CHAPTER 14

Compared application of two knowledge modelisation methodologies on a car engine cylinder head design problem

Future expectations in the use of a generic design application development tool

A. Saucier[1,2], C. Vargas[1,2], P. Court[1], P. Albert[3], P.A. Yvars[1]
[1]Peugeot SA, DTII.PMT.ICM, 62 Bd. Victor-Hugo, 92208 Neuilly/Seine - France
[2]Laboratoire Universitaire de Recherche en Production Automatisée, ENS Cachan, 61 Av. du Président Wilson 94235 CACHAN cedex - France
[3]Ilog, 2 Avenue Galliéni, B.P. 85 - 94253 Gentilly Cedex - France

The growing interest in the specification and development of user friendly and easy to maintain design tools has urged engineers to develop and use design knowledge acquisition and modelisation methodologies. In this chapter, we present a comparative study of the application of two knowledge modelisation methodologies in the field of mechanical design. The modelisation capabilities offered by these methodologies are presented, illustrated, and discussed referring to their application to the development of a car engine cylinder head design and optimisation system at PSA. This study takes place in a global project for the creation of a generic development tool for design applications. This project includes three phases: the development and use of a design knowledge modelisation methodology, the specification and development of a Design Description Language, and the specification, development and use of a system allowing the execution of the modelised problem, linked with traditional CAD/CAM systems.

R. Soenen et al. (eds.), *Advanced CAD/CAM Systems*

14.1. INTRODUCTION

This chapter presents a comparative study of the application of two knowledge modelisation methodologies to a mechanical design problem: the design of a car engine cylinder head.

The cylinder head problem presents a good amount of the difficulties a designer can encounter, including complex product structure, non monotonous design process, sub problems linked to each other and constrained. It should therefore be a good test case for the methodologies.

KADS (Knowledge Analysis and Documentation System) has been developed in the KADS 1 and KADS 2 projects.

DDAM (Deklare Design Analysis Methodology) has been developed within the Deklare project to specifically handle design problems (and particularly mechanical design problems).

The partners of the Deklare project are the Computer Science Department of the University of Aberdeen (UK), the ILOG (F), IKERLAN (S), COPRECI (S) and PSA (F) companies.

The quality of the models obtained with the method (completeness, precision, etc.) will allow us to evaluate the methodologies.

We will also consider the reusability of these models as data structures in the development of a knowledge based system able to solve the cylinder head design problem.

14.2. THE CYLINDER HEAD DESIGN PROBLEM

14.2.1. How does a cylinder head work?

The cylinder head is one of the most important parts in the engine. It participates in several functions, including induction, compression, combustion of the air/gasoline mix and ejection of exhaust gas.

These functions are supported by several parts of the cylinder head:

1 the air/gasoline mix comes from the intake manifold into the induction duct,
2 the mix goes into the combustion chamber through holes opened by the valve mechanism,
3 the closing of the valve allows the compression of the mix,
4 the explosion is obtained by the spark produced by the spark plug,
5 once the exhaust valves are opened, the exhaust gas are rejected into the exhaust ducts,
6 the cooling and lubrication of the cylinder head are made by water and oil passages in the cylinder head body.

14.2.2. Description of the problem

The problem can be divided into two main sub-problems:

1 the design of the valve opening and closing mechanism (valve lifters, springs,..),
2 the design of the cylinder head body, including the induction and exhaust ducts, the combustion chamber, the water and oil passages.

The design of the body and the mechanism can be divided into several tasks. These tasks are strongly linked to each other and constrained.

As a consequence, the sub problems have to be solved simultaneously, that explains why the development of a system is needed.

Two kinds of improvements are expected:
1 the shortening of the design cycle time,
2 the improvement of the design process, and of the quality of the product.

The cylinder head used as a test case for the application is a 16 valve, cross flow, with direct valve gear cylinder head.

14.3. THE DDAM AND COMMONKADS® METHODOLOGIES

14.3.1. Knowledge acquisition and modelisation methodologies

The survey of available acquisition and modelisation methodologies [Dek 93c] has showed that two types could be distinguished:
1 the general purpose methodologies,
2 the dedicated methodologies.

