
pressure relief valve 1 (A and B)
pilot-operated check valve (A)
counter-balance valve 1 (A)
counter-balance valve 1 (A and B)
counter-balance valve 2 (A)
pressure compensator (C) and proportional valve (DCV)
deceleration valve (A and B)
meter-out valve (A)
meter-out valve (B)
check valve combination (AB)
variable pressure-compensated restrictor valve (D)
pressure relief valve 2 (AB)
variable displacement pump (PUMP)
motor (ACT)
proportional valve (DCV
closed-centre spool (DCV)

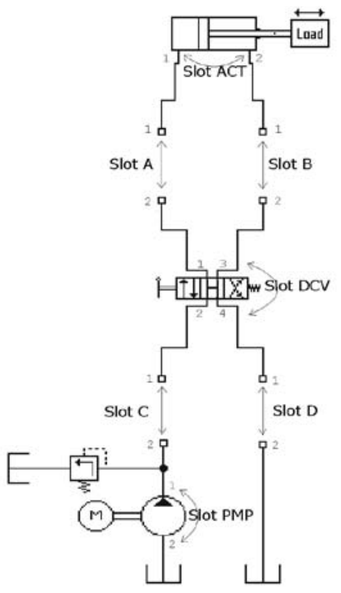

Fig. 3: *An appropriate template and 17 elements and template slots are used to describe the archive solutions*

Figure 4 summarises this ‘stage model’ of the fluid power systems design process, indicating how the tasks are broken down and ordered. These are self explanatory except in the final stage, which is that of selecting ‘contextual’ components, those that are suggested by the decisions that have been made in earlier stages.

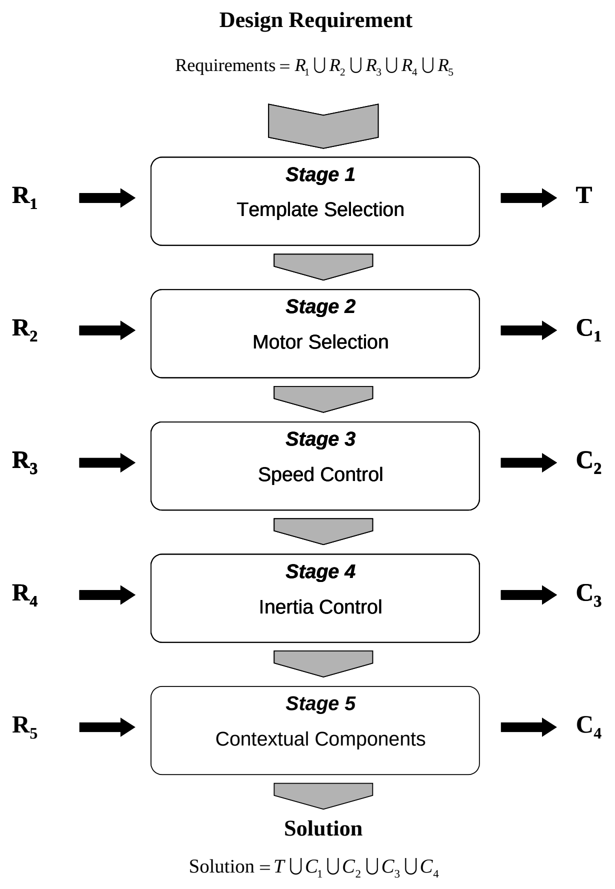

Fig. 4: *Design methodology*

4.6 Machine Learning

For this system, artificial neural networks (ANNs) (Lippman, 1987) have been chosen as the machine learning technique to be used for acquiring the design knowledge. A network may be ‘trained’ to respond to certain input patterns by producing some associated output pattern. This is achieved by repeatedly presenting examples of the correct combinations of inputs and outputs to the network, and gradually altering the connection weightings so that, in the case of every example, the input produces the desired network output.

In the devised configuration design methodology, the template selection stage would be performed by a trained ANN. For each template, there would be trained networks for making the speed and inertia control selections, and the contextual components selection. Only relevant sub-sets of the entire set of requirements form the inputs to each network, with the appropriate component/slot combinations forming the output (see Fig. 5). The outputs from each stage are collated to give the solution.

4.7 Design System Implementation

To investigate the appropriateness of this approach to automating the system, a prototype system has been constructed by the authors. The decision was made to limit this system to reasoning about solutions based upon a single template, so that the system could be rapidly developed and appraised. Hence, no initial template selection stage is required.

