4th WDK Workshop on Product Structuring
October 22-23, 1998
Delft University of Technology
Delft, The Netherlands

Supporting the Design of Product Families through Constraint-Based Reasoning

Barry O'Sullivan
Department of Computer Science,
National University of Ireland, Cork, Ireland.
b.osullivan@cs.ucc.ie

Abstract

A product family can be regarded as a collection of products which are similar – similarity being defined from a number of perspectives. For example, a collection of products may be regarded as a product family if they provide the same overall function, they have similar properties or they are built from the same parts. The objective of this paper is to demonstrate that constraint-based reasoning offers a promising basis for the development of support tools for designers concerned with the design of product families. A constraint-based design environment is currently being developed which assists a designer in designing alternative product structures from an initial statement of required functional and physical properties. Each product structure can be regarded as an intentional specification of a product family - each member of which provides the same functionality and are physically similar in structure. In this paper it is demonstrated how a constraint-based model of a product family can handle the complexity of product family design. In addition, this paper discusses how a a constraint-based model of a product family is readily extensible and facilitates the specification of new product families.

1 Introduction

This paper presents an approach to supporting the design of product families. This approach is based on a combination of constraint-based reasoning and design science. A product family can be regarded as a collection of products which are similar – similarity being defined from some perspective. For example, a collection of products may be regarded as a product family if they provide the same function, they have similar properties or they are built from the same parts.

An approach to supporting the conceptual phase of engineering design has been previously reported [16, 18]; this approach is based on a combination of constraint processing [12] and on a generalisation of function-means trees [5]. The approach assists a designer in developing a set of schemes for a product from an initial statement of required functional and physical properties. The designer is assisted in developing alternative candidate function decompositions to provide the required functionality.

The designer is also assisted in defining alternative product structures based on these function decompositions. In effect the designer is supported in designing a collection of products, each member of which satisfies the same product specification. Furthermore, each product is either functionally similar, structurally similar or have similar properties to other products which have been developed. This paper discusses how this work can be applied to the design of product families.

Constraint processing is a relatively new area of interest in the field of Artificial Intelligence. It is an area which is involved with developing techniques for solving the *Constraint Satisfaction Problem* (CSP) [12, 14]. Informally, a CSP is a problem described by a set of variables, each of which has a domain of legal values, and a set of constraints which restrict the simultaneous assignment of values to variables. A constraint can be regarded as some relationship which must hold between some subset of the variables in the problem. When each variable has been assigned a legal value from its domain, without violating any of the constraints, the CSP is said to be solved. Constraint processing techniques have been applied to many aspects of design, such as variational design [6], model-based diagnostics [2], creative and innovative design [15, 24], configuration [13, 20], Design For X [7], costing of designs [10], conceptual design [18, 19, 22] design of mechanical systems [23], supporting design re-use [8], conflict resolution and management in design [1, 9, 17], and supporting concurrent engineering [3, 4]. Further reviews of the application of constraints to design are available [11, 21].

The objective of this paper is to demonstrate how support for the design of product families can be integrated into an existing constraint-based approach to supporting conceptual design. The paper demonstrates that a constraint-based approach to modelling the options available within a product family is useful. Section 2 discusses the nature of product families and the motivation that exists in today's market for their existence. In Section 3, a simple example is used to demonstrate how constraints can be used to support the design and maintanence of product families. In Section 4 a number of concluding remarks are made.

2 The Nature of Product Families

The importance of product families has grown over the past number of years due to an increasing demand for highly diversified product ranges. Customers are placing apparently conflicting demands on manufacturers of products. These demands typically involve increased customisation, reduced order sizes and shorter delivery lead-times. In a response to these pressures, many companies regard competence in the design of product families as a key competitive success factor.

There are many definitions of what constitutes a product family. In this paper a product family is regarded as a collection of products which are similar in some predefined way. For example, a collection of products may be regarded as a product family if they provide the similar functionality, they have similar properties or they are built from the same parts.

