HOW CASE-BASED METHODS CAN AUTOMATE FLUIDIC CIRCUIT DESIGN

Benno STEIN
University of Paderborn
Dept. of Mathematics and Computer Science—Knowledge-Based Systems
D–33095 Paderborn, Germany
++49 (0) 5251 60 3348 (Phone) / 3338 (Fax)
stein@uni-paderborn.de

Designing a system means to transform a set of demands, D, towards an explicit system description, S. In the field of fluidic circuit design, components like pumps, valves, and cylinders are used to construct S. From a configurational standpoint a designer selects, parameterizes, and connects components such that D is fulfilled by the emerging circuit.

Actually, fluidic circuit design is not tackled at the component level. Instead, a designer develops a mental model of the desired system, which is placed at the level of function, F. Hence, a more adequate characterization of the design process is $D \longrightarrow F \longrightarrow S$.

Using the concept of fluidic axes, the step $F \longrightarrow S$ can be automated by means of case-based reasoning. Motivated by these observations we have developed a case-based design approach for hydraulic systems.

Keywords: Fluidic Circuit Design, Case-Based Reasoning, Design Automation, Hydraulics

1 INTRODUCTION

Fluidic drives are used to realize a variety of production and manipulation tasks. Even for an experienced engineer, the design of a fluidic system is a complex and time-consuming task, which, at the moment, cannot be automated completely. Designing a system means to transform a set of demands, D, towards an explicit system description, S. From a configurational standpoint a designer of a fluidic system selects, parameterizes, and connects components like pumps, valves, and cylinders such that D is fulfilled by the emerging circuit.

$$D \longrightarrow S$$

However, fluidic circuit design is not tackled at the component level. Instead, a designer develops a mental model of the desired system, which is placed at the level of function, F. A more adequate characterization of the design process is the following.

$$D \longrightarrow F \longrightarrow S$$

Based on the concept of fluidic axes, it is possible to automate the step $F \longrightarrow S$. A fluidic axis fulfills some subfunction f, and, in order to realize a complex function $F := \{f_1, \ldots, f_n\}$, several fluidic axes must be coupled in the right way. Motivated by these observations we have developed a case-based design approach for hydraulic systems, where the following components interplay:

- A case base CB with hydraulic axes from previously solved design problems.
- A similarity function σ that maps from a desired function $f \in F$ to hydraulic axes in CB.
- A rule-based modification concept for the adaptation of unsatisfactory fitting axes.

- A composition scheme that hierarchically couples several axes respecting F.

This paper outlines the design approach;[1] it is organized as follows. The next section gives an introduction to case-based reasoning. Section 3 develops the main contribution of this paper, a case-based design approach for fluidic systems. Section 4 presents a prototypic design assistant that operationalizes the outlined ideas.

2 CASE-BASED REASONING

Let a case combine a description of a problem along with a solution. Basic idea of case-based reasoning (CBR) is to exploit previously solved cases when solving a new problem. I. e., a collection of cases is browsed for the most similar case, which then is adapted to the new situation. The commonly accepted CBR cycle shown in Figure 1 goes back to Aamodt and Plaza (1994) and is comprised of four steps:

1. *Retrieve.* A case relevant for the problem is retrieved.
2. *Reuse.* Having performed more or less adaptations, the retrieved case may be reused.
3. *Revise.* Having evaluated the adapted case, additional repair adaptations may be applied.
4. *Retain.* The new case, consisting of the problem along with a solution, is stored.

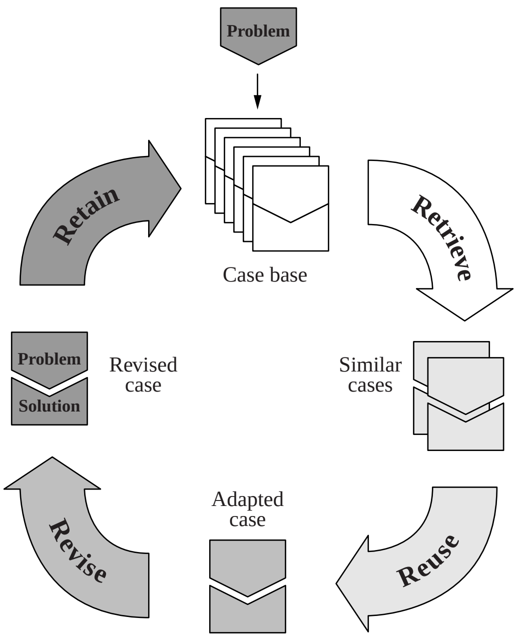


Figure 1: The classical CBR cycle.

2.1 Design Problem Solving and CBR

Configuration, design, synthesis—these terms stand for problem classes where the AI paradigm “generate and test” has been applied rather successfully (Brown and Chandrasekaran 1989, Cunis et al. 1989, Marcus and McDermott 1989). CBR, however, follows the paradigm “retrieve and adapt” (Leake 1995). Both concepts can work fine together to solve design problems.

