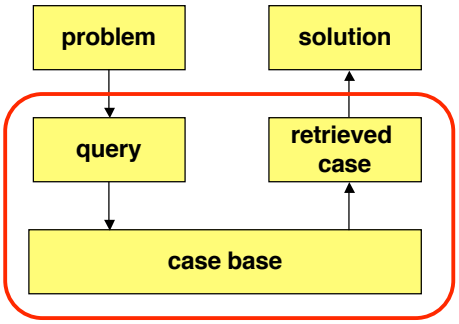
Case-based Reasoning for Diagnosis

1

Case-Based Reasoning



CBR system

Design aspects:

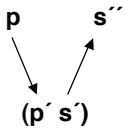
- modelling
- retrieval
- adaptation
- reasoning

2

What is a "Case"?

A case in CBR is a description of a problem together with its solution.

Principle of CBR:
(p = problem, s = solution)



In diagnosis:

p = description of failure situation, disturbed process
s = description of diagnosis steps, identified cause, therapy

3

History of CBR

"I have but one lamp by which my feet are guided, and that is the lamp of experience. I know no way of judging of the future but by the past."

Patrick Henry, American politician, 1775

1972	Tulving: Episodic and Semantic Memory
1977	Schank & Abelson: Scripts, Plans, Goals and Understanding, Lawrence Erlbaum
1987	Riesbeck & Bain: A Methodology for Implementing Case-Based Reasoning Systems
1989	Hammond: Case-Based Planning: Viewing Planning as a Memory Task, Academic
1993	Kolodner: Case-Based Reasoning, Morgan Kaufmann
1993	First European Workshop on CBR, Springer

4

Some CBR Applications

Classification e.g. of archeological objects

Diagnosis e.g. of machine failures

Planning e.g. of repair tasks

Construction e.g. of buildings

Case-based support for human decision making

Case-based information retrieval

Case-based help-desk services

Experience-based image interpretation

...

5

Diagnosis Support for a Flexible Manufacturing System

Case study carried out for an aircraft production company in Germany

Goal: Reduce breakdown times of large milling machines (FMS) for aircraft part production

Approach: Mutate failure reports into case descriptions for CBR diagnosis support

6

Example of Failure Report for FMS

Failure:	**When starting the spindle, the fuses of the drive amplifier frequently blow**
Hypothesis:	**Tachometer is faulty (clutch or bearing)**
Test:	**Disassembly and test of tachometer, failure remains**
Hypothesis:	**Faulty component in drive amplifier**
Test:	**Exchange of components, failure remains**
Hypothesis:	**Faulty thyristors**
Test:	**Function test of thyristors ok**
Hypothesis:	**Faulty field rectifiers**
Test:	**Function test of field rectifiers not ok, exchange of field rectifiers, failure disappears**

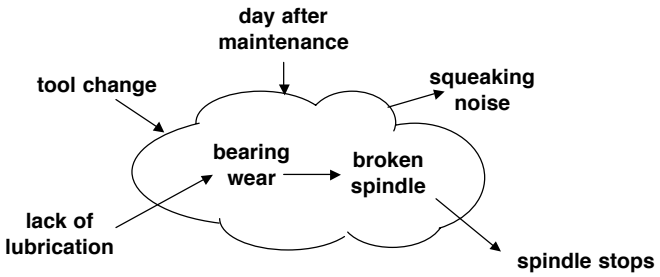
7

Complete Failure Descriptions

<u>In principle</u>:

failure description = situation description including all information which may be *connected* with failure

"Connected" in the sense of an (unknown) cause-effect graph:



8

Formalizing Observations

Failure description in terms of observations which partially specify space-time around a failure

Specification of an observation based on 4 questions:

Where?	**<component specification>**	*"main spindle"*
What?	**<property or behavior specification>**	*"noise"*
How?	**<quality or value specification>**	*"squeaking"*
When?	**<time specification>**	*"during start-up phase"*

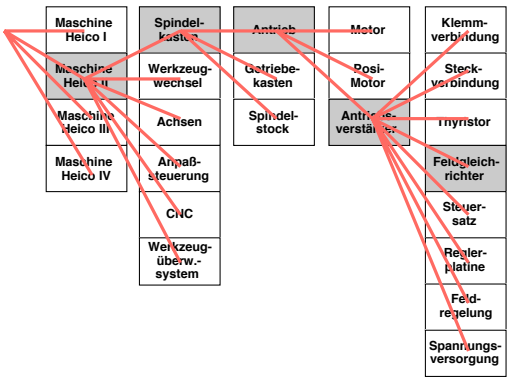
"There is squeaking noise at the main spindle during the start-up phase"

