# Incremental Adaptation for Conceptual Design in EADOCS

**B.D. Netten**[1], **R.A. Vingerhoeds**[1,2]

EADOCS (Expert Assisted Design of Composite Structures) is the implementation of a multi-level approach to conceptual design. Constraint-, case- and rule-based reasoning techniques are applied in different design phases to assemble and adapt designs at increasing levels of detail. This paper describes a strategic approach to decomposition, formulation of target design problems, and incremental retrieval and adaptation. Design problems considered, cannot be decomposed dynamically into tractable subproblems. Design cases are retrieved for requirements and preferences on both functionality and the solution. Cases are adapted in three phases: adaptation, modification and optimisation.

## 1 INTRODUCTION

EADOCS is an expert assisted conceptual design system for thin-walled fibre reinforced composite panels in aircraft structures. The conceptual design phase starts with the specification of design requirements, objectives and preferences. The result should be the "best" conceptual design. Subsequent phases for preliminary and detailed design are supported by numerical design tools, which require an initial conceptual design as a starting point for optimisation and analysis. Good initial designs are usually generated manually. EADOCS has been developed to support this conceptual design process for a specific application. The multi-level conceptual design process is supported by a more generally applicable hybrid system of constraint-, case- and rule-based reasoning.

This paper discusses the implementation of case-based reasoning and in particular the adaptation procedures in EADOCS. Terminology used in this paper is based on [1]. Relevant characteristics about the application and conceptual design approach are presented in section 2. Details are described in [2], [3], [4], and [5].

## 2 EADOCS

### 2.1 Problem specification and verification

The conceptual design problem is initially specified by requirements and preferences on functionality of the design solution, as well as on the solution itself. Preferences express the designer's experience and interpretation of feasibility and optimality of design solutions.

A functional requirement, specified by an object with threshold values, is a subproblem of the set of specifications. Multiple requirements can be specified for different operational conditions. The specifications are an aggregation of design subproblems, or objects for multiple classes of functionality. Figure 1 gives an example of several functional classes (square boxes) that can be specified. A functional class is defined for loading conditions with attributes for required normal and shear loads. For different operational conditions, different loading conditions can be specified as objects (rounded boxes), each of which must be carried by a single panel design.

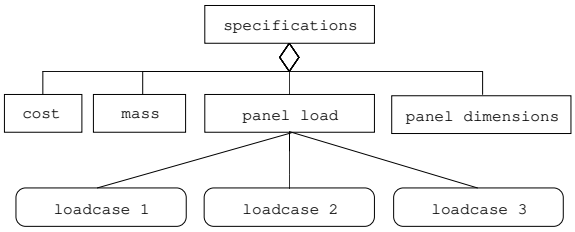


**Figure 1:** Design problem specification

Initial specifications can also contain preferences or requirements on part of the solution, for example by a restriction on materials. During the design process, additional selections are made on the solution to refine the remaining design problems.

The objective is to generate a conceptual design that best satisfies the specifications. A design model to generate a feasible solution from scratch is not available. This implies that a solution has to be generated, its behaviour analysed and evaluated against specified functionality, and adapted to optimise its functionality.

### 2.2 Abstraction and decomposition

#### *2.2.1 Solution decomposition*

Designs are represented hierarchically in an object-oriented data structure. Designs are segregated or decomposed into a structure of components for skin and stiffening, which can be further decomposed into layers, Figure 2. A component is specialised into classes of solutions, e.g. for panel types and layer materials. Panel designs are declared at two levels of abstraction:

[1] Delft University of Technology, Faculty of Technical Mathematics and Informatics, Julianalaan 132, 2628 BL Delft, Netherlands, [bart|rob]@kgs.twi.tudelft.nl

[2] University of Wales Swansea, Dept. of Electrical and Electronic Engineering, Singleton Park, Swansea SA2 8PP, United Kingdom

© 1996 B.D. Netten, R.A. Vingerhoeds
Workshop on Adaptation in Case-Based Reasoning
ECAI 96: *12th European Conference on Artificial Intelligence*

**Prototype design** is a qualitative representation of a design by solution classes, i.e. type of panel and stiffening, and classes of materials.

