AN APPROACH TO FORMULATE AND TO PROCESS DESIGN KNOWLEDGE IN FLUIDICS [1]

Benno Stein [*] **Elmar Vier** [**]

[*] *Department of Mathematics and Computer Science (Prof. Dr. H. Kleine Büning), Knowledge-based Systems, University of Paderborn, D–33095 Paderborn, Germany*
email: stein@uni-paderborn.de

[**] *Department of Measurement and Control (Prof. Dr.-Ing. H. Schwarz), Faculty of Mechanical Engineering, University of Duisburg, D–47048 Duisburg, Germany*
email: vier@uni-duisburg.de

Abstract: At present, the design of complex fluidic systems cannot be automated. This results from the creativity that is needed when synthesizing a new circuit to meet the desired demands, but also from complex technical dependences that have to be processed. Thus we start with the following working hypothesis here: There still exists a preliminary design of a circuit which can be "repaired" respecting the demands by a sequence of suited modifications.

This paper contributes to the above idea within the following respects: It identifies different types of hydraulic design knowledge and presents an approach to formalize and to process this knowledge.

Keywords: Computer Aided Design, Expert Systems, Control Systems, Hydraulics.

1. INTRODUCTION

> *The metagoal of design is to transform requirements, generally termed function, which embody the expectations of the purpose of the resulting artifact, into design descriptions.*
> *(Gero, 1990), p.28*

Formally stated, the purpose of a design process is the transformation of a complex set of functionalities D (= demands) into a design description C (= configuration):

$$D \longrightarrow C$$

"$\longrightarrow$" stands for some transformation, C is considered the artifact's entire set of components and their relations. The transformation must guarantee that the artifact being described is capable of generating the set D of demands. Due to the complexity and the diversity of a design process, no universal *theory of design* can be stated, i. e., in the very most cases no direct mapping is given between the elements $d \in D$ and the objects $o \in C$. [2]

Applied to fluidics, D may define courses of forces, damping rates, or maximum pressure values, while C stands for a circuit's diagram.

Working on a design problem means to balance two behavior sets: the set of desired or expected behavior, B_e, and the set of observed behavior, B_C. B_e can directly be derived from a designer's understanding for D, whereas B_C is the result of an analytical investigation of C that, in the fluid domain, often encloses complex model formulation and simulation tasks (Nakashima and Baba, 1989; Piechnick and Feuser, 1994; Stein, 1995).

This balance process forms a design cycle in which the expected behavior B_e (the desired properties of a circuit) controls the modification of the circuit C. Within an evaluation phase, the analyzed behavior B_C and the expected behavior B_e are compared to each other providing input for a next modification step. Figure 1 illustrates the dependences.

[1] The authors acknowledge support of the "Deutsche Forschungsgemeinschaft (DFG)", Germany.

[2] A special case of a direct mapping between $d \in D$ and $o \in C$ is the so-called "catalog look up".

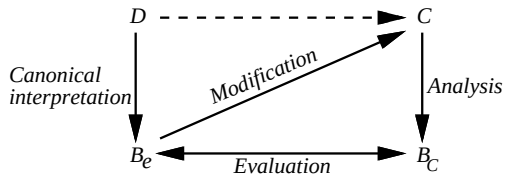


Figure 1. A model of design according to Gero.

The paper in hand concentrates on the modification step depicted in Figure 1; it discusses the role of modification knowledge in hydraulics (Section 2) and presents an approach to formulate and to process this knowledge (Section 3 and 4 resp.). This approach has prototypically been realized.

2. IMPROVING HYDRAULIC SYSTEMS

It is an inherent property of our approach not to start hydraulic circuit design from scratch. Hydraulic manipulation jobs vary from simple lifting problems up to the realization of complex robot kinematics, and, given a demand description D for such a manipulation job, the design of an appropriate drive is a truly creative job.