9

Component Specification

Specifying the "where" of a failure by naming components

Coarse-to-fine component specification based on hierarchical structure



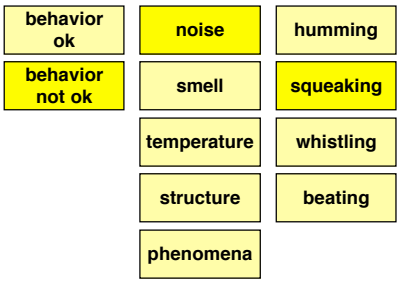
10

Property Specification

Specifying the behavior of components by

- attribute-value pairs
- properties (= merged attribute-value pairs)

Example for hierarchical property specification:



11

Time Specification

Time specification relative to known time points or operational phases of the FMS

Temporal relations:

before, after, during, frequently, sometimes following, ...

Reference times:

Monday 13.5.2000, tool change, start-up phase, maintenance, ...

Comparison of time specifications requires temporal reasoning

12

Semantics of Natural Language Terms

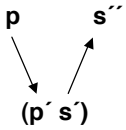
Terms are structured according to basic semantic relations:

- IS	synonym relation (classes or individuals)
- ISA	subsumption relation (classes)
- PART-OF	component-aggregate relation (classes or individuals)
- INSTANCE	membership relation (individual in class)

⇒ support for retrieval and adaptation

13

Intended Logics for CBR



- P = class of problems with description p
- P´(p´) = class of problems with description p´
- S´(s´) = class of solutions with description s´
- some problems in P´ have a solution in S´
- P and P´ have common instances
- s´´ is an instance of S´ "adapted" to p

In general, P´ will also have instances with solutions not in S´ !

Example:

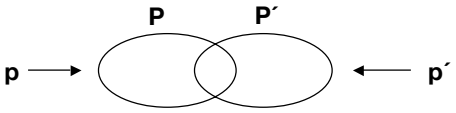
P´= "machine does not work" S´= "turn on main switch"

s = " main fuse blown" $s \notin S$

14

Case Retrieval

Given a current problem p, retrieve a *compatible* case p´ such that P and P´are not mutually exclusive, i.e. may share a common case.



Example:

Current case: *main drive does not work properly*

Retrieved case: *clutch slips*

Current case description has common instances with retrieved case description because

clutch PART-OF *main_drive*

slips ISA *not_work_properly*

15

Retrieval Algorithm

Assumption: No news is good news

If a component is not included in a case description, its behavior is ok.

Two case descriptions p and p´ are compatible if no observation o of p is incompatible with any observation o´ of p´.

Two observations

o = (Where, What, How, When)

o´ = (Where´, What´, How´, When´)

are incompatible, if

- Where AND Where´ are satisfiable, and
- What AND What´ are satisfiable, and
- When AND When´ are satisfiable, and
- How AND How´ are not satisfiable.

16

Semantic-based vs. Similarity-based CBR

Similarity-based case retrieval:

- "flat" case descriptions with features (attribute-value pairs)
- similarity is based on weighted distances between corresponding features

Semantic-based case-retrieval:

- natural language based
- descriptions with flexible specificity
- clear retrieval concept
- no obvious ranking scheme

17

Conclusions

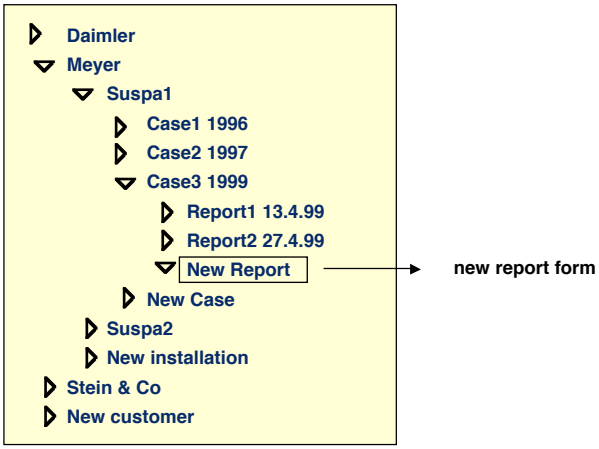
- CBR is a intuitive problem solving method.
- Problem descriptions for CBR diagnosis support can be structured in a uniform way based on where-what-how-when information.
- The semantics of natural language terms can be exploited for "intelligent" case retrieval.
- Typically, case descriptions are unspecific, hence a CBR system is bound to propose wrong solutions.