Among the available general purpose methodologies, (KADS, KOD, MERISE,...), we have selected CommonKADS. Indeed KADS offers an open representation formalism, works on a computerised tool, and results of eight years of European project work.

Among the mechanical design dedicated methodologies, [Pah 88], [Fin 89a], [Fin 89b], we have applied DDAM [Dek 93a]. This method has been developed within the Deklare project.

14.3.2. DDAM

14.3.2.1. Origin of the methodology

DDAM (Deklare Design Analysis Methodology) [Dek 93a] is a knowledge acquisition and modelisation methodology dedicated to the mechanical design field. The formalism established for the knowledge representation has to be sufficiently open to integrate the different levels of design problems, from assembly to complex design problems.

The design problem modelisation with DDAM includes two steps: the product

modelisation, and the process modelisation.

The methodology is composed of a user's manual and a set of format sheets to be filled by the user [Dek 93a].

14.3.2.2. The product modelisation

During his speech, the designer uses several complementary points of view on the problem (physical , functional, geometrical, etc...). To have a model as complete as possible, it is necessary to represent all these points of view.

The product is thus modelised by several models (physical, functional, geometrical). The constraints that can be applied on the elements of these models are also to be represented.

The physical model:

It is composed of "articles", "assemblies", "parts" and "design features".

An article is a set of assemblies or parts. An assembly is a set of parts. A part is composed of one or several design features. Several parts can have some design features in common.

The physical model is obtained by drawings, technical documentation, and expertise documents.

The functional model:

The functional model is composed of "articles", "thinking blocks", "concepts", "technical solutions" and "design features".

Blocks and concepts represent the functions to be realised by the article (the product). Concepts are elementary functions that cannot be divided. Technical solutions correspond to the realisation of a concept. Design features give a physical representation to technical solutions.

The functional model is established by the knowledge engineer, based on all available documents concerning the design problem. When a new function is identified on a document, the Knowledge engineer has to determine to which level this function belongs (concept, thinking block). For each concept, a list of technical solutions has to be made, and their corresponding design features have to be parameterised.

The link between the physical and functional models is made at the top of the hierarchy (article), and at the bottom (design feature). Otherwise, both models are independent and complementary. Indeed, if the designer works with functions, the corresponding parts are progressively defined by the parameterisation of their design features. On the contrary, when the designer wishes to reuse existing parts, the corresponding functions are automatically instanciated by the corresponding design features.

The geometrical model:

This model links an object with its geometrical representation through an enumeration of geometric basic objects (curves, surfaces, solids) and operations (boolean operations).

The constraints:

The constraints are used to express particular conditions that need to be

verified by the elements of the models. A constraint can link several parameters of several parts. Constraints can be of geometric, numeric, or symbolic nature.

14.3.2.3. Modelisation of the design process

The formalism used for the representation of the design process in DDAM is inspired by the survey of Design Analysis Methodologies [DEK 93c], by the expertise model of the CommonKADS methodology [SHR 92], by the control model of the Archix system [THO 91], and by the task model of the expert system development tool SMECI [ILO 92].

The design process in DDAM is composed of several parts:

1 the elementary tasks, represented as a task tree,
2 the control structures
3 the problem solving strategies.

The task tree is a static representation of the design process and is deducted from the product functional model. The task tree is usually different from the functional model. The functional model is composed of thinking blocks, concepts, technical solutions. It gives the structure of the problem to be solved. The task tree is a simplification of this structure.

We differentiate two types of tasks:

1 the low level corresponds to elementary calculations (leaves of the task tree),
2 the high level handles the management of sub tasks. These high level tasks constitute a control structure and define the strategy to apply in case of failure of a sub task.

Each task is described by the following attributes:

1 goal of the task,
2 set of sub tasks,
3 local strategy (in case of failure of a sub task),
4 set of methods: list of available methods to reach the goal of the task.
5 description of the method: how to reach the goal.

The DDAM methodology allows the modelisation of :

1 the different tasks needed for the design of the product, the control structures and strategies associated.
2 the links between different tasks.