**Conceptual design** is a quantitative representation of a design by objects for the prototype solution classes with attribute values assigned.

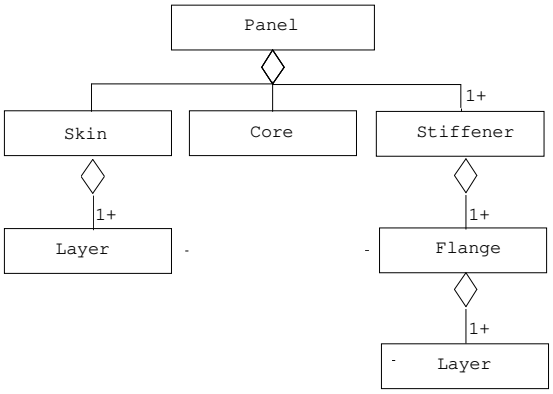


**Figure 2:** Solution decomposition

Design variables are defined by objects and attributes. Qualitative variables for prototype design are the solution classes. Many variables are non-numerical or discrete, and numerical analysis and optimisation routines can only be applied in the final phases of conceptual design.

### *2.2.2 Behaviour analysis and prediction*

The functionality provided by a solution depends on the behaviour of its components. Specified functionality cannot be directly predicted for a solution, or for components of the design. Classes of solutions are characterised by different behaviour. Different panel types, for example, have different modes of buckling behaviour. Each mode of buckling is defined in a different class, attributes and operations. Each class of solutions has a specific set of numerical analysis routines to determine its specific behaviour. The behaviour of a solution can only be determined from the behaviour of its components.

Numerical analysis routines require input of a conceptual design with all objects and attributes defined. Application of these routines is only possible in the final phase of conceptual design to improve and optimise a conceptual design.

Prediction of behaviour and functionality is essential for selecting and adapting solutions. However, the strong interdependencies between components, do not allow accurate prediction of component or design behaviour and functionality. Approximations can be applied to predict behaviour. Selections or adaptations could be abducted from these predictions. In EADOCS, however, heuristics are declared directly for prototype selection (section 2.3.1) and concept modification (section 5). These heuristics are based on behavioural approximations and semi-empirical relations. Predictions based on heuristics have to be verified by case retrieval or numerical analysis.

### *2.2.3 Problem decomposition*

In section 2.1, the initial design problem is specified by functional requirements and preferred solutions. Functional requirements are specified for a complete design or for specific components only. Indirectly, these requirements apply to all segregated components.

Specified functional requirements can be decomposed into subproblems for multiple components. Components are subjected to different specialised requirements. Each component should be designed and analysed for its particular set of subproblems. The (sub)solution of a component, however, will strongly affect other decomposed subproblems. Solving one subproblem requires reformulation of specialised subproblems for other components.

Design behaviour strongly depends on component behaviour. A subproblem cannot be uniquely related to a particular component in a design. Specifications cannot be decomposed into tractable subproblems for components. Dynamic decomposition for sequential retrieval and adaptation of subsolutions for subproblems cannot be applied, as for example in Déjà Vu [6] and PARIS [8].

At a higher abstractionlevel, a prototype solution has to be generated for the complete design. Components have to be designed simultaneously in following design phases. In EADOCS, decomposition is implicitly encoded in operations throughout the design process. Subproblems, for which heuristic solutions are known, are explicitly declared for behavioural categories or modifications.

## 2.3 Conceptual design phases

Design specifications, solutions and their behaviour are considered at two levels of abstraction for prototypes and concepts. These levels correspond to three design phases for respectively (Figure 3);

1. qualitative design of prototype solutions,
2. quantification of a prototype solution, and
3. quantitative adaptation and optimisation.

### *2.3.1 Prototype selection*

In the first design phase, the objective is to reduce the search space to feasible classes of solutions for the prototype design and components. Prototype solutions are assembled by selection of solution classes that can be composed and integrated into a feasible design. Optimality is finally estimated to select one or more alternative prototypes for quantification.

Prototype selection requires semi-qualitative reasoning about design problems and solutions. The selection process is formulated at the abstraction level of prototype designs:

- Functional requirements are categorised into typical problems with a qualitative interpretation. A functional category is defined by a pattern on values of attributes from a class of functionality. For each class of functionality, attribute values are initially discretised into intervals or subsets of values. Functional requirements are specified as typical categories or discretised onto these categories.