A previously solved design problem that contributes a good deal to the desired solution may bound difficult synthesis and adaptation tasks to a tractable rest problem. Following this idea, the starting position of a design problem should be created with CBR methods, while for the heuristic and search-intensive adaptation tasks other AI paradigms come into play.

As mentioned at the outset, a design problem is stated by a set of *user demands*, D; a solution to a design problem is a *system*, S, which can be understood as a collection of objects or as some kind of construction plan. S is a solution of the design problem D, if the behavior of the system S complies with D.

[1]There exist other approaches to hydraulic circuit design, such as (Piechnick and Feuser 1994, Fluidon GmbH 1992). The main difference to the approach presented here is that only predefined circuit topologies are treated. An exception is the SCHEMEBUILDER system, which allows for the construction of simple parallel topologies (Oh et al. 1994, da Silva and Dawson 1997); however, the system is not able to verify its design proposal by a simulation.

Remarks. There exist two concepts of how a problem's solution can be defined: One of them codes the problem solving *process*, the other codes the *result* of a problem solving process, for example in the form of a system description S. From this distinction result two analogy concepts in CBR, namely that of derivational analogy (belonging to the former) and that of transformational analogy (belonging to the latter) (Carbonell 1986, Goel and Chandrasekaran 1989, Hinrichs and Kolodner 1991). For reasons of clearness, the considerations of this paper are oriented at the latter, i. e., at the system description view, but they could be reformulated to the process-centered view as well.

Definition 2.1 (Case, Case base, Query). Let $\mathbf{D}$ be a set of demand sets, and let $\mathbf{S}$ be a set of systems. A *case* C is a tuple $C = \langle D, S \rangle$, $D \in \mathbf{D}, S \in \mathbf{S}$, where S constitutes a solution for D. A set CB consisting of cases is called a *case base*. A case of the form $q = \langle D, \cdot \rangle$ is called *query* or problem definition to a case base.

When given a query $q = \langle D, \cdot \rangle$ to a case base CB, two jobs must be done to obtain a solution to q. (*i*) Retrieval of a similar case c, and (*ii*) adaptation of c such that D is fulfilled.

Weß (1995) mentions three approaches to define similarity: Similarity based on predicates, similarity based on a preference relation, and the most generic concept, similarity based on a measure. In connection with design problem solving, only the last is powerful enough, and the following definition will formalize a similarity measure for design case bases.

Definition 2.2 (Case Similarity, Similarity Measure). Given is a symmetric function $\sigma : \mathbf{D} \times \mathbf{D} \to [0; 1]$, which additionally has the reflexivity property, $\sigma(D_1, D_2) = 1 \Leftrightarrow D_1 = D_2$. Moreover, let $c_1 = \langle D_1, S_1 \rangle$ and $c_2 = \langle D_2, S_2 \rangle$, c_1, $c_2 \in CB$, be two cases. Then the *case similarity* $sim : CB \times CB \to [0; 1]$ is defined by means of σ in the following way: $sim(c_1, c_2) = \sigma(D_1, D_2)$; σ is called a similarity measure.

Remarks. (*i*) The semantics of σ shall be as follows. The more similar two demand sets D_1 and D_2 are, the larger shall be their value $\sigma(D_1, D_2)$. (*ii*) The symmetry property guarantees that $sim(c_1, c_2) = sim(c_2, c_1)$; the reflexivity property defines the self-similarity of a case.

3 CASE-BASED DESIGN OF FLUIDIC SYSTEMS

Typically, a case-based reasoning approach to a design problem is realized in a monolithic manner—by mapping a complex set of demands, D, directly onto a system $S \in CB$. This approach is absolutely futile here since a case base CB that provides adequate solutions for the entire variety of fluidic demand sets can neither be set up nor maintained.

In contrast to such a monolithic view, the presented approach is grounded on the principle of "functional composition" (Stein 1996). The principle says that

1. each set of demands, D, can be decomposed into a set of functions, $F = \{f_1, \ldots, f_n\}$,
2. each function $f \in F$ can be mapped one to one onto a hydraulic axis that realizes f,
3. the coupling type between the hydraulic axes (series, parallel, sequential, etc.) can be derived from D.

While the first point goes in accordance with reality, point 2 and point 3 imply that no subfunction f is realized by either a combination of several axes or by constructional side effects.

Under this working hypothesis a demand set D can be transformed towards a fluidic system S within two steps: by designing a fluidic axis A for each f implied by D, and by coupling these axes in a qualified way. Figure 2 depicts this view.

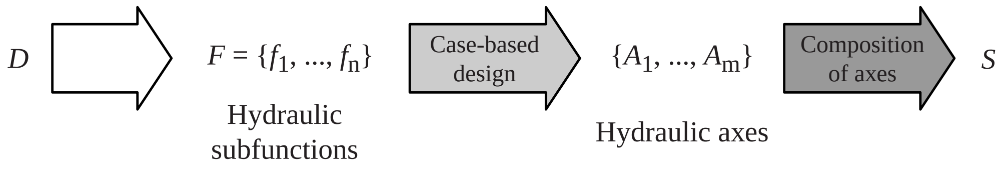


Figure 2: Automating fluidic circuit design by functional composition.