Twenty-five archive cases were used to form the training data for the remaining networks. Separate data sets, containing the relevant translated requirements (as input patterns) and components (as output patterns), are constructed to train the network associated with each stage, using the *backpropagation* training algorithm (Rumelhart, Hinton and Williams, 1986) to learn the relationships between the requirements and the components. The artificial networks were implemented using the neural net simulation tool SNNS (Zell, 1996). The input (requirement) and output (solution) for an example design case, together with a solution configuration is given below in Fig. 6.

Analysis of the performance (see Section 7) – in the light of practical considerations of acquiring sufficient data for training, and the acknowledged immaturity of current machine learning techniques – suggested that alternative approaches might be usefully adopted for acquiring and using the necessary design knowledge. One such, quite different approach is detailed below for comparison.

4.8 Design System Implementation

To investigate the appropriateness of this approach to automating the system, a prototype system has been constructed by the authors. The decision was made to limit this system to reasoning about solutions based upon a single template, so that the system could be rapidly developed and appraised. Hence, no initial template selection stage is required.

Twenty-five archive cases were used to form the training data for the remaining networks. Separate data sets, containing the relevant translated requirements (as input patterns) and components (as output patterns), are constructed to train the network associated with each stage, using the *backpropagation* training algorithm (Rumelhart, Hinton and Williams, 1986) to learn the relationships between the requirements and the components. The artificial networks were implemented using the neural net simulation tool SNNS (Zell, 1996). The input (requirement) and output (solution) for an example design case, together with a solution configuration is given below in Fig. 6.

Analysis of the performance (see Section 7) – in the light of practical considerations of acquiring sufficient data for training, and the acknowledged immaturity of current machine learning techniques – suggested that alternative approaches might be usefully adopted for acquiring and using the necessary design knowledge. One such, quite different approach is detailed below for comparison.