- Composition of prototype designs is determined by the class hierarchy. At each aggregation level one or more solutions can be selected from specialised classes for the components.

- Each solution class is characterised by a prototypical behaviour with respect to functionality, optimality criteria or integration with other solution classes. From theoretical and semi-empirical behavioural relations, design parameters can be determined to characterise behaviour. Design parameters are expressions from functional and solution attributes, and are also discretised as secondary features. Categories of behaviour can also be defined for classes of solutions, similar to functional categories. Relative performance can be expressed for each solution class and category as a qualitative constraint on prototypical behaviour.

Constrained-based reasoning can be applied to assemble prototype solutions for prototypical behaviour of its solution classes for required functionality, integration and optimality.

### *2.3.2 Concept selection*

The second design task is to quantify a selected prototype solution. Quantification requires the transformation of a qualitative solution into a quantitative solution that satisfies the specified requirements and optimality criteria. Due to the strong interdependence of component behaviour, such transformations cannot be declared for instantiation of solution objects, see section 2.2.

Quantification is performed by case retrieval and adaptation. The initial design problem for case-based retrieval is declared by primary and secondary features for functional and behavioural categories and prototype solution classes. Design cases are structured in objects for functionality, behaviour and solutions, similar to prototype and conceptual designs. Complete designs as well as their components can be retrieved as (sub)solutions. Implementation of case-based reasoning for concept selection will be described in section 3.

### *2.3.3 Concept modification and optimisation*

Adaptation of a retrieved solution in the previous design phase is based only on retrieved information. Retrieved functionality and behaviour cannot provide accurate predictions for behaviour current specifications. When the adapted solution is completely defined, it can be analysed numerically. Numerical behavioural results are significantly more accurate and reliable, and retrieved behaviour becomes obsolete.

Design variables that can be optimised comprise non-numerical, discrete and continuous variables. Before numerical optimisation can be performed, the non-numerical and discrete variables have to be modified. Modification is performed by specialist operations for repair and improvement (section 5).

## 3 CASE-BASED REASONING

Case-based reasoning is applied for the quantification of a prototype solution into an initial conceptual solution, see also Figure 3. The retrieval mechanism should be flexible for following reasons:

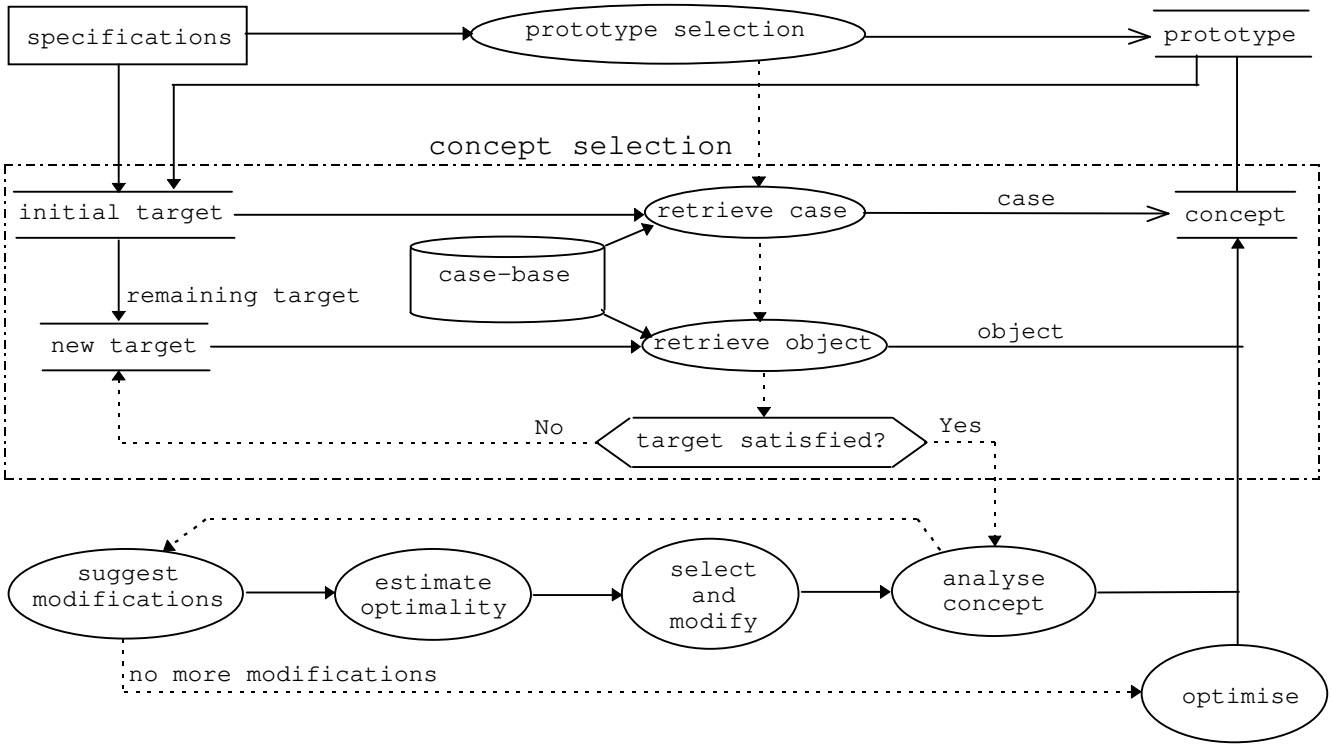


**Figure 3:** Conceptual design process

1. Design cases are obtained from previous design sessions in EADOCS, from other CAD systems or literature. The number of cases is small in relation to the number of alternative designs, and the distribution over the design space is strongly biased by previous design objectives. A case has been analysed only for its intended functionality. The solution can satisfy many other functional requirements as well, but this cannot be retrieved from the case. Only when similar prototypical cases exist, which have been designed for these other requirements, such functionality can be induced from multiple retrieved cases. In all other situations, accurate behaviour predictions are required.

2. The combination of specifications strongly determines the typical design solution of each case. The strong interdependence of component behaviour require solutions that are strongly determined by the specific functional requirements and preferred solutions. However, it is unlikely that the exact combination will be specified as a target for retrieval.

3. Many cases will match on some of the target specifications, and it is not possible to identify the appropriate case subsolutions for retrieval and integration into a new solution. Not every case or case part can be composed or integrated into a new solution. Initial filtering of solution classes before retrieval is required. In EADOCS, the prototype selection phase provides this filtering to guide the retrieval by the bounds on the prototype solution classes.

Targets for retrieval can be defined for the complete set of specifications, as well as for individual requirements or preferences. The source for retrieval is also specified by preferred or prototype solution classes.

Behaviour of components and their integration into a new solution should be predicted accurately for adaptation. Numerical analysis routines can only be applied successfully for initially adapted solutions. Adaptation by reuse of subsolutions, based on retrieved behaviour and functionality, should be limited to composition only. Adaptation of subsolutions for integration of behaviour should be postponed until a complete design has been analysed in the modification phases.

Conflicts in integration of components can be avoided initially by retrieval of a complete case design. Target and source for retrieval are defined by the complete set of specifications and the selected prototype solution. The initially retrieved design will match a subset of the target problems. For the remaining targets, minimal adaptations are retrieved in following iterations. This incremental retrieval and adaptation process will be described in following sections.

A case memory structure is created to support matching and selection of cases and objects. Intervals of attribute values are reused from prototype selection. Cases are indexed onto the class attribute intervals by their objects and attribute values. Each attribute interval is an index. The memory structure is an extension of the network structure described in [7]. This network is further developed for the domain model based on the object oriented case structure and categories. A case object is organised in a subnetwork for its class by its indices. The end nodes of each subnetwork identify the case-object. A collect network organises all objects to identify each case. This network has following features:

- Partial matching of any object and case with target objects for functionality, behaviour and solutions.
- Filter objects and cases for prototype solutions.
- Determine global similarity for objects and cases for each target separately.
- Determine changes in global similarity for any change in target or prototype.

The incremental retrieval and adaptation is implemented as a sequential process to simplify control structure. Although the memory structure performs previous tasks simultaneously for all objects and cases, simultaneous retrieval and adaptation have not been considered.