Our working hypothesis is that we still *have* a preliminary design C' of a circuit which, roughly speaking, incorporates the potential to fulfill D. Put another way, there exists a sequence of modifications $m_1 \dots m_l$ of C' that transforms C' towards the desired circuit C.

The question whether or not this is a useful working hypothesis shall not be discussed in detail here. However, the following aspects are worth to be noted:

- For a restricted field of application, circuit design may be automated completely. Note that knowledge for the modification or the repair of a circuit can formulated rather application-independently.
- To automate circuit design completely, the demand specification problem must be tackled at first—even for narrow application fields.
- Engineers tend to fall back on a previously solved design problem whose solution is modified respecting the new demands. Thus it is conceivable that a case base with well selected cases could serve as a "design entry point". [3]

Back to our modification thread. Hydraulic systems are defined by a set of components along with a topology specifying relations between these components. Components in turn are described by both invariable characteristics and variable characteristics, so-called parameters. As a consequence, qualitatively different types of modifications stand to reason (Vier and Stein, 1998):

(1) *Parameter Modification.* Parameters can be altered easily within their given ranges. Examples: the threshold pressure of a relief valve, the gain of a controller.
(2) *Characteristics Modification.* Changing a component's characteristics means to replace the component—a modification that causes some extra effort.
(3) *Topology Modification.* Modifications of this type change the arrangement of components and their connections as well as the structure of the control system. Topology modifications provide the most profound and far-reaching effects.

Given a preliminary design C', unfulfilled demands must be detected, and a suited modification measure must be selected and applied. This is not a trivial job. For instance, it can hardly be foreseen whether a particular measure is always a remedy for a malfunction; usually several measures have to be tested before an improvement is achieved (Krafthöfer, 1997; Uecker, 1997). Modifying C' towards C is a complex search process that must be controlled by smart propose-and-revise heuristics.

3. FORMULATING DESIGN KNOWLEDGE IN FLUIDICS

As justly mentioned, the identification, validation, and classification of modification knowledge for hydraulic systems is a non-trivial engineering problem. However, getting this knowledge operationalized on a computer is even more complex. Some reasons for this are the following:

Expressiveness. Design knowledge typically is very compact; an example:

"An insufficient damping can be improved by installing a by-pass throttle."

This measure encodes a lot of implicit engineering know-how, among others the following:

(1) a by-pass throttle is connected in parallel
(2) the component to which it is connected is a cylinder
(3) if there are several cylinders in the system, an engineer knows the best-suited one
(4) a by-pass throttle is a valve

Flexibility. Engineers use design knowledge in a flexible way; i. e., a particular piece of knowledge can be applied to different contexts in a variety of hydraulic circuits.

[3] We are maintaining a case library with hydraulic circuits; at present this library contains about 150 selected circuits.

Flexibility is a major reason which makes it difficult to encode the expressiveness of the above example on a computer. Consider we were confronted only with hydraulic systems of the same topological set-up, then measures like the above ("Install a by-pass throttle.") could simply be hard-wired within a ("design") algorithm.

Specifying implicit knowledge explicitly is one possibility getting the knack of the outlined problems. For these purposes we have been developing a description language tailored to hydraulic circuit design, which is presented in the following.

3.1 *Basic Concepts*

Basically the three modification types (concerning parameters, characteristics, and topology) can be addressed with the action types depicted in Figure 2.

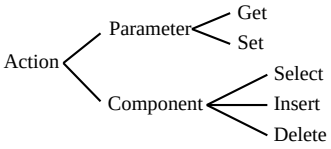


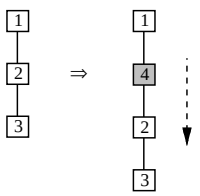
Figure 2. Different types of actions.