Two approaches to the design of product families have been identified [25]. The first of these approaches relates to the *"market oriented design of the product family"*. In this approach a company looks to the marketplace in order to segment the market. Once the market place has been segmented a company can focus on the development of a product offering to particular segments of the market. The second approach relates to the "*constructional realisation of the product family*". This approach requires a company to focus internally on existing technologies used in the design of product programs. The objective is to maximise the degree of standardisation and modularity amongst existing technologies in order to maximise the variety of products which can be offered to customers. This approach is of particular interest when adopting a formal approach to supporting the design of product families.

The perspective on product family design adopted here is that the design objective is to provide customers with as large a diversity of related products as possible. The customer's requirements may, or may not, be known in detail at "design-time". At design-time all that is known is an abstract specification for a product. However, the product structures which are developed should allow for considerable customisation on the behalf of the customer. While the customer should be offered considerable freedom to specify the particular product they require, there will naturally exist some restrictions on what is technically or economically feasible. In Section 3 an example scenario will be used to describe the role of constraint-based reasoning in product family design.

3 Case-Study: Vehicle Design

A case-study will be used to demonstrate how constraint processing can be used to support the design of product families. The case-study used in this paper is based around vehicle design. This case-study has been inspired by a well known design problem presented in the constraint processing literature [13]. For the purposes of this paper the design task is to develop families of products which can be used to transport humans. An approach for supporting product structuring using constraints will be described briefly in Section 3.1. Section 3.2 will discuss one constraint-based product structure in detail and show how it can be regarded as an intentional specification of a product family. Section 3.3 will describe how a new family can be created from existing product family specifications.

3.1 Constraint-based product structuring

In Figure 1 an existing approach to generating constraint-based product models from an initial statement of required functional and physical properties is illustrated. This approach has been reported previously [16, 18]. The details of the approach will be discussed briefly here, but a detailed example is beyond the scope of this paper. Readers are encouraged to refer to the existing published literature.

The first step in the approach is to define the set of requirements for the product being developed. This set of requirements is referred to as the *design specification*. The design specification will contain details of both the functional and physical properties which are required.

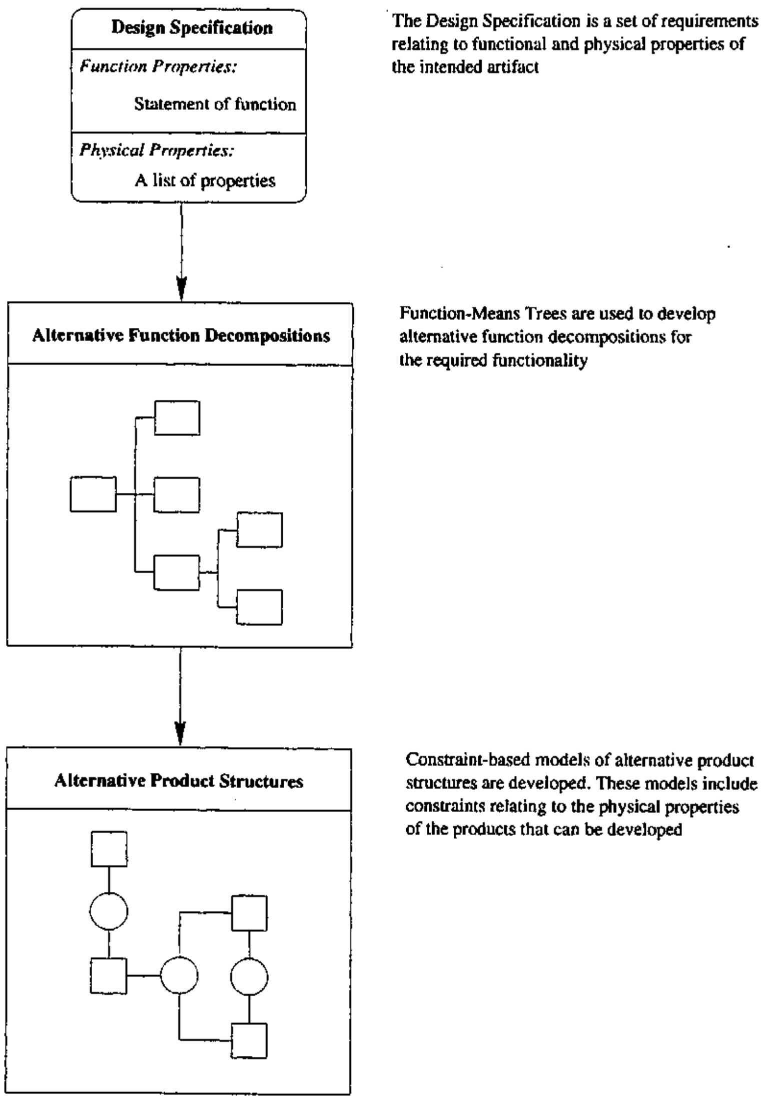
Design Specification
Function Properties:
Statement of function
Physical Properties:
A list of properties
The Design Specification is a set of requirements relating to functional and physical properties of the intended artifact
Alternative Function Decompositions
Function-Means Trees are used to develop alternative function decompositions for the required functionality
Alternative Product Structures
Constraint-based models of alternative product structures are developed. These models include constraints relating to the physical properties of the products that can be developed