# 4 INCREMENTAL CASE ADAPTATION

## 4.1 Initial case retrieval

The objective is to retrieve an initial conceptual design as a starting point for incremental retrieval and adaptation of case parts. The initial design is the instantiation of the best matching case.

The target for initial retrieval is defined by objects for each functional requirement. Each target object is propagated through the network. Global similarity is determined by the matching score for each object and case. The prototype solution classes are bounds on the source on filter the matching score for retrievable cases and objects. Similarity is determined for each case, by summation of similarities over case and target objects.

Case retrieval should be based on the objective function, which is a weighted summation of optimality criteria, equality and inequality constraints and preferred solutions. Equality and inequality constraints are represented by the similarity function. Preferences solutions are already filtered by matching. During this design phase, however, a retrieved case is not yet adapted for current target specifications. Similarity indicates any deviation or required adaptation for functional requirements. Retrieved optimality should also be corrected for any required adaptations. This, however, is not yet possible, and the retrieved optimality criteria cannot be reused as a valid selection criterion.

Minimisation of incremental adaptation is preferred. This approach assumes that cases have already been optimised for their intended functionality and further optimisation can be performed locally in following design phases.

The most similar case is retrieved and instantiated as the initial conceptual design. This selected case will partially match the target for two reasons:

1. It is very unlikely that a case exists which satisfies all current specifications. A case matches partially and its similarity measure is reduced.

2. It is very likely that some of the target objects have not been defined for the retrieved case, which receives a similarity = 0.

## 4.2 Incremental adaptation

Adaptation for unsatisfied targets of the initial concept is possible by substitution or instantiation of subsolutions from other cases. Transformation of subsolutions to integrate their behaviour into the conceptual solution is performed during modification. The

similarity measure for the initially retrieved concept is differentiated for each target object. Target objects with a reduced similarity measure identify remaining targets, which could not be solved by or have not been defined for the initial case. The remaining targets cannot be reformulated as subproblems for particular case parts and subsolutions.

Retrieval of subsolutions from other cases for adaptation of an initial concept is approached differently. The objective for incremental retrieval is to search the case-base again for implicit subsolutions for remaining targets.

The new target for retrieval of subsolutions can be formulated on following considerations:

- Minimal adaptation to the conceptual design reduces potential conflicts in interactive behaviour with the initial case solution. Minor adaptations can be retrieved from cases with almost similar prototypical designs that satisfy a target.
- Case functionality and behaviour only provides positive evidence for the case solution. Similar prototypical solutions can be assumed to be categorised similarly for their behaviour and functionality, even when this has not been defined in the case. If similar designs have been applied for different functional requirements, their cases can only be matched by their solution features.

Searching the case-base for satisfied targets will match a similar set of cases as for the initial target. The case-base can be searched again, for only the unsatisfied and undefined targets of the initial concept. The new target is formulated by the set of remaining target objects and the conceptual solution. The source is still bounded by prototype solution classes to avoid retrieval of non-adaptable subsolutions. Cases have to be found that satisfy most of the remaining targets and are most similar to the conceptual design.

Even the best matching case has a slightly different solution than the new solution for the conceptual design. These differences indicate a sufficient adaptation for the new solution to satisfy also a remaining target. Functionality that has not been specified is ignored as the possible cause for these differences. The differences can be superimposed onto the initial concept by substitution of new attribute values or instantiation of additional component objects. Composition of subsolutions is well defined by the data structure for components.

When the best matching case still cannot solve all target objects, the process can be continued for the remaining target objects. This process is halted when no significantly matching cases can be found for the remaining target objects. Further adaptation has to be performed by modification and optimisation.

Incremental retrieval can only provide weak positive evidence for suggesting adaptations to specific functional requirements. Functionality for previously satisfied requirements may not be defined for new cases. Evidence for possible conflicts cannot be provided consistently from the case-base and is, therefore, not considered. Retrieved subsolutions will not suggest accurate adaptations. Superposition of subsolutions is to be safeguarded by restricting adaptation to conservative changes in attribute values. Non-conservative adaptations are ignored. For example, the number of plies in a layer is not reduced, and a change in fibre orientation can be adapted by instantiation of a new layer. The advantage of this approach is that initial adaptations can be obtained without expensive computations or modifications, and that adaptations can be suggested that have not been defined as modifications.