However, a difficulty regarding the formulation of design knowledge results from the *location* where in a circuit an action should take place—less from the modification type. I. e., a piece of knowledge describing a modification at a circuit always consists of two parts:

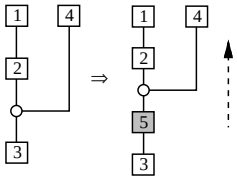
modification := action specifier + location specifier

Exactly defining the place where an action shall take place is the larger part of the modification problem. In particular it must be possible to insert or delete components relative to other components. Figure 3 sketches out the basic construction scheme for location specifiers. Qualifiers work as a filter for a set of components by checking the components' parameters against provided values.

The following figures illustrate the usage of location specifiers. Note that the right-hand-side of the examples could also form the starting point of a modification, if *insert_component* is replaced by *delete_component*.



insert_component (in_series 1 3, (4))



insert_component (in_series 4 3, (5))

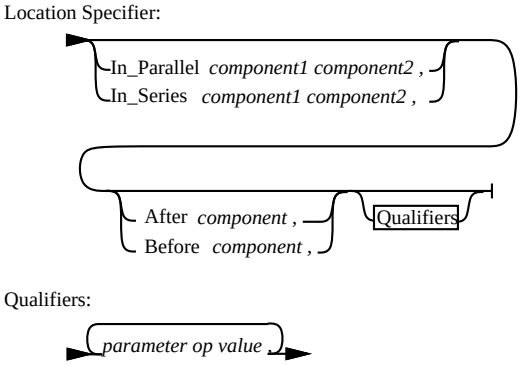
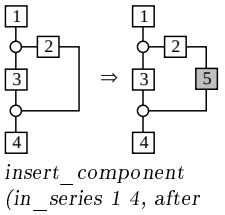
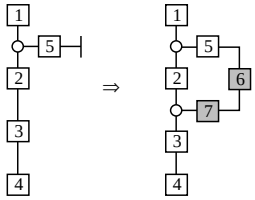


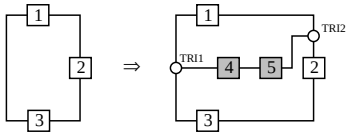
Figure 3. The structure of location specifiers.



insert_component (in_series 1 4, after 2, (5))



insert_component (in_parallel 2 3, after 5, (6 7))



1. insert_component (in_series 1 3, (TRI1))
2. insert_component (in_series 1 2, (TRI2))
3. insert_component (in_series TRI1 TRI2, (4 5))

3.2 *Modification Schemes*

Our design language provides only a small set of core functions. These functions realize the primitive, that is to say, atomic actions presented in the previous subsection:
get_parameter, set_parameter, select_component, insert_component, delete_component.

To gain flexibility and to enable the realization of user-defined abstraction hierarchies, core functions can be composed to more complex macros. Within this macro language are also different types of loops and branching concepts realized; (Schlotmann, 1998) contains a precise specification.

E. g., the following code defines a macro that determines a circuit's maximum operating pressure:

```
macro max_op_pressure () {
  p_aux := 0,
  foreach e in select_component((type=pump)) {
    p := get_parameter(e, P_LIM);
    if p > p_aux then { p_aux := p; }
  }
  return(p_aux);
}
```

Reconsider the design cycle in Figure 1: A modification is the result of a comparison and evaluation of the expected behavior B_e with the analyzed behavior B_C. Differences between B_e and B_C are called symptoms; symptoms are observed at components, and to repair a symptom, a modification of the circuit is necessary.

Modification schemes provide a concept to integrate the three aspects "component", (related) "symptoms", and (possible) "modifications". They are built on top of the macro layer (see Figure 4).

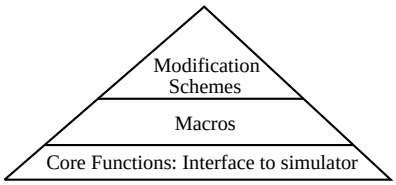


Figure 4. The different levels of abstraction.