14.3.3. CommonKADS®

14.3.3.1. Origin of the methodology

The first developments related to the CommonKADS methodology began in the

early 80's at the University of Amsterdam. They came out of an analysis of the current methods used to develop expert systems. At that time, an expert system was mainly characterized by the fact that one used Rules in the programming of the application.

Bob Weilinga, from the university of Amsterdam took another promizing way, considering that the expert system approach should be characterized by the type of the problems (diagnosis, design, planning, etc.) rather than by the technics used to implement the system.

The approach proposed by CommonKADS thus relies on the definition of a library of models of problem classes, and on the definition of mappings between the abstract description of the problem, and the application domain model. This approach has been intensively developed and tested within two Esprit projects.

These ideas have led to the birth of the CommonKADS methodology, widely supported by the European community. Its development followed three axis:

Research work within European Esprit projects, to establish and consolidate the foundations of the methodology.

1 Industrial work led by European I.T. companies such as Ilog or Boolesian, or by U.S. companies such as the Bechtel institute.

2 Application work by many industrial companies using KADS to implement decision support applications.

14.3.3.2. Principles of CommonKADS

The main principle behind CommonKADS is genericity. It is based upon the fact that there exists application classes, corresponding to problem classes.

For each problem class that has been sufficiently studied, one can define an abstract model of the problem and of the associated problem solving methods. As this generic description is independent of any application domain, is may be reused for different applications based upon the same problem class.

The objective of the method is thus to provide:

1 the formalism allowing the representation of abstract problem, and their associated solving methods.

2 the formalism needed to specify the entities and relations within a specific application domain.

3 a method allowing the description of the mapping between the terms of the abstract problem model, and the application domain description.

Furthermore, CommonKADS comes with libraries that predefine the solutions of certain problem classes [Alb 92], [Ilo 92].

14.3.3.3. The CommonKADS Models

The CommonKADS model of application is made of three nested level of models.

1 the Domain layer represents the entities of the application domain,

2 the Inference layer represents the operations performed during the reasoning process, and the roles played by the domain entities in these operations,
3 the Task layer defines the precise control structure that links together the operations defined at the inference level.

The links between these three layers are as follow:
1 the task level controls the inference level,
2 the inference level is mapped on the domain level.

14.3.3.4. The domain model

Within CommonKADS, the domain model describes the entities that are part of the application domain of the considered application. The notation used is an extension of the notation introduced by the Object Oriented Method O.M.T. It is used to define the following entities:
1 objects model: the data model is object oriented. It is made of classes linked by an inheritance relation. Each class defines a set of properties characterized by their type.
2 relations model: the objects may be linked together by binary relations. A binary relation defines two roles which are the projection of the relation on each of the linked classes. The composition relation receives a special support. Views can be defined, that group together a set of objects and relations.
3 dynamic model: this model is composed of expressions and of relations between expressions. The expressions may be formulae, predicates or rules. They are used to define the computations or the states associated to the objects.

14.3.3.5. Inference structures

The inference structures are the backbone of CommonKADS. They specify the operations (or inference steps) used to solve the problem, as well as the information used or produced during the reasoning process.

The formalism derives from the data-flow notation, which it extends. An inference structure represents the organisation of a set of inference steps linked by roles. Two types of roles are distinguished :
1 the dynamic roles that represent the information flow (i.e. the input and the output of the inference steps).
2 the static roles that represent the domain dependent information used by the inference steps to realise their functions. A static role may for example be associated to a set of rules.

The operations as well as the roles may be described at different levels of abstraction. An inference step may be "primitive". In this case, it models a terminal operation which is not further described.

It may also be "decomposed". In this case, it represents an abstract operation which is further specified by the inference structures that represent its decomposition. The roles associated to an inference step may also be decomposed. We define below the meaning of the graphical notations.

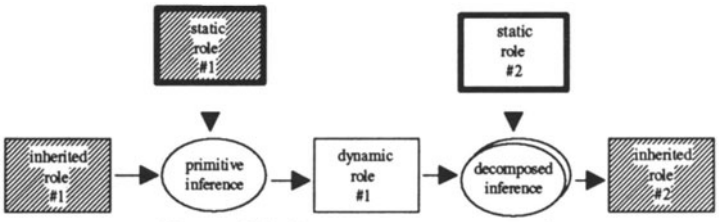

Figure 1. Inference structure example

The figure above contains the set of notations used to define an inference structure. An inference structure represents the organisation of a set of operations: the inference steps represented by ellipsis.