# 5 HEURISTIC MODIFICATION

When all retrieved subsolutions have been substituted or instantiated, the behaviour is analysed for every component of the adapted conceptual design. Behaviour is analysed and verified in numerical routines for every specification. Numerical analysis results describe behaviour in more detail than predictions for categories in the prototype and concept selection phases. Analysis results express for example the stresses and strains in particular layers for each of the specified panel loads. Based on these accurate results, further local modifications are suggested, evaluated and applied to the adapted concept.

The availability of design knowledge is a major restriction in supporting and automating conceptual design and adaptation. It is not practical to define operations for all possible modifications and strategies for constraint satisfaction and optimisation. Only the most significant modifications are defined by heuristic operations.

Modification operations are defined only for instantiations of new objects, and transformations of non-numerical and discrete attributes. The discrete stepsizes of these design parameters can have a significant effect on feasibility and optimality. More refined repairs and improvements are obtained for continuous and discrete numerical attributes during optimisation.

Modifications are defined as specialised operations for individual attributes or objects. The specialised operations are triggered by significant discrepancies between analysed behaviour and required functionality. For each discrepancy, several attributes can be modified or new objects instantiated. Each modification can also have a significant effect on behaviour of other components and for other specifications. Heuristic operations cannot predict these behavioural side-effects. EADOCS has no adaptation or modification strategies.

A strategy for modification is developed incrementally by combination of specialist modifications. For the most significant discrepancies in analysed behaviour and specifications, multiple modifications are suggested. Each suggested modification represents a minimum or maximum modification. This set of modifications includes multiple suggestions for a single attribute or object, from which a minimum and maximum modification can be determined. The set of modifications provide an estimation of a range of minimal required repairs and maximal allowable improvements to each of the design parameters.

In addition, the effects on specified optimality criteria can be estimated for each suggested modification. These estimates provide a ranking for required repairs and allowable improvements. The optimal combination of modifications can be selected and applied to current design.

Modifications are also based on behaviour predictions and should be analysed for accurate verification. This modification process is repeated, until the conceptual design is feasible and no further improvements can be suggested. The modification phase is described in more detail in [3] and [4]. The modified design is finally optimised numerically by the Complex method [2]. Numerical routines optimise the conceptual design more accurately, include continuous numerical attributes and optimise all numerical variables simultaneously. Accuracy of the Complex method is sufficient for conceptual design.

## 6 COMPARISON TO OTHER SYSTEMS IN THIS WORKSHOP

In this volume other approaches to adaptation are described for case-based reasoning systems such as Déjà Vu [6], PARIS [8], ReSyn/CBR [10], and [9]. The application in EADOCS has two significant differences with the other applications:

1. The domain model represented for decomposition and adaptation is weak and is not directly applied for decomposition and adaptation during case retrieval and reuse.
2. The objective in EADOCS is to design optimal solutions, for which two design phases are necessary in addition to adaptation for constraint satisfaction; prototype selection and numerical optimisation.

Two abstraction levels are defined, similar to Déjà Vu and PARIS. In EADOCS the same data structure is used at both levels, although the relations and operations are declared differently.

Subproblems cannot be solved independently for current application, section 2.2.3. Dynamic decomposition of specifications into tractable subproblems is not possible, as for example in Déjà Vu. Parts of different cases can be reused, but not in a structured manner as in Déjà Vu or PARIS.

The objective of adaptation guided retrieval approach in Déjà Vu is implicit in EADOCS' process model. There are only a few heuristic adaptations and modifications applicable after retrieval. Impossible adaptations are identified by prototype selection before retrieval. All other "adaptations" can be performed by substitution, instantiation, modification or optimisation. Computational costs strongly increase with each level of adaptation. Minimisation of unnecessary numerical optimisations is one of the motivations for the conceptual design process model in EADOCS [3].