Figure 1: Constraint-based product structuring

The functional properties are used as a starting point for developing alternative function decompositions which the designer explores to further learn about possible ways of solving the design problem. In the approach which has been developed, the function decomposition step may be assisted using a generalised function-means tree approach [5].

The alternative function decompositions can be used as a basis for developing a number of alternative product structures. A product structure is a collection of structured descriptions of parts or modules which have particular relationships between them and are subject to a number of constraints. Each product structure can be regarded as an intentional specification of a family of products, each member of which provide the same functionality and are physically similar in structure.

Figure 2 presents a list of alternative product structures which could have been de-

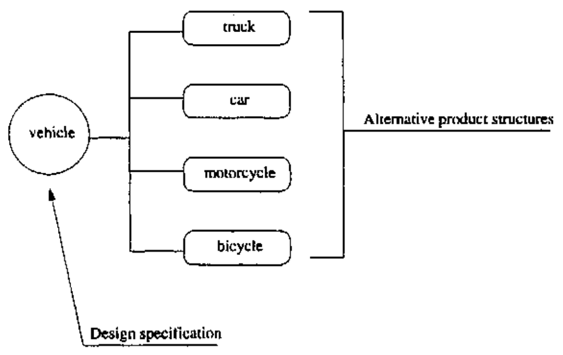

Figure 2: Possible solutions to the vehicle design problem

veloped by a designer trying to solve the vehicle design case-study. These alternatives are a *car*, a *truck*, a *motorcycle* or a *bicycle*. The generic product structure of each of these alternatives can be readily modelled in a constraint programming language.

In the following sections aspects of the constraint model of the *car* product structure will be explored in further detail. This discussion will highlight how customisation can be supported by considering a constraint-based representation of a product structure as an intentional specification of a product family based on a particular structure. By interacting with instances of these product structures, it is not difficult to develop products which can be regarded as being members of the same product family.

3.2 Customisation: Exploring the product family

Figure 3 illustrates the structure of *car* as having an *engine*, a *frame* and a *package*[1]. There are a number of additional features which may be present in a *car* - a *sunroof* or an *air conditioner*. Figure 3 also illustrates the options that are available for each element of the *car*. For example, the *frame* of a *car* may be *convertible*, *saloon* or *hatchback*.