These operations consume and produce data. These information named "dynamic roles" are represented by simple rectangles attached to the ellipsis by links that may be oriented.

If the connection mode between a role and an inference step is not known, the rectangle is attached to the ellipsis by a non-oriented link. In this case it is the control of the reasoning that determines dynamically the nature (input/output) of the role.

The inference steps may need domain dependent knowledge to be implemented. These information, named "static roles", are represented by bold rectangles.

An inference step may also be primitive or decomposed. In this later case, it is represented by a double ellipsis. The decomposition of the inference step will be made of one or many inference structures.

Finally, within an inference structure that represents the decomposition of an inference step, the inherited roles defined at the upper level of the decomposition are represented by hashed rectangles.

The binding of the inference structure, that describes the problem independently of the domain, to the domain model is specified on the roles. For each role, one describes the domain elements that it represents.

14.3.3.6. The task structures

The task level describes the precise algorithmic control applied to the inference structures. As well as the data-flow model they originated from, the inference structures do not bare any control information, which makes them more reusable. They only describe the operations and the associated data-flows. These are the tasks that describe the control.

The task structure represents a tree of goals and methods. A goal corresponds to an inference step, and a method corresponds to a specific control applied to an

inference structure.

14.4. APPLICATION OF THE METHODOLOGIES TO THE DESIGN AND OPTIMISATION OF A CYLINDER HEAD

14.4.1. Results of the modelisation with DDAM

14.4.1.1 The product structure

The structure of the cylinder head is represented by the physical and functional models.

The physical model:

The physical model describes the set of parts of the cylinder head and the associated mechanisms.

The cylinder head model is composed of parts (cylinder head body, valve guide, valve seat, valve seal) and an assembly: the timing mechanism (spring, seat, double cone, valve, valve lifter, spring retainer).

The physical model of the cylinder head is rather simple. Indeed, many functions are supported by the cylinder head body: the gas ducts, the water and oil passages, the guiding of the valve lifters and camshafts, the fastening of guides, intake and exhaust manifolds. The physical model is not precise enough to define the cylinder head body, but it is sufficient to define parts like valve lifters, springs.

The functional model:

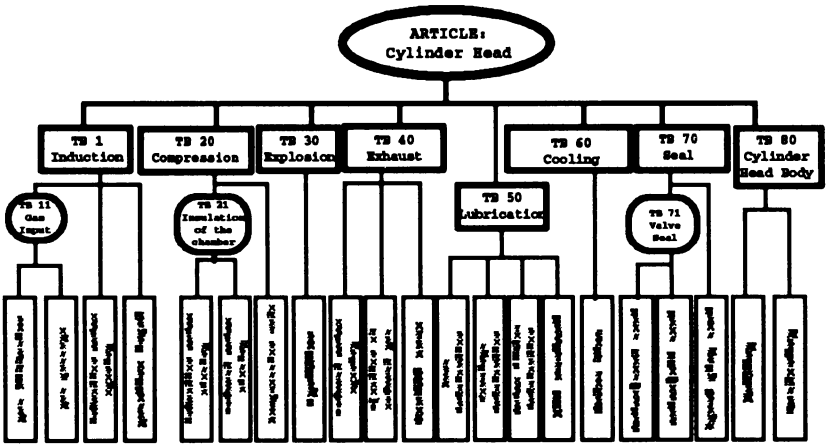

Figure 2. Functional model of the Cylinder Head

The figure 2 presents the functional model of the cylinder head. It contains the thinking blocks and concepts needed for the description of the problem: that is all the functions to be realised.

For instance: the "induction" thinking block is divided into several concepts like "induction valve opening". The technical solution associated to this concept is the induction valve gear.

The corresponding design features determine completely the valve opening mechanism, i.e. the spring, the spring retainer, the double cone, the valve lifter, etc.

In our case, the functional model is much more detailed than the physical one. This decomposition allows a progressive design in terms of functions rather than parts.