18

Case study AMS

Developing an "Application Management System" for the employees of an industrial lubrications supplier

Case structure:



19

Report Form

DATE:	13.5.99		
OPERATOR:	Hans Meyer		
INSTALLATION:	Suspa1		
OBSERVATIONS			
work_piece	material	aluminum	4.4.99
work_piece	surface	sticky	4.4.99
lubrication	type	C.P. 288/08	since 15.1.99
HYPOTHESIS			
Low MV			
ACTION			
Increase MV to 11%, add 1.5 l 988/67			
EFFECT			
work_piece	material	aluminum	5.7.99
work_piece	surface	OK	5.7.99

20

Model-based Configuration

21

Configuration Problems

What is a configuration problem?

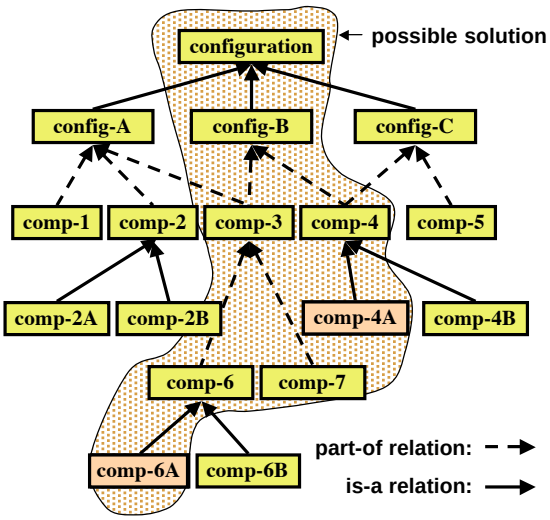
Construct an aggregate (a configuration) given
- **generic descriptions of parts**
- **compatibility constraints between parts**
- **a concrete task description**

Model-based configuration:

The configuration process is based on a declarative model of possible configurations.

22

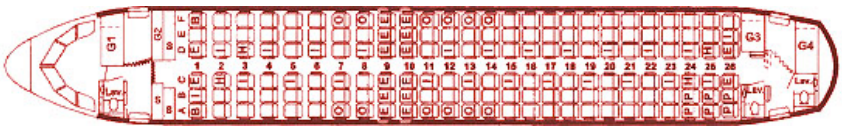
Illustration of Configuration



- boxes (frames) specify aggregate and component properties
- has-part relations bind components to aggregates
- is-a relations describe variants of entities
- constraints between entities (not shown) restrict choices and parameter combinations

23

A Real Configuration Task



Placement of cabin equipment (seats, kitchens, toilets, etc.) in view of

- customer wishes
- technical constraints
- legal constraints
- optimality criteria

24

What are the Logics of Configuration?

Domain knowledge:

What components belong to an aggregate?
What are the properties of components?
Which constraints must always be satisfied?

→ **axiomatization of domain**

Task description:

Construct a configuration meeting the general requirements of the domain and specific requirements of a customer

→ **logical model construction**

A logical model can be constructed as a consistent instantiation of the knowledge base.

25

Example

Domain knowledge:

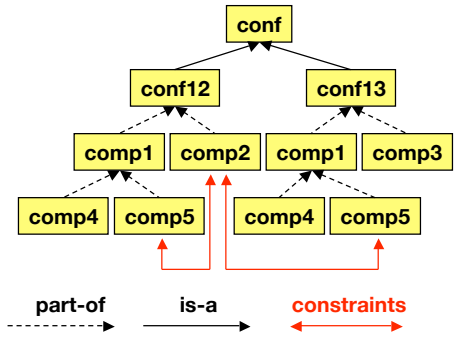
conf(Z) <- comp1(X) ∧ comp2(Y) ∧ part(X, Z) ∧ part(Y, Z).
conf(Z) <- comp1(X) ∧ comp3(Y) ∧ part(X, Z) ∧ part(Y, Z).
comp1(Z) <- comp4(X) ∧ comp5(Y) ∧ part(X, Z) ∧ part(Y, Z).
contains(X, Y) <- part(Y, X).
contains(X, Y) <- contains(X, Z) ∧ part(Y, Z).
false <- conf(X) ∧ contains(X, Y) ∧ contains(X, Z) ∧ comp2(Y) ∧ comp5(Z).

Customer requirements:

conf(a).
comp4(b).
contains(a, b).

Finding a consistent instantiation:

part(c, a).
comp1(c).
part(d, a).
comp3(d).
part(b, c).
comp5(e).
part(e, c).