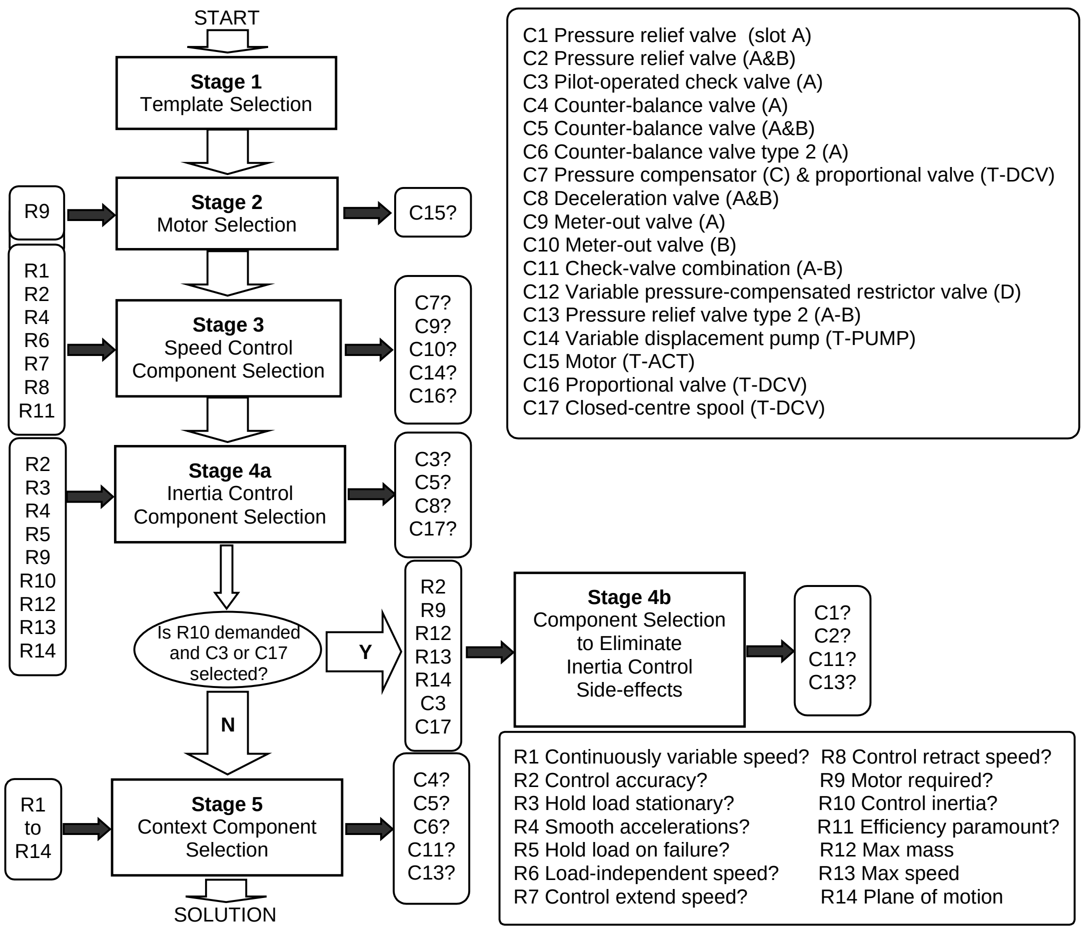
START
Stage 1
Template Selection
R9
Stage 2
Motor Selection
C15?
R1
R2
R4
R6
R7
R8
R11
Stage 3
Speed Control
Component Selection
C7?
C9?
C10?
C14?
C16?
R2
R3
R4
R5
R9
R10
R12
R13
R14
Stage 4a
Inertia Control
Component Selection
C3?
C5?
C8?
C17?
Is R10 demanded and C3 or C17 selected?
Y
N
R2
R9
R12
R13
R14
C3
C17
Stage 4b
Component Selection to Eliminate Inertia Control Side-effects
C1?
C2?
C11?
C13?
R1
to
R14
Stage 5
Context Component Selection
C4?
C5?
C6?
C11?
C13?
SOLUTION
C1 Pressure relief valve (slot A)
C2 Pressure relief valve (A&B)
C3 Pilot-operated check valve (A)
C4 Counter-balance valve (A)
C5 Counter-balance valve (A&B)
C6 Counter-balance valve type 2 (A)
C7 Pressure compensator (C) & proportional valve (T-DCV)
C8 Deceleration valve (A&B)
C9 Meter-out valve (A)
C10 Meter-out valve (B)
C11 Check-valve combination (A-B)
C12 Variable pressure-compensated restrictor valve (D)
C13 Pressure relief valve type 2 (A-B)
C14 Variable displacement pump (T-PUMP)
C15 Motor (T-ACT)
C16 Proportional valve (T-DCV)
C17 Closed-centre spool (T-DCV)
R1 Continuously variable speed?
R2 Control accuracy?
R3 Hold load stationary?
R4 Smooth accelerations?
R5 Hold load on failure?
R6 Load-independent speed?
R7 Control extend speed?
R8 Control retract speed?
R9 Motor required?
R10 Control inertia?
R11 Efficiency paramount?
R12 Max mass
R13 Max speed
R14 Plane of motion