## 7 FINAL REMARKS AND CONCLUSIONS

Two characteristics of the application domain require a specific approach to case-based reasoning and adaptation; strong interaction of components on design functionality, and the objective to design optimal solutions. Non-numerical and discrete design parameters require a multi-level approach to optimisation at the abstraction level of prototype designs and the detailed level of conceptual designs. EADOCS implements these levels in a three step approach of prototype selection, concept selection and concept modification. Case-based reasoning is applied for concept selection.

Two specific problems in case-based retrieval and adaptation for current application have been identified:

- Specifications cannot be decomposed into tractable subproblems for components.
- Domain knowledge is not available to predict behaviour and suggest adaptations without detailed numerical analysis for a complete design.

An incremental approach has been presented for incremental retrieval and adaptation. The case-based retrieval is applied to quantify a qualitative design. Targets are defined differently for the retrieval of cases and for incremental adaptation with case parts. Case reuse is simplified to retrieve maximum information for the quantification at minimum costs. Specialised modification operations require accurate prediction of behaviour, which can only be provided by numerical analysis of a conceptual design. Conflicts resulting from integration of case parts into a conceptual solution are solved by modification and optimisation in the revision phase.

EADOCS is still under development and in its prototype phase. The general framework for controlling the design steps, and the modification is implemented in Smart Elements, a rule-based object-oriented environment. A new approach for prototype selection is currently developed in Smart Elements, to replace the initial module in C. The case-based reasoning system is implemented as a separate module in C. The network for the case memory from [7] has been further developed for handling hierarchically structured cases, and matching is still restricted to global similarity within the network. The analysis and discrete optimisation routines of Sapano [2] are available for the extension of conceptual design to preliminary design and optimisation.

## 8 REFERENCES


**[1]** A. Voβ, Exploting previous solutions - made easy, ftp://ftp.gmd.de//GMD/ai-research/Publications/Fabel/Prev-sol-voss.ps.gz

[2] P.G. Van Bladel, *Design of Fibre Reinforced Composite Panels for Aerospace Applications*, PhD thesis, Faculty of Aerospace Engineering, Delft University of Technology, 1995.

[3] B.D. Netten, R.A. Vingerhoeds, H. Koppelaar, L. Boullart, Expert Assisted Discrete Optimisation of Composite Structures. In: *European Simulation Symposium 1993*, A. Verbraeck and E.J.H. Kerckhoffs (eds.), Delft, October 25-28,143-148, 1993.

[4] B.D. Netten, R.A. Vingerhoeds, H. Koppelaar, Expert Assisted Conceptual Design: An Application to Fibre Reinforced Composite Panels, *Design Research in The Netherlands*, Report 37 (eds.) R.M. Oxman, M.F.Th. Bax, H.H. Achten, Faculty of Architecture, Planning, and Building Science, Eindhoven University of Technology, Eindhoven, 125-139, 1995.

[5] B.D. Netten, Knowledge Based Conceptual Design: An Application To Fibre Reinforced Composite Panels. PhD thesis, Faculty of Technical Mathematics and Informatics, Delft University of Technology, (to be submitted 1996).

[6] B. Smyth and M.T. Keane, Experiments on Adaptation-Guided Retrieval, *Case-Based Reasoning, Research and Development*, M. Veloso and A. Aamodt (eds.), Lecture Notes in Artificial Intelligence 1010, Springer Verlag, Berlin, 313-324, 1995. Also in ECAI96, Workshop Adaptation in Case-Based Reasoning, this Volume.

[7] B.D. Netten and R.A. Vingerhoeds, Large-Scale Fault Diagnosis for On-Board Train Systems, *Case-Based Reasoning, Research and Development*, M. Veloso and A. Aamodt (eds.), Lecture Notes in Artificial Intelligence 1010, Springer Verlag, Berlin,.76-76, 1995.

[8] R. Bergmann, W. Wilke, PARIS: Flexible Plan Adaptation by Abstraction and Refinement, ECAI96, Workshop Adaptation in Case-Based Reasoning, this Volume.

[9] B. Fuchs, A. Mille, B. Chiron, Using Explanations to Guide Adaptation, ECAI96, Workshop Adaptation in Case-Based Reasoning, this Volume.

[10] J. Lieber and A. Napoli, Adaptation of Synthesis Plans in Organic Chemistry, ECAI96, Workshop Adaptation in Case-Based Reasoning, this Volume.