26

Example of Concept Definition in KONWERK

KONWERK is a configuration system prototype developed at the AI Lab (LKI) of Hamburg University in 1986 - 1994. The commercial system Engcon has been developed based on KONWERK.

Concept "galley" describes service station in Airbus A340

```
def-concept
   :name galley
   :super-concept {cabin-interior-component rectangle}
   :parameters
       ref-nr [integer 2531000 2533999]
       door {1 2 4}
       trolleys {0 2 3 4 5 6 7 8 9 10}
       half-size-trolleys {0 1 2 3 4 5}
       meals [integer 28 140]
        type {longitudinal transversal}
       height {full half} (default 'full)
   :relations
       part-of [passenger-class]
```

27

Example: Concept Definition

```
(ist! (eine Klasse)
      (ein Konstruktionsobjekt
      (Teil-von    (eine Passagierkabine)
      (Hat-Teile   #{#[(ein Einrichtungsgegenstand) 0 433] ':=
                   #[(eine Küche) 0 10]
                   #[(eine Toilette) 1 12]
                   #[(ein Flugbegleitersitz 1 16]
                   #[(ein Passagiersitz 5 395]})
      (Sitzabstand       [28inch 62inch])
      (Sitze/Reihe       {5 6 7 8 9})
      (Passagiere/Toilette     [15 60])
      (Passagiere|Flugbegleiter      [8 50])
      (Mahlzeiten/Passagier    [0 6])))
```

28

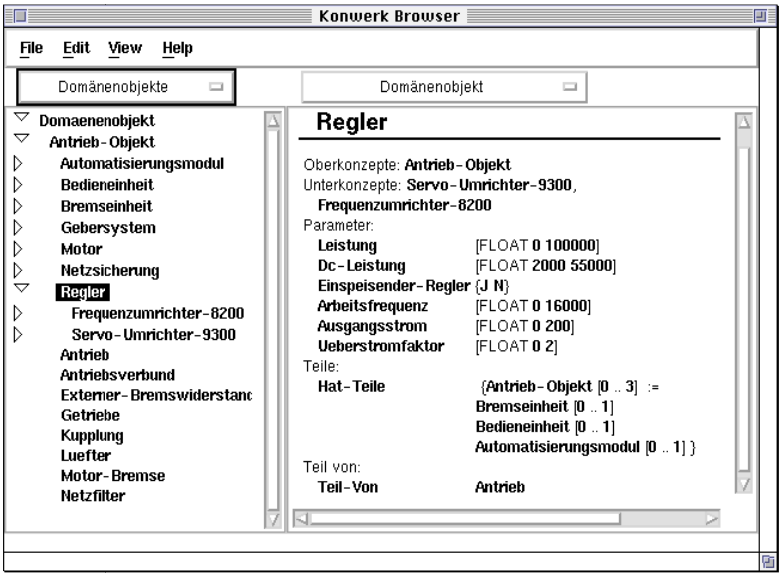
Object Descriptors in KONWERK

Object descriptors define object classes (concepts) by specifying possible instances. (Compare with concept expressions in a DL).

Specific values:	**red, 35t, car37**
Choice sets:	**{red yellow green black blue}**
Intervals:	**[10km/h 300km/h]**
Predicates:	**(:satifies evenp)**
Concepts:	**(a car)** **(a chassis (axle_load [10t 40t]))**
Atomic concepts:	**(a symbol (self {red yellow green black blue}))** **(a number (self [0 inf]))**
Logical operators:	**(:and [50 100] (:satisfies evenp))**