Taking this simplifying view of the design process, the step $F \longrightarrow \{A_1, \ldots, A_m\}$ can be realized by CBR methods—provided that the following can be developed: a similarity measure for hydraulic functions and an adaptation concept for hydraulic axes.

The following subsections introduce the case-based design approach in greater detail. We start by illustrating the three before-mentioned abstraction levels of fluidic design problems, D, F, and S. The next but one subsection develops a similarity measure for functions $f \in F$. This measure is vital to realize the retrieve step in the CBR approach: For a given f it identifies the most similar fluidic axis A from a case base of axes. The subsection thereafter is devoted to the revise step: It is shown in which way a misfitting axis can be adapted. The last subsection describes the step $\{A_1, \ldots, A_m\} \longrightarrow S$, i. e., it is shown how the retrieved and adapted axes are connected to a system.

3.1 Abstraction Levels of Fluidic Design Problems

A fluidic design problem can be described at different levels (layers) of abstraction. From the standpoint of a design process the following layers are important: the demand layer, which defines the desired set of demands D, the functional layer, which defines the implied function F, and the component layer, which defines the system S.

Demand Layer, D. The layer of demands contains the entire specification for a fluidic system. Vier et al. (1997) discuss possible demands in detail, such as tolerance constraints, operating restrictions, boundary values, etc. Central elements of D, however, are the cylinder profiles, which prescribe the courses of the forces, the velocities, or the pressure. Implicitly, these profiles characterize particular phases of the working process, such as a speed phase, a slowing down phase, or a press phase.

Figure 3 shows cylinder profiles for a hydraulic system that operates in the low pressure range and that contains two working elements, w_1, w_2, which have to perform a combined manipulation and pressing task.

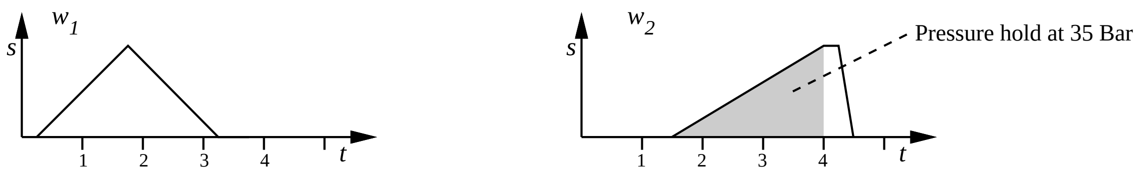


Figure 3: Desired cylinder profiles for a hydraulic system.

Functional Layer, F. A functional view can be derived from D by identifying the working phases within the cylinder profiles, by globally arranging these phases, and by combining them within functions f. Typically, each function $f \in F$ is realized by a fluidic (here: hydraulic) axis. The functional layer specifies these axes as well as their couplings according to the global phase interplay, say, the order restrictions of the phases.

Figure 4 shows the functional layer that corresponds to the demand layer of Figure 3. Here, four phases have been identified (see Table 1), globally arranged, and combined within two functions, f_1 and f_2. The respective hydraulic axes must be coupled sequentially to realize D—a fact which is reflected by the coupling hierarchy.

Working element	Phase	Phase Type
w_1	$P_{1.1}$	Constant drive: $0.5\,m/s$
w_1	$P_{1.2}$	Constant drive: $-0.5\,m/s$
w_2	$P_{2.1}$	Pressure hold: 35 Bar
w_2	$P_{2.2}$	Driving in

Table 1: Phases identified in the cylinder profiles.

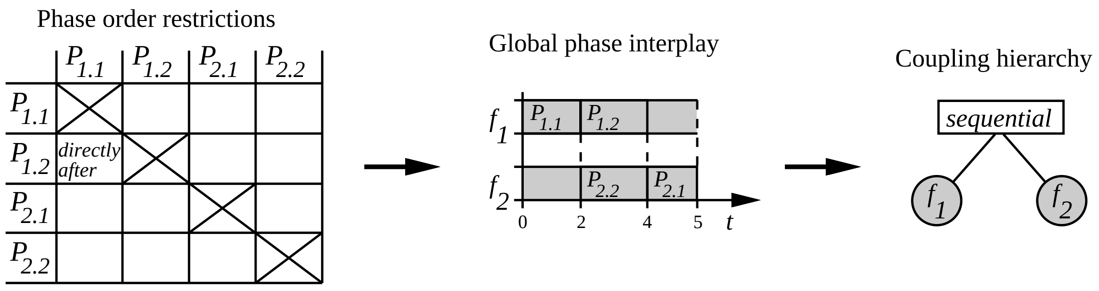


Figure 4: Corresponding functional description for the above cylinder profiles.

Component Layer, S. The component layer defines the structure and all components of the fluidic system. They form, along with the component parameters, the solution of the design problem. For each function of the functional layer there is a hardware counterpart in the form of a fluidic axis. These axes are coupled according to the coupling hierarchy.