The syntax for a modification scheme is as follows:

```
class name {
  gates { gate_1; ... gate_n; }
  parameters { var_1 type_1; ... var_m type_m; }
  repair_rule (priority_1) {
    symptoms { symptom_1; ... symptom_n1; }
    modification { ... } ... modification { ... }
  ...
  repair_rule (priority_l) {
    symptoms { symptom_1; ... symptom_l1; }
    modification { ... } ... modification { ... }
}
```

Remarks. "name" designates the name of a component class to which a modification scheme belongs. The keywords **gates** and **parameters** introduce local variables for a component's connections and parameters respectively. Each repair rule defines both a list of symptoms that quantify a misbehavior and a list of modifications to repair the misbehavior.
Note that within a modification scheme two types of choice points exist: At first, amongst the repair rules the most important rule (so to speak, the most crucial symptom) must be selected; at second, within a repair rule the most adequate modification has to be chosen.

The example below is a part of a cylinder modification scheme. The scheme shows how the problem of a non-extending cylinder piston can be addressed.

```
class cylinder {
  gates { A; B; }
  parameters {
    A_R characteristic; // ring area
    v parameter; } // velocity
  repair_rule (strict) {
    symptoms { v = 0; }
    modification {
    // Decrease resistances of involved components.
      foreach e in get_resistors(this) {
        increase_resistance(e, this, 0.1); } }
    modification {
    // Lower tank pressure.
      foreach e in get_tank_suppliers(this) {
        if get(e, p) > 0 then {
          set(e, p, add(get(e, p), -2)); } }
}
```

3.3 *Meta Knowledge*

Usually a *set* of modifications stands to reason to repair a malfunction. Note that an evaluation that exhibits to which level a modification measure was successful requires an expensive simulation. It is quite obvious that heuristics are required which access a measure's global consequences. Currently, the following criteria have been investigated for evaluation and ranking (Vier and Stein, 1998):

- A modification's *effectiveness* is most important.
- The *repercussion* on the design of the hydraulic system describes undesired side effects, which must be expected when carrying out the modification.
- Another important criterion is the *effort* required to realize a modification. It is directly related to the modification types parameter, characteristics, and topology.

To each modification alternative three assessment values v_{ef}, v_{re}, and v_{et} are assigned, either qualitatively or by means of a quantitative analysis. The v_i are in $[0;1]$; $v_{\mathrm{ef}} = 1$ stands for high effectiveness, $v_{\mathrm{re}} = 0$ stands for small repercussion, and $v_{\mathrm{et}} = 0$ stands for low effort. To obtain the absolute confidence $\mathcal{K}$, these values are weighted by the positive confidence factors κ_{ef}, κ_{re}, and κ_{et}, where

$$\kappa_{\mathrm{ef}} + \kappa_{\mathrm{re}} + \kappa_{\mathrm{et}} = 1$$

If, for example, the damping factor of a cylinder is judged to be too low, the modifications listed in the first column of Table 1 could be a possible remedy.

Modification Measure	v_{ef}	v_{re}	v_{et}	$\mathcal{K}$
throttle in mainstream	0.1	0.4	0.8	0.390
throttle in side stream	0.4	0.4	0.5	0.435
throttle in by-pass	0.8	0.4	0.5	0.635
damping network	0.9	0.8	0.1	0.605
...				

Table 1: Modifications that increase the damping.

Note that the modifications can be applied solitary or in combination; however, each of them modifies the structure the hydraulic circuit. Here the confidence factors are $\kappa_{\mathrm{ef}} = 0.5$, $\kappa_{\mathrm{re}} = 0.15$, and $\kappa_{\mathrm{et}} = 0.35$.

Installing a throttle in a by-pass to the cylinder (see Figure 5) is ranked first option. The resulting drain

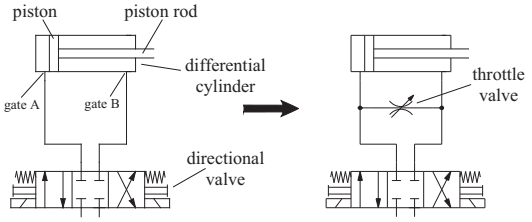


Figure 5. Set-up before and after modification.

flow through the by-pass throttle moves the eigenvalues of the related transfer function to a higher damping.