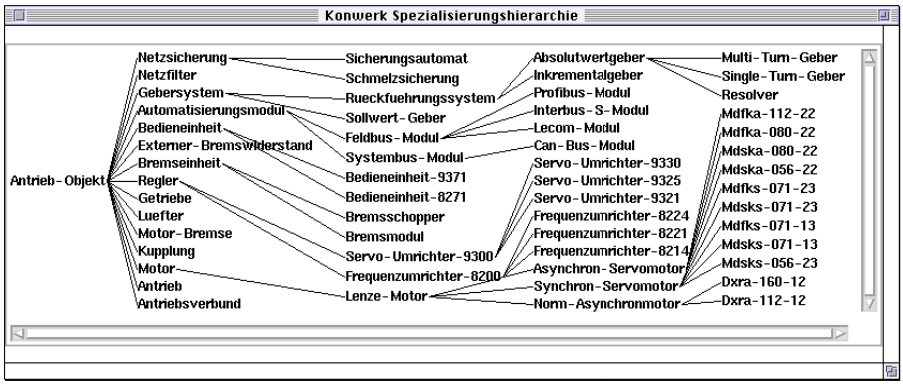
29

KONWERK Browser



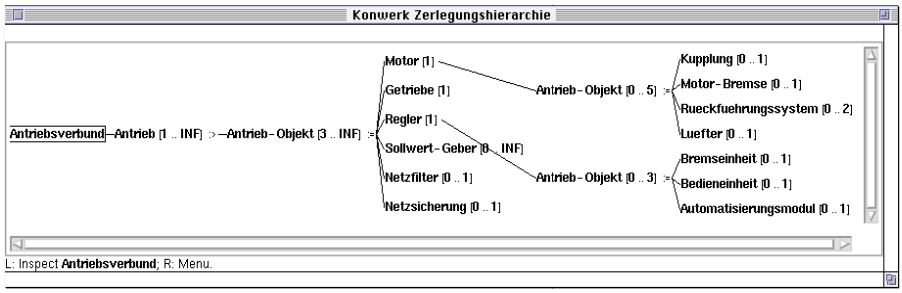
30

KONWERK Specialization Hierarchy



31

KONWERK Decomposition Hierarchy



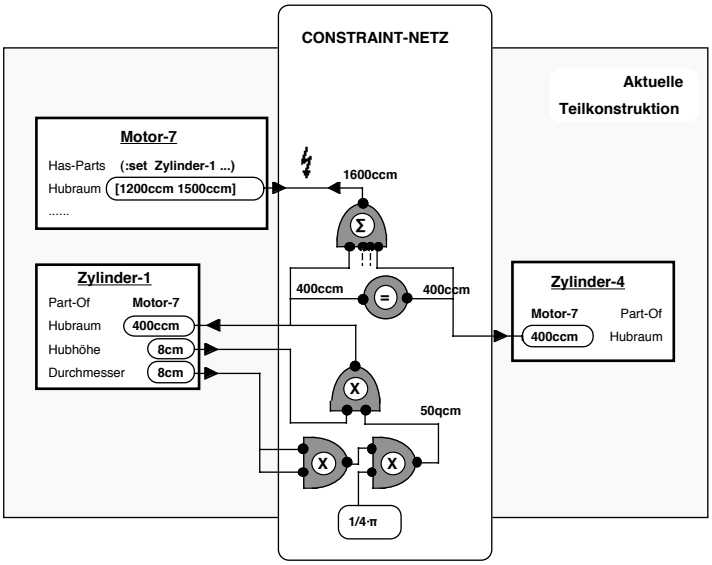
32

Constraints in KONWERK

- **Constraint classes**
  Predominantlydomain-independent computational procedures
  Examples: adder, multiplier, sum, equal
- **Conceptual constraints**
  Description of a domain-specific constraint type, instantiation rules
  Example: motor displacement = sum of cylinder displacements
- **Constraint instances**
  Dynamically generated at configuration time
- **Constraint net**
  Propagates values through all constraint instances, recognizes conflicts

33

Constraint Net in KONWERK



34

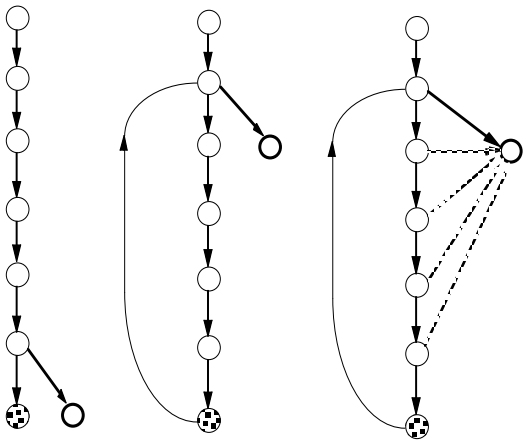
Central Configuration Cycle

1. Select a partial configuration
2. Determine agenda of executable configuration steps
3. Select a configuration step
4. In case of choices, use one of the processes:
   - default value assignment
   - computational procedure
   - user interaction
   - library solution
   - local breadth-first search
5. Constraint propagation and conflict recognition

35

Backtracking

In case of a conflict, backtracking occurs. One may select one of 3 backtracking strategies:



chronological backtracking

"intelligent" backtracking

"intelligent" backtracking with preservation of unaffected data

36

Conclusions

- Configuration is logical model construction
- The KONWERK configuration tool supports model construction by providing
  - an expressive object description language,
  - specialization and composition hierarchies,
  - a constraint system
  - declarative configuration control
- Application-oriented system development often occurs in ignorance of the logical interpretation of a task

37