Figure 5 shows a component layer that corresponds to the functional layer of Figure 4. For each of the functions, f_1, f_2, a hydraulic axis (A_1 and A_2) is given.

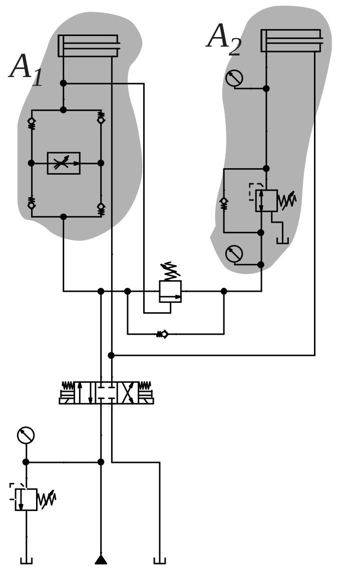


Figure 5: Circuit that realizes the functional description of above.

3.2 A Similarity Measure for Hydraulic Functions

The desired demands, D, at a hydraulic system imply a set of hydraulic subfunctions, $\{f_1, \ldots, f_n\}$, each of which to be realized with a particular hydraulic axis A. Supposed there is a case base, CB, with cases of the form $\langle f, A \rangle$, and a query, $\langle f_d, \cdot \rangle$, for which a suited hydraulic axis shall be retrieved from CB. Then a mapping, σ, with the following properties is required.

Phase type	POSITION	CONSTANT	ACCELERATE	HOLD.POS	PRESS	FAST	HOLD.PRESS
POSITION	1	0.3	0.5	0	0.3	0.7	0
CONSTANT	0.3	1	0	0	0.7	0.7	0
ACCELERATE	0.5	0	1	0	0.2	0.4	0
HOLD.POS	0	0	0	1	0.3	0.0	0.6
PRESS	0.3	0.7	0.2	0.3	1	0.0	0.8
FAST	0.7	0.7	0.4	0.0	0.0	1	0.0
HOLD.PRESS	0	0	0	0.6	0.8	0.0	1

Table 2: The similarity between two phase types, σ_{pt}.

1. σ is defined on the domain $\mathcal{F} \times \mathcal{F}$, where $\mathcal{F}$ is a set that comprises the possible hydraulic functions f.
2. σ is a similarity measure as defined on page 3.
3. "$\sigma(f_d, f_r) = \max\{\sigma(f_d, f) \mid \langle f, A\rangle \in CB\}$" $\Leftrightarrow$
   "$\langle f_r, A_r\rangle \in CB$ is the best case in CB to realize f_d".

Remarks. The similarity measure σ estimates two hydraulic functions respecting their similarity. It maps from the Cartesian product of the hydraulic functions domain onto the interval $[0; 1]$. The last property, 3, postulates that σ should become maximum only for those cases in CB that realize the demanded function f_d best.

The elementary characteristic of a hydraulic function f is defined by both the sequence and the type of its phases. Valuating two hydraulic functions respecting their similarity thus means to compare their phases. However, the phases in turn are characterized by their type, duration, distance, force, or precision, and each of which can be quantified by a special phase similarity measure. Table 2 gives a sample quantification for σ_{pt}, the similarity of two phases' types, which is the most important phase characteristic and which is used in the following definition.

Definition 3.1 (Similarity Measure for Hydraulic Functions Based on Phase Types). Let the hydraulic function f_1 designate the phase sequence $(p_{1_1}, \ldots, p_{1_n}) \in \mathcal{F}$, and let the hydraulic function f_2 designate the phase sequence $(p_{2_1}, \ldots, p_{2_m}) \in \mathcal{F}$ for some $n, m \in \mathbf{N}$. A phase type based similarity measure for hydraulic functions, $\sigma : \mathcal{F} \times \mathcal{F} \to [0; 1]$ is defined as follows:

$$\sigma(f_1, f_2) = \frac{1}{\max\{n, m\}} \sum_{i=1}^{\min\{m,n\}} \sigma_{pt}(p_{1_i}, p_{2_i})$$

Remarks. Obviously does σ fulfill the conditions of a similarity measure. Note that in the case $n \neq m$ not all phases will match and hence not all phases are considered in the computation of σ. A fact, which does reflect our comprehension of similarity in most cases—it does not if, for instance, f_1 and f_2 are described at different levels of detail. This and other shortcomings have been addressed in the work of Hoffmann (1999): He proposed and realized an algorithm that enables a smart matching between phase sequences varying in length.

3.3 Adapting Fluidic Axes

By means of the above similarity measure for hydraulic functions, σ, the k most similar cases can be retrieved from a case base of hydraulic axes given a query $\langle f_d, \cdot\rangle$. Note, however, that these cases merely form solution *candidates*; usually, a retrieved case must be adapted to fulfill the demanded hydraulic function f_d. Case adaptation plays a key role in case-based design (Kolodner 1993) and is performed within the reuse step (bring to mind Figure 1 again).