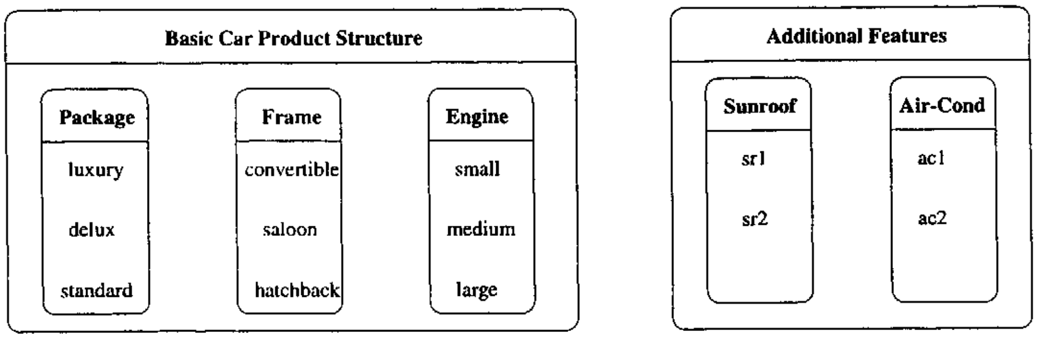

Figure 3: Product structure of the *car*

In Figure 4 a constraint-based model of the *car* product structure is presented. This

[1]This example is based on an example from the constraints literature [13].

model is described in the constraint programming language Galileo [4]. Galileo is a constraint programming language based on first-order logic. Lines 9–10 of Figure 4 define a *car* as a *vehicle* whose *type* is *a_car*. Lines 11–13 state that all cars have the following parameters: *package* of type *package_type*, *frame* of type *frame_type* and *engine* of type *engine_type*. Lines 15–22 define what options are available for the *package*, *frame* and *engine* of a *car*.

```
1  module car.
2
3  domain vehicle
4      =::= ( type : vehicle_type ).
5
6  domain vehicle_type
7      =::= { a_car, a_truck, a_motorcycle, a_bicycle }.
8
9  domain car
10     =::= { C: vehicle(C) and C.type = a_car and
11                exists( C.package : package_type ) and
12                exists( C.frame   : frame_type ) and
13                exists( C.engine  : engine_type ) }.
14
15 domain package_type
16     =::= { luxury, delux, standard }.
17
18 domain frame_type
19     =::= { convertible, saloon, hatchback }.
20
21 domain engine_type
22     =::= { small, medium, large }.
```

Figure 4: A constraint-based model of the *car*

The constraint-based model of the *car* can be regarded as an intentional specification of a product family based around the concept of a *car*. Considering only the model presented in Figure 4 a customer is free to select any car from a family of 27 – since there is a total of 27 cars specified by this program. Thus, it can be seen that adopting a constraint-based approach to describing product families is a very convenient way of handling the complexity of describing the members of a product family.

```
1  domain sunroof_type
2      =::= { sr1, sr2 }.
3
4  domain air_conditioner_type
5      =::= { ac1, ac2 }.
6
7  all car(C): C.package = delux or C.package = luxury implies
8      exists ( C.sunroof: sunroof_type ).
9
10 all car(C): C.package = luxury implies
11     exists ( C.air_conditioner: air_conditioner_type ) and
12     C.air_conditioner <> ac1.
13
14 all car(C): C.package = standard implies
15     C.frame <> convertible.
```

Figure 5: Handling constraints on options in the car example

There may be many situations in which there will be constraints on what constitutes a valid combination of options that a customer is permitted to select. A number of additional options which may exist in a *car* are included in Figure 3. These options include a *sunroof* and an *air_conditioner*. There may also be particular conditions under which a customer has the opportunity to select from additional options when selecting a particular product from a product family. This scenario is modelled in Figure 5. Lines 7–8 state that if a *car* is either *delux* or *luxury* then the *car* has a *sunroof*. Lines 10-12 states that *luxury* cars have an *air_conditioner* of type *ac1*. Lines 14-15 state that *standard* cars cannot be *convertible*.

One of the advantages of using a constraint-based approach to supporting the description and customisation of product families is that any restrictions that need to be specified can easily be represented as constraints. In many cases there may be particular DFX guidelines which must be incorporated into a model so that a customer does not request a product which is incompatible with some phase of the product life-cycle.

3.3 Specifying new product families

It has already been stated that a product family can be regarded as a collection of similar products. Using a constraint-based approach to supporting the design and maintenance of product families, a company can specify new measures of similarity which can be used to create new product families.