Fig. 5: *Detailed design methodology*

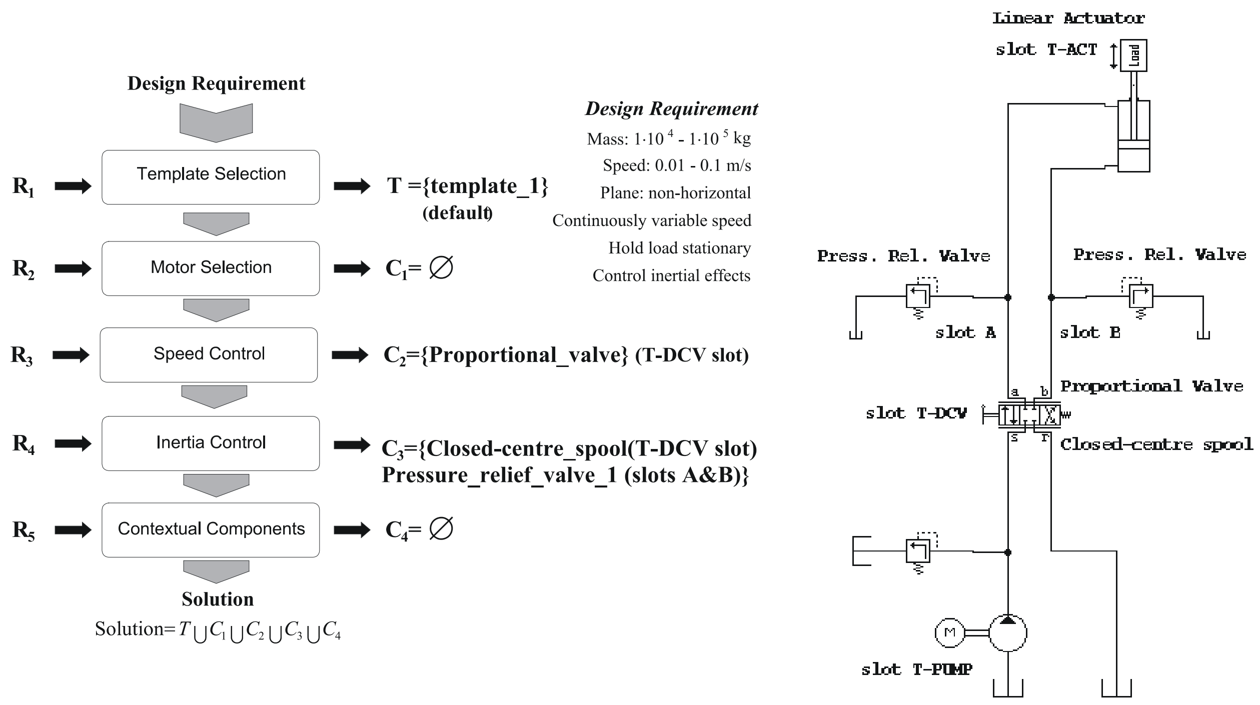

Fig. 6: *An example design case. showing the requirement set, the solution set and the configuration design*

Table 2: The parameters for describing the design requirements, the values that each can assume and the type of each (f = functional parameter, c = characteristic parameter)

Parameter name (explanation)	Possible values	Type
Continuously variable speed (Is the speed range to be continuously variable?)	yes, no	f
Hold load stationary (Must there be the facility to hold the load stationary at any position?)	yes, no	f
Smooth accelerations (Should the accelerations / decelerations of the system be jolt-free?)	yes, no	f
Hold load on failure (Should the load be held stationary in the event of system failure?)	yes, no	f
Load-independent speed (Should the speed be independent of magnitude of the load?)	yes, no	f
Control extend speed (Should the speed of actuator extension be controlled?)	yes, no	f
Control retract speed (Should the speed of actuator retraction is be controlled?)	yes, no	f
Solution requires motor (Does the solution implement rotary motion?)	yes, no	f
Inertial control (Should inertial effects to be controlled?)	yes, no	f
Energy efficiency paramount (Is energy efficiency of paramount importance?)	yes, no	c
Control accuracy (What level of control accuracy is required?)	high, low	c
Maximum load (What is the maximum operational force/load?)	high, low	c
Maximum speed (What is the maximum operational speed?)	high, low	c
Plane of motion (In which plane is the motion/force to be applied?)	horizontal, off-horizontal	c

5 A Case-informed Reasoning Automatic Configuration Design System

The Case-Based Reasoning paradigm in AI is analogous to the use of previous design experiences to solve new design problems. Maher and Garza (1997) provide an overview of the considerable body of CBR research.

Many approaches conform to a strict two-stage model of CBR: a design case is retrieved from the memory of cases, and then evaluated and adapted to the needs of the current problem. The current paper presents an alternative approach; rather than being used to adapt solutions (which requires complex, low-level domain knowledge), relatively simple generalised knowledge is used in the case-retrieval mechanism to attempt to identify useful elements of a number of cases, combinable in constructing the new solution. Thus, the term case-informed reasoning has been adopted to punctuate the distinction between this and more conventional case-based approaches.

5.1 Requirements and Solution Representation

The representation of the requirements for this approach is essentially the same as for the neural network implementation. It was recognised, here, however, that certain of the parameters seemed to be describing some aspect of the performance or characteristic of the system to be designed as a whole (these are termed *characteristic parameters*). In contrast to these, the remaining parameters refer more directly to the functions that the system must fulfil (*functional parameters*). This distinction is exploited later on in the model of the design process, and illustrates how additional knowledge might be used in a problem-solving task. Table 2 shows these parameters, the values that each can take, and their classifications. The values of parameters are used to index the cases during the reasoning phase.

The representation of the solutions is essentially that utilized in the ANN stage-model implementation, that is to say, a template which can be populated by components or component groups place in appropriate slots.