The following definition specifies the terms “modification” and “adaptation”. While each adaptation represents a modification, the inverse argumentation does not hold: A modification of some case does establish an adaptation only, if the modified object of the case—in our setting a modified hydraulic axis A'—does fulfill the demanded function f_d to a higher degree than the unmodified axis A of the original case.

Definition 3.2 (Modification, Adaptation). Let $c = \langle f, A\rangle \in \mathit{CB}$ be a case, and let $q = \langle f_d, \cdot\rangle$ be a query. A *modification* of c respecting q is a function $\mu : \mathcal{F} \times \mathit{CB} \to \mathcal{F} \times \mathcal{A}$, with $\mu(f_d, c) = \langle f', A'\rangle$. A modification of c is called an *adaptation* of c if the following condition holds:[2]

$$sim(\langle f', A'\rangle, q) > sim(\langle f, A\rangle, q)$$

Case adaptation can be realized in different ways. A popular approach is the formulation of adaptation knowledge in the form of (heuristic) modification rules (Stein and Vier 1998, Barletta and Hennessy 1989, Hennessy and Hinkle 1991). In technical domains where the behavior of the system to be adapted is well understood, a particular type of modification rules, called “scaling” rules here, can be employed to encode modification knowledge.

Definition 3.3 (Scale Function, Scalable, Scaling). Given is a query $q = \langle f_d, \cdot\rangle$, a subset of the demanded hydraulic function $f'_d \subseteq f_d$, and a case $c = \langle f, A\rangle \in \mathit{CB}$. A function $scale : \mathcal{F} \times \mathit{CB} \to \mathcal{F} \times \mathcal{A}$ is called *scale function* of c respecting f'_d, if the following conditions hold:

(*i*) $scale(f'_d, c) = c' = \langle f', A'\rangle$ with $f'_d \subseteq f'$, and

(*ii*) $sim(c', q) > sim(c, q)$

c is called *scalable* with respect to f_d, c' is called *scaling* of c.

Remarks. In other words, with respect to a part of the desired function, $f'_d \subseteq f_d$, there is a case $c = \langle f, A\rangle$ in the case base whose hydraulic axis A can be modified—say: scaled—towards A' in such a way that A' provides f'_d and c' is more similar to q than is c. Obviously does each scaling establish an adaptation.

Example. Consider the design of a lifting hoist where $c = \langle f, A\rangle$, the most similar case found respecting the query $q = \langle f_d, \cdot\rangle$, does not fulfill the maximum force constraint $(F_{\text{cyl}}, x_d) \in f_d$. Given this situation, c can be scaled up to fulfill f_d if the force difference between the existing and the desired system is of the same order of magnitude (see Figure 6).

In this simple example the scaling of the force is possible since the responsible underlying physical connections, the balance of forces, can be quantified. A reasonable scale function could utilize this law as follows. It adapts the force value x of c according to the demanded value x_d by scaling the piston area, A_{cyl}, to a new value with respect to the maximum pressure allowed, $p_{\max}$.

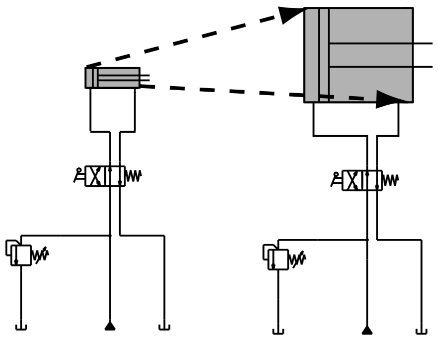


Figure 6: Scaling a cylinder respecting a desired force.

[2]The condition is equivalent to the following: $\sigma(f', f_d) > \sigma(f, f_d)$

Formally, the scale function takes two arguments (recall Definition 3.3); the first of which defines the subset of f_d to be achieved by scaling, the second is the case to be modified:

$$scale(\{(F_{\text{cyl}}, x_d)\}, c) = c' = \langle f', A' \rangle \in \mathcal{F} \times \mathcal{A}, \text{where}$$

$$f' = f_d \setminus \{(F_{\text{cyl}}, x)\} \cup \{(F_{\text{cyl}}, x_d)\},$$
$$A' = A \setminus \{(A_{\text{cyl}}, y)\} \cup \{(A_{\text{cyl}}, y')\}, \quad \text{with } y' = \frac{x_d}{p_{\max}}.$$

Note that condition (*ii*) of Definition 3.3 is fulfilled: The similarity between the scaled case c' and the query q is strictly larger than the similarity between the original case c and q.

3.4 Coupling Fluidic Axes

This subsection outlines how several hydraulic axes are combined towards a system S. Recall that the starting point of a design problem is a demand set, D, which implies a set of hydraulic subfunctions, $F = \{f_1, \ldots, f_n\}$. A retrieve step yields a hydraulic axis A_f for each function $f \in F$.