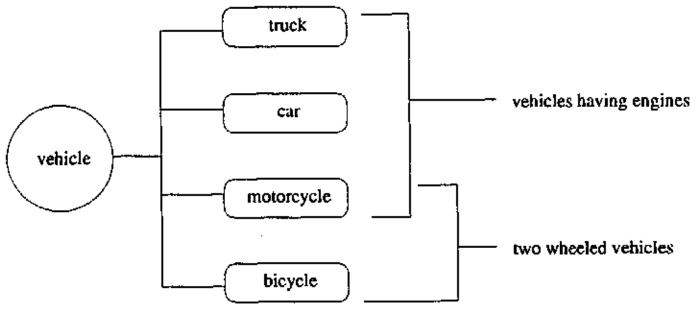

Figure 6: Identifying new product families

Figure 6 depicts a situation which may arise if a company attempts to consolidate particular product families into new product families. For example, a company who manufactures vehicles may not be satisfied with a product family profile based on trucks, cars, motorcycles and bicycles. Instead a company may wish to develop product families around the concepts of vehicles having engines or vehicles having two wheels while maintaining the existing product family profile. This is an easy task if product families are described in a constraint-based manner. Figure 7 illustrates just one way of introducing two new product family categories – *vehicles_with_engines* and *two_wheeled_vehicles*.

In Figure 7, lines 1–2 define the family of *vehicles_with_engines* as those products based on the notions of a *car*, a *truck* or a *motorcycle*. Lines 4–6 define a family of *two_wheeled_vehicles* as the set of vehicles which have two wheels.

In a similar way, a company can create many different product families from exist-

```
domain vehicles_with_engines
    =::= { E: car(E) or truck(E) or motorcycle(E) }.

domain two_wheeled_vehicles
    =::= { T: vehicle(V) and
               number_of_wheels_on( V ) = 2 }.
```

Figure 7: Specifying new product families

ing ones. Since a product family can be regarded as a collection of similar products, all that is required is that the company describe the appropriate measure of similarity. There is an implicit similarity due to product structures being developed to fulfil the same overall functionality. However, similarity can be defined based on parts/modules/components, on interfaces between parts/modules/components, routes of manufacture *etc.*

4 Concluding Remarks

In this paper an approach for supporting the design of product families was presented. The approach is based on an existing constraint-based approach to supporting conceptual design and product structuring. The approach to supporting conceptual design and product structuring generates constraint-based models of alternative product structures. Each of these product structures can be regarded as an intentional specification of a product family.

Customers are free to specify any product in a particular product family by interacting with its constraint-based product structure representation. A company can modify its product family offering by defining new product families by specifying new similarity relations.

Acknowledgements

The work reported here was funded by the European Commission under ESPRIT project number 20501, with acronym CEDAS. CEDAS is an acronym for "Concurrent Engineering Design Adviser System". The author is also a member of the IiMB Subgroup for Design Coordination. IiMB is also funded under ESPRIT working group 21108. IiMB is an acronym for "Integration in Manufacturing and Beyond".

References

[1] D. Bahler, C. Dupont, and J. Bowen. Mixed quantitative/qualitative method for evaluating compromise solutions to conflict in collaborative design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 9:325–336, 1995.

[2] Anton Biasizzo and Franc Novak. A methodology for model-based diagnosis of analog circuits. Technical Report CSD-TR-95-11, Jožef Stephan Institute, Slovenia, 1995.

[3] James Bowen. Using dependency records to generate design coordination advice in a constraint-based approach to Concurrent Engineering. *Computers in Industry*, 33:191–199, 1997.

[4] James Bowen and Dennis Bahler. Frames, quantification, perspectives and negotiation in constraint networks in life-cycle engineering. *International Journal for Artificial Intelligence in Engineering*, 7:199–226, 1992.

[5] Jacob Buur and Mogens Myrup Andreasen. Design models in mechatronic product development. *Design Studies*, 10(3):155–162, July 1989.

[6] Jack C. H. Chung. Constraint-based variational design. In Walter Hower, Djamila Haroud, and Zsófia Ruttkay, editors, *Proceedings of the AID'94 workshop W9 on Constraint Processing in Computer-Aided Design*, pages 44–54, 1994.