6 A CIR Model of the Configuration Design Process

The model of the design process adopted here is based upon the recall and integration of elements of previous design solutions from memory, rather than the recall of a complete solution. Each of these elements appears to have solved some aspect of the design problem that is also a feature of the current problem, and, moreover, has done so in an (apparently) similar context.

6.1 Explanation Rules

Analytical knowledge of the domain is required in order to decompose the case solutions into useful elements. This knowledge is in the form of *explanation rules*. Each of these rules has the form:

$$\{x \cdot x \in S\} \rightarrow f \qquad (1)$$

where S is the set of solution elements, and f is one of the functional requirements. The rule may be read as, 'this set of solution elements can be used to provide functional requirement f'. There may be a number of different rules for each functional requirement, reflecting the different ways in which these requirements can be achieved. In addition, individual solution elements may be related to more than one functional requirement.

These rules allow the construction of hypothetical explanations of the manner in which a particular design solution achieves its functionality. In the archive of design examples, each design solution is paired with the corresponding requirements from which it was generated. The rules explain the presence of each element in the solution by virtue of its satisfying (or contributing to the satisfaction of) one or more of the functional requirements.

A set of about 30 of these explanation rules was developed by analysing explicit, factual, knowledge of the type to be found in textbooks and handbooks. There are about 30 such rules, and each functional requirement is referred to by at least one rule. By way of an example, the rules for the *pilot-operated check valve in template slot A,* here abbreviated to POCV_A, are as follows:

- POCV_A → hold load stationary.
- POCV_A → hold load on failure.

Under certain conditions, this element stops the flow in the system. The rules describe how this behaviour can be used (on its own) to satisfy one or both of two particular functional attributes.

In addition, there are also rules that state:

- CBV1_AandB → hold load stationary.
- DECV_A&B → hold load stationary.
- CC_DCV → hold load stationary.

These, however, are *not* design rules (unlike those in, for example, an expert system) since, given a particular functional requirement to satisfy, they do not provide sufficient information to allow the choice to be made between the sets of solution elements that can be used to achieve that function. What is lacking is some indication of the *context* in which a certain set of elements will achieve the function: these are *analytical* rather than *synthetic* rules.

One basis for making this choice lies in the archive of example design cases, in which the use of solution elements in context can be seen.

6.2 The CIR Algorithm

This set of explanation rules and the case-base of designs are used to solve new configuration design problems in the following manner. A new set of design requirements is presented to the system. Initially, the new design solution consists of the template alone. For each *functional* requirement that is demanded (i.e., has the value *yes*), a search is made of the case memory to identify those cases in which this requirement is asked for. If the attribute is satisfied in more than one case, then the *best* case is that which has a set of requirements that is the most similar *in its entirety* to that of the new problem.

Once this best case is found, its design solution is examined, and, using the explanation rules, the set of solution elements in the circuit that provides the functional requirement under consideration is identified. If it does not already exist in the current solution, each of the elements in this set is then added to the solution.

If, however, no example of the satisfaction of a particular functional requirement exists in the case-base, then an explanation rule associated with the requirement is selected by default to suggest the solution element(s) to use.

The process then continues with the next functional requirement asked for, and so on, until all have been satisfied, at which point, a complete design solution is considered to have been constructed.

In this system the requirements are described in terms of binary-valued parameters, and a simple Hamming distance-type measure is applied to determine the 'distance' of the set of requirements of a case from the new set of requirements. The number of values of corresponding requirement parameters that differ between the two sets is the distance between them. That case having the fewest differences from the new set is considered to be the best matching case.

7 System Performance

This paper has presented two different approaches to automating the configuration design process with the emphasis on illustrating the requirements and use of knowledge in this type of problem solving. Accordingly, and for reasons of brevity, only the key points of the system performance are summarised below. Exhaustive discussion of these and a number of other similar approaches can be found in Potter (2000).

7.1 Stage-Model System Performance

The performance of the system when presented with test cases suggests that some of the underlying associations relating elements of the requirement to the components have been successfully learned – that is, inferential domain knowledge has been acquired. However, in response to some of the test cases, poor design decisions are made. There are a number of possible reasons why this is so:

The amount of data is less than would generally be considered acceptable for training ANNs (and since few circuit designs, *complete with their corresponding requirements*, seem to be available, this may be a feature of the domain). In addition, ANNs may not be the most suitable form of representing the design knowledge, or else, valuable information may have been lost when making the translation of the archive data into numeric form required by the networks.