Axes can be connected by means of a parallel coupling, a series coupling, or a sequential coupling. The result of such a coupling can be considered as a new hydraulic axes which in turn can be connected to other axes. This way, coupling hierarchies can be defined in recursive manner. To automate the generation of a connecting network between several axes, we introduce four generic building blocks: (*i*) an axis building block with a single input and a single output, (*ii*) a control building block with two inputs and two outputs, (*iii*) a coupling building block with two inputs and two outputs, and (*iv*) a service building block with a single input and a single output.

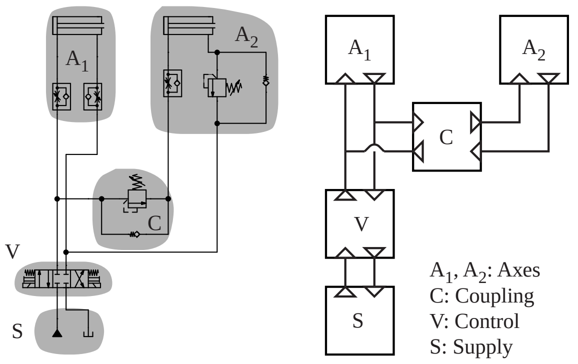


Figure 7: Circuit diagram and related building block representation.

Figure 7 shows a hydraulic system and its appropriate building block representation; Figure 8 shows the three coupling types.

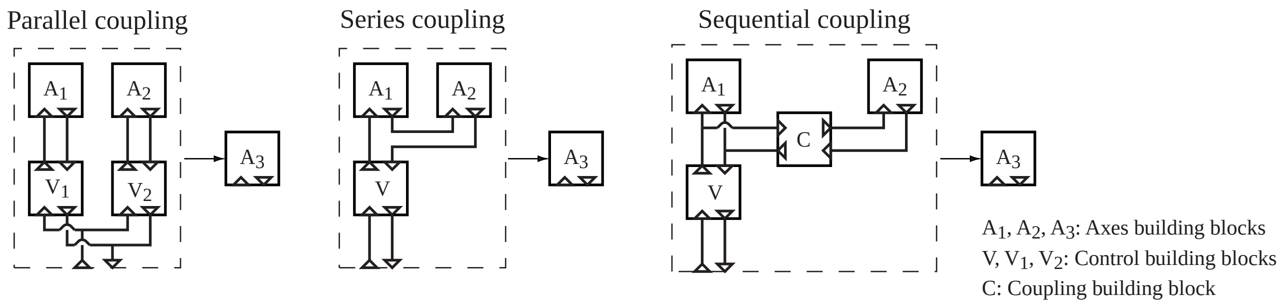
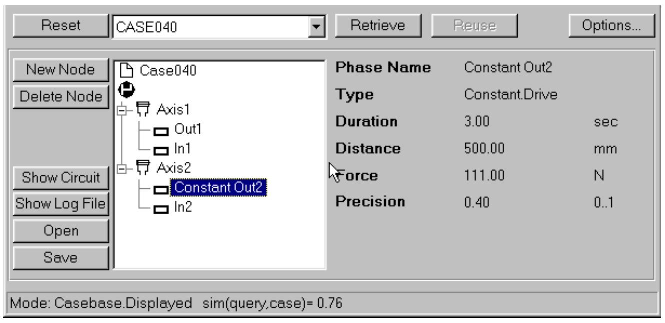

Parallel coupling
Series coupling
Sequential coupling
A1
A2
V1
V2
A3
A1
A2
V
A3
A1
A2
C
V
A3
A1, A2, A3: Axes building blocks
V, V1, V2: Control building blocks
C: Coupling building block

Figure 8: The three coupling types in their building block representation.

4 REALIZATION

The concepts described in the previous section have been embedded within a design assistant[3] and linked to FluidSIM, our drawing and simulation environment for fluidic systems (Stein 1995, Stein et al. 1998). The design assistant enables a user to formulate his design ideas by specifying both a set of functions F and a coupling hierarchy between the elements in F. For each $f \in F$ a sequence of phases can be defined, where for each phase a set of characteristic parameters, such as duration, precision, or maximum values can be stated. Figure 9 shows the interface part of the design assistant that realizes the specification of F; this front end is used for the acquisition of new cases as well as for the formulation of queries.



Phase Name	Constant Out2	
Type	Constant.Drive	
Duration	3.00	sec
Distance	500.00	mm
Force	111.00	N
Precision	0.40	0..1

Mode: Casebase.Displayed sim(query,case)= 0.76

Figure 9: Interface for design queries.

Having started the retrieval mechanism of the design assistant, the case base is searched for the hydraulic axes fitting best the specified function. In a next step these building blocks are automatically scaled and composed towards a new system. Finally, a drawing for the circuit is generated, which directly can be simulated and evaluated respecting the desired demands using FluidSIM. Figure 10 shows a query (left window), the functional description of the generated design (middle window), and the hydraulic drawing of the generated design.