4. PROCESSING DESIGN KNOWLEDGE

4.1 *Search in the Design Space*

Starting point of the regarded design problem is a preliminary design in form of a circuit C' with unfulfilled demands. The design search space is comprised of all circuits that can be derived from C' by applying a given set of modification schemes. Cycling through the process depicted of Figure 1 means walking through the design space. A path from the root C' downto a solution defines a sequence of modifications that "repairs" all unfulfilled demands in C' (cf. Figure 6).

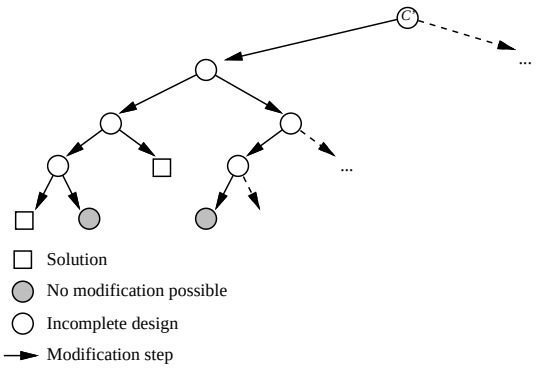


Figure 6. Exploring the design space.

Each modification step in Figure 6 is comprised of the following five jobs:

(1) Simulation of the circuit over the intended driving process. [4]
(2) Evaluation of the circuit behavior with respect to the demands. The result is a set of symptoms.
(3) Interpretation of all modification schemes with respect to their repair rules. The result is a set of applicable modifications.
(4) Scheduling of all applicable modifications.
(5) Realization of the best-rated modification.

At present, step 4, scheduling, has been realized rather rudimentary. Applicable modifications are sorted according to the following strategy:

(A) Modifications relating sources and sinks of power (pumps, cylinders).
(B) Modifications relating conducting elements (hoses, pipes).
(C) Modifications relating power control (valves).

Within each such group, the absolute confidence value $\mathcal{K}$ of a modification is used to define an order (see subsection 3.3).

If no applicable modification can be found within step 3, backtracking is invoked. If no symptoms can be detected within step 2, the circuit establishes a solution.

For complex circuits or if several involved malfunctions are to be repaired, this strategy is to shortsighted. Then, a blackboard architecture is much more adequate (Hayes-Roth, 1985; Hayes-Roth, 1983). Figure 7 shows a possible structure.

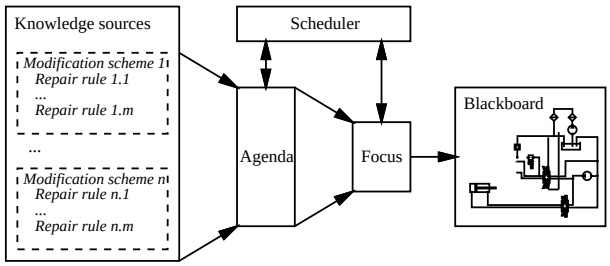


Figure 7. Blackboard organization of the design.

The blackboard makes the design object, the circuit, available. The modification schemes form knowledge sources providing hydraulic design knowledge in the form of repair rules. Applicable modifications appear on the agenda, and a focus concept helps to concentrate on a selected number of knowledge sources. The scheduler controls the search by choosing modifications from the agenda.

Note that a smart scheduling requires the combination of several strategies, among others the following:

"Exploit divide-and-conquer properties."
Modifications with no side effects should be carried out first, to fix the related malfunction.

"Sort demands."
Assess to which phase of the design process a demand is related to obtain a suitable sequence when processing modifications. For example, it is not advisable to optimize a controller while a working element does not provide the desired velocity.

"Avoid loops."
Avoid modifications that lead to new unfulfilled demands of earlier design phases.