The methods of expressing both the requirements and the solutions may contain inappropriate elements (or elements described inappropriately), or, conversely (and especially so in the case of the requirements), may be missing certain elements which are vital to the complete expression of this design problem.

The stage model of the design process may be an inaccurate reflection of the actual process, causing poor decisions to be made.

7.2 CIR System Performance

As would be expected, given the use of the explanation rules in the algorithm, this system provides solutions in which all the desired functionality is embodied in some identifiable form. To this extent, the solutions that are generated appear to be plausible. Nevertheless, there are aspects of the implementation and performance that bear further scrutiny, the more important of which are listed here.

This approach to design synthesis relies on a number of features of the fluid power design task that are not necessarily shared by all engineering design tasks. In particular, the design requirements can be stated in such a way that their expression of the desired functionality can be directly related to elements or sets of elements within designs. This may not be true of some other design tasks.

The elements chosen to describe the design requirement are relatively few in number and the same for each case. This impoverished description is at odds with the sophistication of the task itself.

The matching algorithm is unsophisticated. Obviously, there is much potential for improving upon the methodology outlined in this paper. More thought might be given to the order in which the functional requirements are considered. The rules might be embellished with more contextual information and, in general, the representations used might be made more realistic. Since they have been used in similar situations previously, it is assumed that the retrieved solution elements will be compatible in every case — this is unlikely to be so.

The CIR system's performance appears to be better than that of the ANN system, giving consistently plausible designs. The next task is to consider how good these design are – that is, to what extent do the solutions demonstrate the application of expertise.

8 Conclusion

If successful automation of complex human problem-solving exemplified by engineering design is to be achieved then a proper understanding of the knowledge that is used and its application is necessary.

This paper has presented two quite different approaches to the automation of the circuit configuration phase of fluid power design. The design and implementation details of the system associated with each approach illustrate the types, content and representation of the knowledge that is typically used by designers, and therefore required to be embodied in computational systems if successful automation is to be achieved.

The neural network implementation shows both how the usual human process of problem decomposition can be adopted to provide a methodology for proposing the application of strategic knowledge, and how machine learning can be used for acquiring and applying inferential knowledge to the design task.

The Case-Informed Reasoning implementation illustrates how previous experience – in the form of existing designs – can be analysed to furnish component-requirement relational rules and the context necessary for piecing together solutions to new problems.

Acknowledgements

The research work reported here was undertaken in the Engineering Design Centre at the University of Bath, UK, funded through the UK EPSRC grant GR/L26858.

References

Bowen, A. J. 1985. Automated configuration of hardware for dedicated microprocessor application systems. *Knowledge Engineering in Computer-Aided Design*, J. S. Gero, ed., Elsevier Science Publishers, North-Holland.

Burrows, C. R. and **Edge, K. A.** 1987-1997. Fluid Power Engineering. *Bath International Fluid Power Workshops*, Research Studies, Baldock.

Burton, R. and **Sargent, C.** 1989. The use of expert systems in the design of single and multi-load circuits. *International Conference on Fluid Power Transmission and Control ICFP '89*, Hangzhou, China.

Carbonell, J. G., **Michalski, J. K.** & **Mitchell, T. M.** 1983. An overview of machine learning. Michalski, J. K., Carbonell, J. G. and Mitchell, T. M., *Machine Learning: an Artificial Intelligence Approach*, Tioga, Palo Alto, Ca, pp. 3–23.

Gero, J. S. and **Sudweeks, F.** 1998. *Artificial Intelligence in Design ’98* (Darlington, M. J., Potter, S., Culley, S. J. and Chawdhry, P. K., Cognitive Theory as a Guide to Automating the Design Process). Kluwer, Dordrecht.

Darlington, M. J. and **Potter, S.** 1998. *A fluid power systems design archive, version 2*. Internal Report No. 49/98, Department of Mechanical Engineering, University of Bath, Bath, UK.

Darlington, M. J. 1998. *Problem reduction in complex domains.* Internal Report No. 52/98, Department of Mechanical Engineering, University of Bath, Bath, UK. (Available on-line from: http://www.bath.ac.uk/Departments/Eng/edc/home.html)

Dreyfus, H.L. and **Dreyfus, S.E.** 1986. *Mind over Machine*. Free Press, NY.