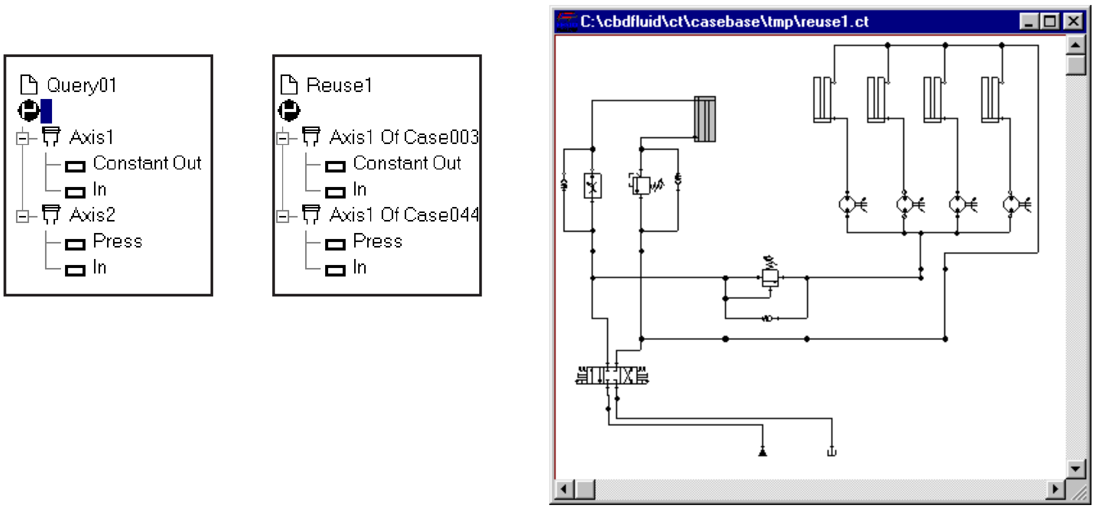


Figure 10: A design query, the functional description of a solution, and the related drawing.

4.1 Evaluation

Clearly, a direct evaluation of generated design solutions must be limited within several respects since

1. an absolute measure that captures the quality of a design does not exist, and

2. the number of properties that characterizes a design is large and their quantification often

[3]The design assistant has been realized and evaluated as a part of the doctoral thesis of Hoffmann (1999).

Axes number	Retrieve	Reuse	$sim \geq 0,8$	$sim \geq 0,9$	Simulation O.K.	Expert modification
1	$<<$ 1s	0.10s	17	13	10	10x(+), 6x(o), 1x(–)
2	$<<$ 1s	0.63s	16	11	9	8x(+), 7x(o), 1x(–)
3	$<<$ 1s	0.91s	17	10	7	7x(+), 8x(o), 2x(–)
4	$<<$ 1s	1.43s	15	8	5	3x(+), 10x(o), 2x(–)
5	$<<$ 1s	2.00s	18	6	1	1x(+), 15x(o), 2x(–)

Table 3: Runtime results and modification effort for automatically generated designs.

requires a high effort.[4]

Anyway, the quality of a generated design can also be rated *indirectly*, by quantifying its "distance" to a design solution created by a human expert. In this connection, the term "distance" stands for the real modification effort that is necessary to transform the machine solution into the human solution. The experimental results presented in the following table describe such a competition. The underlying experiments have been performed by Hoffmann (1999); a more detailed discussion of evaluation concepts as well as related problems can be found at the same place.

Description of the table columns:

- *Axes number.* Describes the number of axes of each test set; a test set contains 20 queries.
- *Retrieve.* Average time of a retrieve step in seconds.
- *Reuse.* Average time of a reuse step in seconds.
- $sim \geq 0.8$. Number of generated designs with a similarity ≥ 0.8.
- $sim \geq 0.9$. Number of generated designs with a similarity ≥ 0.9.
- *Simulation O.K.* Number of generated designs whose simulation results fulfill the demands of the query.
- *Expert modification.* Evaluation of a human expert. In this connection a (+), a (o), and a (–) designate a small, an acceptable, and a large modification effort necessary to transform the machine solution into a solution accepted by the human expert.

Test environment was a Pentium II system at 450 MHz with 128 MB main memory; the operating system was Microsoft Windows NT 4.0.

5 SUMMARY

This paper deals with the automation of fluidic circuit design. Underlying working hypothesis is the principle of functional composition, which claims that a fluidic design process can be emulated by two transformation steps. The first step realizes the mapping from the demands on the hydraulic functions, $D \longrightarrow F$; the second step realizes the mapping from the hydraulic functions onto the hydraulic system, $F \longrightarrow S$.

[4]Characterizing properties include: number of components, reliability, cost, effort for setting into operation, effort for maintenance, degree of standardization, fault-proneness.

Main contribution of the paper is the presentation of concepts that operationalize the step $F \longrightarrow S$ by means of case-based reasoning methods. These methods have been operationalized within a design assistant and compared to design solutions from a human expert, where they showed fairly good success.