14.4.1.2 The design process

The design of a cylinder head is a complex design problem. Indeed the designer has to deal with a good number of sub problems and constraints.

The task tree gives a global view of the static description of the process.

On behalf of this tree, each task is described independently. We can distinguish two sorts of tasks (fig 3):

1 On the one hand, tasks that participate in the definition of the parts of the physical model (calculations of the dimensions and positions of the objects).

2 On the other hand, higher level tasks control their sub tasks and manage potential failures.

Part of the information contained in these higher level tasks refer also to problem solving strategies , and how to apply them depending on the state of the design process.

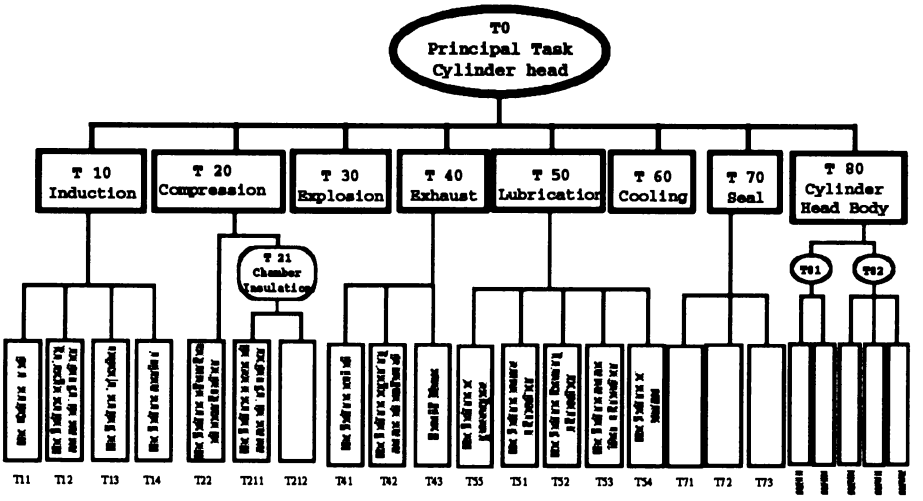

Figure 3. Task tree

Example of part of a low level task:

```
"T 11 Induction duct definition":
        Method 1 (programme):
                // Induction duct definition
                to calculate (diameter, l..) ;
                to define neutral fibre
                method 2 (programme)
                // Induction duct position
                give geometric position of induction duct
```

The task 11 defines the induction duct of the cylinder head.

The method 1 contains a set of calculations for the definition of the characteristics of the duct.(for instance, the diameter, the shape of the neutral fibre, etc.).

The method 2 defines the position of the duct depending on the constraints (for instance: to avoid interferences with other parts) and predefined characteristics (the duct should be as vertical as possible).

The following task 10 presents a high level task

```
"T 10 Induction definition":
Task (goal):        T10  to define cylinder head induction
Sub tasks (sub goals):
        { T11 induction duct; T12 valve opening;
          T13 injector; T14 venturi effect}
Local Strategy (program):
        // What method to choose
        method = designer-choose-the-method,
        or method = method 1
Methods:            { method 1; method 2; method 3;
                              method 4; method 5 }
 method 1 (program):
        // Execute all tasks, deterministic or not function
        execute T11, after T12, after T13, after T14
 method 2 (program):
        // To define only induction duct
        To execute T11
```

The task 10 controls the low level task 11 that corresponds to calculations of dimensions and positions.

14.4.2. Results of the modelisation with CommonKADS

14.4.2.1. Introduction

The CommonKADS model allows the representation of two types of knowledge

1 knowledge concerning the design domain (domain level)
2 knowledge concerning the problem solving mechanism tasks and inferences (task and inference levels)

14.4.2.2. Product structure

The following CommonKADS objects have been used:

1 **concepts**: physical or logical object of the application field.

For instance:

Camshaft Father: valve movement mechanism References Cylinder Head [73] camshaft Introduction duct Father: Cylinder Head body References Cylinder Head [58] induction ducts

2 **Relation**: types of relations between concepts: for instance: "is characterised by"

3 **Expression**: formula used for calculation or any other type of information.