Ericsson, K. A. and **Simon, H. A.** 1984. *Protocol Analysis*. MIT Press, Cambridge, MA, USA.

Franke, D. W. 1998. Configuration Research and Commercial solutions. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, Vol. 124, pp. 295-300.

Henke, R.W. 1983. *Fluid Power Systems & Circuits*. Penton, Cleveland, Ohio.

Kelley, G. A. 1955. *A Theory of Personality – the Psychology of Personal Constructs*. Norton, New York.

Kota, S. and **Lee, C.-L.** 1993. General framework for configuration design: part 2 – application to hydraulic systems configuration. *Journal of Engineering Design*, 44, pp. 291-303.

Lenat, D. B. 1983. The role of heuristics in learning by discovery: three case studies. Michalski, J. K., Carbonell, J. G. and Mitchell, T. M., *Machine Learning: an Artificial Intelligence Approach*, Tioga, Palo Alto, Ca, pp. 243–306.

Lippmann, R. P. 1987. An introduction to computing with neural nets. *IEEE Acoustics, Speech and Signal Processing Magazine*, 42, pp. 4 - 22.

Maher, M. L. and **Garza, A. G. de S.** 1997. Case-based reasoning in design. *IEEE Expert Magazine*, pp. 34 - 41.

Marcus, S., **Stout, J.** and **McDermott, J.** 1988. VT: An expert elevator designer that uses knowledge-based backtracking. *AI Magazine*, 91, pp. 95 - 112.

McDermott, J. 1982. R1: A rule-based configurer of computer systems. *Artificial Intelligence*, 191, pp. 39 - 88.

Mittal, S. and **Frayman, F.** 1989. Towards a generic model of configuration tasks. *11th International Joint Conference on Artificial Intelligence (IJCAI)*, Detroit, Michigan, USA, pp. 1395 - 1401.

Pahl, G. and **Beitz, W.** 1996. *Engineering Design: a systematic approach, 2nd ed*. Springer, London.

Pinches, M. J. and **Ashby, J.G.** 1989. *Power Hydraulics*. Prentice Hall, New York.

Potter, S. 2000. *Artificial Intelligence and Conceptual Design Synthesis*. PhD thesis, University of Bath.

Rumelhart, D. E., **Hinton, G. E.** and **Williams, R. J.** 1986. Learning internal representations by error propagation. Rumelhart, D. E. and McClelland J. L., *Parallel Distributed Processing : Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, MA., USA.

Schreiber, G., **Wielinga, B.** and **de Hoog, R.** 1994. CommonKADS: a comprehensive methodology for KBS development. *IEEE Expert Magazine*, pp. 28 - 37.

Stillings, N. A., **Feinstein, M. H.**, **Garfield, J. L.**, **Rissland, E. L.**, **Rosenbaum, D. A.**, **Weisler, S. E.** and **Baker-Ward, L.** 1987. *Cognitive Science: An Introduction*. MIT Press, Cambridge, MA., USA.

Wielinga, B. and **Schreiber, G.** 1997. Configuration-design problem solving. *IEEE Expert Magazine*, pp. 49 - 56.

Zell. A. et al. 1996. *Stuttgart Neural Network Simulator, User Manual, version 4.1*. Institute for Parallel and High-performance Systems, University of Stuttgart, Germany.

Zone-Ching, L. and **Chi-Chih, S.** 1995. An investigation of an expert system for hydraulic circuit design with learning. *Artificial Intelligence in Engineering*, 9, pp. 153 - 165.

Mansur Darlington
is a cognitive scientist working in the Engineering Design Centre at the University of Bath. He has a particular interest in the capture of the design requirement and how the conceptual and language content of design requirement expression can be harnessed to control and formalize the elicitation process for automation.

Stephen Potter
was formerly a researcher in the Engineering Design Centre within the Department of Mechanical Engineering at the University of Bath. His doctoral thesis is entitled "Artificial Intelligence and Conceptual Design Synthesis".

Steve Culley
is Head of Design in the Department of Mechanical Engineering at the University of Bath. He has researched in the engineering design field for many years. In particular this work has centred on the provision of information and support to engineering designers. He pioneered research into the introduction and use of the electronic catalogue for standard engineering components and has extended this work to deal with systems and assemblies. He has over 100 publications and is currently in the process of writing a book.