Clearly, the principle of functional composition is a simplified view to the fluidic design process since it neglects the holistic view of the human designer. As a consequence, an automatically generated design solution will often be suboptimum respecting the desired demands D. Anyway, following aspects should be considered:

1. The principle of functional composition makes an automation of the design process possible.
2. An automatically generated design can be used as a starting point for the human designer.
3. The presented case-based design approach is adaptive. The case base can be enlarged, it may accommodate more sophisticated solutions, and, as a consequence, the case-based design algorithm will improve in its behavior.

References

Aamodt, A. and **E. Plaza**. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AICOM* 39–59.

Barletta, R. and **D. Hennessy**. 1989. Case adaptation in autoclave layout design. *Proceedings: Case-Based Reasoning Workshop*, hg. von K. J. Hammond. Morgan Kaufmann Publishers. 203–207.

Brown, D. C. and **B. Chandrasekaran**. 1989. *Design Problem Solving*. Morgan Kaufmann Publishers.

Carbonell, J. G. 1986. Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. *Machine Learning: an Artificial Intelligence Approach*, hg. von R. Michalski, J. Carbonnel and T. Mitchell. Morgan Kaufmann. Los Altos, CA. 371–392.

Cunis, R., **A. Günter**, **I. Syska**, **H. Peters** and **H. Bode**. 1989. PLAKON—*An Approach to Domain-independent Construction*. Technical Report 21. BMFT Verbundsprojekt. Universität Hamburg, FB Informatik.

da Silva, J. C. and **D. Dawson**. 1997. The development of an expert system for hydraulic systems design focusing concurrent engineering aspects. *Proceedings: International Conference on Enginieering Design (ICED 97) Tampere*.

Fluidon GmbH 1992. *DSH plus Benutzerhandbuch*. Fluidon Gesellschaft für Fluidtechnik mbH. Aachen.

Goel, A. K. and **B. Chandrasekaran**. 1989. Use of device models in adaption of design cases. *Proceedings: Case-Based Reasoning Workshop*, hg. von K. J. Hammond. Morgan Kaufmann Publishers. 203–207.

Hennessy, D. and **D. Hinkle**. 1991. Initial Results from Clavier: A Case-Based Autoclave Loading Assistent. *Proceedings: Case-Based Reasoning Workshop*, hg. von R. Bareiss. Morgan Kaufmann Publishers. 225–232.

Hinrichs, T. R. and **J. L. Kolodner**. 1991. The roles of adaptation in case-based design. *Proceedings of AAAI-91*. Cambridge, MA: AAAI Press / MIT Press.

Hoffmann, M. 1999. *Zur Automatisierung des Designprozesses fluidischer Systeme*. eingereichte dissertation. University of Paderborn, Department of Mathematics and Computer Science.

Kolodner, J. 1993. *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann.

Leake, D. B. 1995. Case-Based Reasoning: Issues, Methods, and Technology.

Marcus, S. and **J. McDermott**. 1989. SALT: A Knowledge Acquisition Language for Propose-and-Revise Systems. *Artificial Intelligence* 39. 1–37.

Oh, V., **P. Langdon** and **J. Sharpe**. 1994. Schemebuilder: an integrated computer environment for product design. *Computer Aided Conceptual Design*. Lancaster International Workshop on Engineering Design.

Piechnick, M. and **A. Feuser**. 1994. MOSIHS – Programmsystem zur Simulation komplexer elektrohydraulischer Systeme. *AFK, Aachener Fluidtechnisches Kolloquium*. Mannesmann Rexroth GmbH, Lohr, Germany.

Stein, B. 1995. *Functional Models in Configuration Systems*. Dissertation. University of Paderborn, Department of Mathematics and Computer Science.

Stein, B. 1996. *Optimized Design of Fluidic Drives—Objectives and Concepts*. Technical Report tr-ri-97-189. University of Paderborn, Department of Mathematics and Computer Science.

Stein, B., **D. Curatolo** and **M. Hoffmann**. 1998. Simulation in FluidSIM. *SiWis '98. Workshop "Simulation in wissensbasierten Systemen"*, hg. von F. . d. G. ASIM Arbeitsgemeinschaft Simulation. *61*. 61. Fachgruppe 4.5.3 "Simulation und Künstliche Intelligenz" der Gesellschaft für Informatik. Paderborn.

Stein, B. and **E. Vier**. 1998. *Recent Advances in Information Science and Technology. Second Part of the Proceedings of the 2nd IMACS International Conference on Circuits, Systems and Computers, CSC '98*. Second Part of the Proceedings of the 2nd IMACS International Conference on Circuits, Systems and Computers, CSC '98. World Scientific, London. chapter An Approach to Formulate and to Process Design Knowledge in Fluidics.

Vier, E., **B. Stein** and **M. Hoffmann**. 1997. *Strukturelle Formulierung von Anforderungen an hydrostatische Antriebe*. Technical Report MSRT 8/97. Gerhard-Mercator-Universität - GH Duisburg, MSRT.

Weß, S. 1995. *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik / Grundlagen, Systeme und Entscheidungen*. Dissertation. Universität Kaiserslautern.