[4] In this connection the simulation engine of *artdeco* is exploited. It should be noted that *artdeco*'s capability to generate a simulation model from a drawing is a prerequisite to automatically perform and evaluate circuit modifications at all (Stein, 1995).

4.2 *Language-specific Issues*

In first place, our design language resembles concepts of imperative programming languages. Its number of commands is intentionally left small to keep the language concept clear, and to make its application easy. Other concepts are:

- *Identifier Binding.* Identifiers are bound statically to their respective definitions. Their scope is determined by the block that is implicitly defined by a macro or a modification definition.
- *Typing.* The types of formal parameters in a macro parameter list are bound statically, whereas the types of local variables are determined dynamically—a strategy that simplifies the usage of local variables.
- *Program Control Elements.* The iteration over lists adopts the simplicity and elegance from LISP: A variable steps through a list, which in turn is allowed to comprise elements of different types (here: paths, components, parameters).
- *Tailored API.* An application programming interface with core functions for the manipulation and simulation of fluidic systems is provided (cf. Figure 4).

5. SUMMARY

Given a preliminary design C', a sequence of modifications can be found that transforms C' towards the desired circuit C. To automate such a modification approach, among others the following questions need to be answered:

(1) Of which form is the typical modification knowledge in hydraulics?
(2) How can the modification knowledge be operationalized?
(3) How can an adequate sequence of modifications be found?

This paper gives answers to these questions. Its main contribution is a tailored design language that enables an engineer to formulate modification knowledge in hydraulics. This language has prototypically been implemented.

However, key challenge concerning future work is the efficient search in the design space, which has two aspects: (*i*) The development of heuristics that evaluate unfulfilled demands and differentiate between measures, and (*ii*) the development of concepts for a smart design progress control, e. g. in the form of a blackboard architecture.

REFERENCES

Gero, John S. (1990). Design Prototypes: A Knowledge Representation Scheme for Design. *AI Magazine* **11**, 26–36.

Hayes-Roth, Barbara (1983). The Blackboard Architecture: A General Framework for Problem Solving. Heuristic Programming Project HPP–83–30. Stanford University, Computer Science Department, Heuristic Programming Projekt.

Hayes-Roth, Barbara (1985). A Blackboard Architecture for Control. *Artificial Intelligence* **16**, 251–321.

Krafthöfer, Constantino (1997). Untersuchung konstruktiver Maßnahmen zur Beinflussung des dynamischen Verhaltens hydraulischer Antriebe. Study work. Gerhard-Mercator-Universität - GH Duisburg, MSRT.

Nakashima, Yusei and Tomio Baba (1989). OHCS: Hydraulic Circuit Design Assistant. In: *First Annual Conference on Innovative Applications of Artificial Intelligence*. Stanford. pp. 225–236.

Piechnick, Martin and Alfred Feuser (1994). MOSIHS – Programmsystem zur Simulation komplexer elektrohydraulischer Systeme. In: *AFK, Aachener Fluidtechnisches Kolloquium*. Mannesmann Rexroth GmbH, Lohr, Germany.

Schlotmann, Thomas (1998). Formulierung und Verarbeitung von Ingenieurwissen zur Verbesserung hydraulischer Systeme. Diploma thesis. Universität-GH Paderborn, FB 17 Mathematik / Informatik.

Stein, Benno (1995). Functional Models in Configuration Systems. Dissertation. University of Paderborn, Department of Mathematics and Computer Science.

Uecker, Stefan (1997). Statische Auslegung hydraulischer Translationsantriebe bei der Neu- und Änderungskonstruktion. Study work. Gerhard-Mercator-Universität - GH Duisburg, MSRT.

Vier, Elmar and Benno Stein (1998). Modeling of Design Strategies for Hydraulic Control Systems. In: *MIC 98, IASTED International Conference on Modelling, Identification and Control, Grindelwald, Switzerland.*