[7] Marc van Dongen, Barry O'Sullivan, James Bowen, Alex Ferguson, and Mike Baggaley. Using constraint programming to simplify the task of specifying DFX guidelines. In Kulwant S. Pawar, editor, *Proceedings of the 4th International Conference on Concurrent Enterprising*, pages 129–138, University of Nottingham, October 1997.

[8] Pat Fothergill, Jorg Forster, Jose Angel Lakunza, Fco Plaza, and Ines Arana. DEKLARE: A methodological approach to re-design. In *Proceedings of Conference in Integration in Manufacturing*, Vienna, Austria, September 1995.

[9] Djamila Haroud, Sylvie Boulanger, Esther Gelle, and Ian Smith. Management of conflict for preliminary engineering design tasks. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 9:313–323, 1995.

[10] Carolyn C. Hayes and Harold C. Sun. Using a manufacturing constraint network to identify cost-critical areas of design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 9:73–87, 1995.

[11] Walter Hower and Winfried H. Graf. A bibliographical survey of constraint-based approaches to cad, graphics, layout, visualization, and related topics. *Knowledge-Based Systems*, 9(7):449–464, December 1996.

[12] Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[13] Sanjay Mittal and Brian Falkenhainer. Dynamic constraint satisfaction problems. In *AAAI 90, Eighth National Conference on Artificial Intelligence*, volume 1, pages 25–32, Boston, MA, July–August 1990. AAAI Press, Menlo Park, CA, U.S.A.

[14] Ugo Montanari and Francesca Rossi. Fundamental properties of networks of constraints: A new formulation. In L. Kanal and V. Kumar, editors, *Search in Artificial Intelligence*, pages 426–449. Springer, 1988.

[15] D. Navinchandra. Innovative design systems, where are we and where do we go from here part 1: Design by association. *Knowlwdge Engineering Review*, 7(3):183–213, 1992.

[16] Barry O'Sullivan. A constraint-based support tool for early-stage design within a Concurrent Engineering environment. In Kulwant S. Pawar, editor, *Proceedings of the 4th International Conference on Concurrent Enterprising*, pages 119–28, University of Nottingham, October 1997.

[17] Barry O'Sullivan. Conflict management and negotiation for Concurrent Engineering using Pareto optimality. In R. Mackay N. Mårtensson and S. Björgvinsson, editors, *Changing the ways we work - Shaping the ICT-solutions for the next century*, Advances in Design and Manufacturing, pages 359–368, Amsterdam, October 1998. IOS Press. Proceedings of the Conference on Integration in Manufacturing, Göteborg, Sweden.

[18] Barry O'Sullivan and James Bowen. A constraint-based approach to supporting conceptual design. In John Gero and Fay Sudweeks, editors, *Artificial Intelligence in Design '98*, pages 291–308, The Netherlands, July 1998. Kluwer Academic Publishers.

[19] S. Y. Reddy, K. W. Fertig, and D. E. Smith. Constraint management methodology for conceptual design tradeoff studies. In *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, August 1996. Irvine, California.

[20] Daniel Sabin and Eugene C. Freuder. Configuration as composite constraint satisfaction. In *AAAI-96 Fall Symposium on Configuration*, pages 28–36, 1996. also in: Proceedings, Artificial Intelligence and Manufacturing Research Planning Workshop 1996.

[21] Mark Sapossnek. Research on constraint-based design systems. In *Proceedings of the Fourth International Conference on Applications of AI in Engineering*, July 1989. also in Artifical Intelligence in Design 1989.

[22] David Serrano. *Constraint Management in Conceptual Design*. PhD thesis, Massachusetts Institute of Technology, October 1987.

[23] Leon Sterling. Of using constraint logic programming for design of mechanical parts. In Leon Sterling, editor, *Intelligent Systems*, chapter 6, pages 101–109. Plenum Press, New York, 1993.

[24] K. Sun and B. Faltings. Supporting creative mechanical design. In *Artificial Intelligence in Design*, pages 39–56, 1994. Kluwer Academic Press, Netherlands.

[25] Marcel Tichem. *A Design Coordination Approach to Design for X*. PhD thesis, Technische Universiteit Delft, September 1997.