Example:

Compression rate calculation Text : rate = (volume + cubicCapacity) / volume.

4 Hierarchies of the domain

Example:

Cylinder Head Relation : subtype References: Cylinder Head [38] The cylinder head is subdivided into two parts, the cylinder head body and the valve gear mechanism.

A physical model of the cylinder head and the expression of the different relations that exist between the concepts of this model have been established.

14.4.2.3. The design process

The model of the design process can be decomposed in two levels:

1 The task level gives a general view of the different problems to solve (see fig.4).

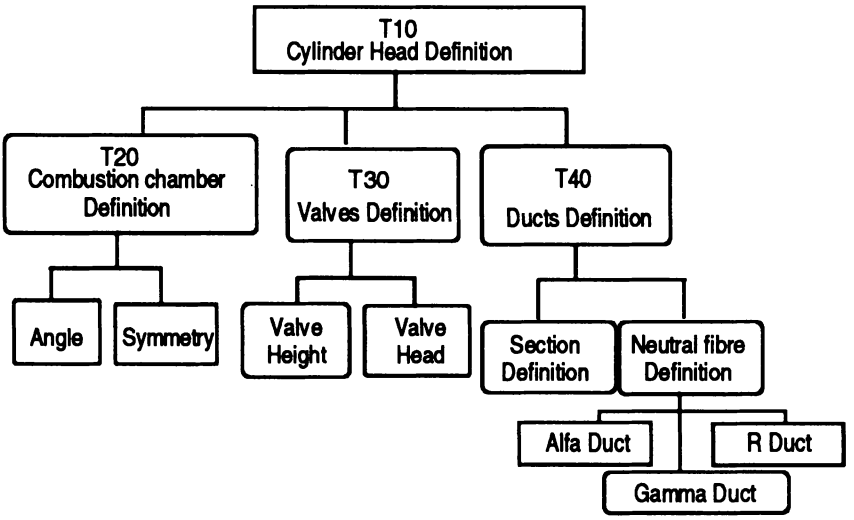

Figure 4. *Task tree*

2 The inference level gives a global view of the reasoning during the design process.

In our case the inferences obtained are not significant abstractions of the problem, as shown in fig. 5.

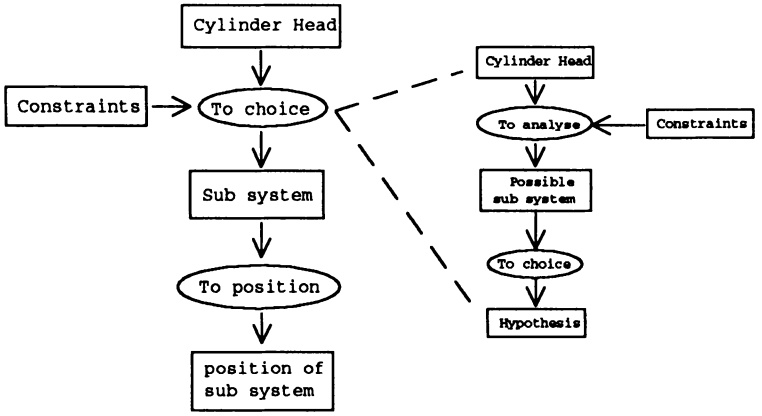

Figure 5. Example of an inference structure

The inference structure can be interesting if the structure of the design process is to be reused in several applications (For instance in another cylinder head design problem i.e. diesel engine or 2 valves / cylinder).

In such a case, it is important to structure the design process to allow the reusability of modules without remodelising the complete problem.

In CommonKADS, some generic inferences already exist but non of them was applicable in our case.

14.4.3. Synthesis of the application of DDAM and KADS

Both methodologies make clear a differentiation between the product and the process. The application has confirmed the interest of this difference.

In DDAM, the product structure description is based on the physical and functional model, these are the ones used in the mechanical design field. These models do not exist in CommonKADS unless they are created.

The functional model obtained with DDAM is the most precise and complete decomposition problem.

This model can be the starting point of the design process description and can result in substantial differences between the process used by the designer and the system.

The "expressions" representation in CommonKADS needs to be specialised for the design field (constraints).

The structure given by CommonKADS for the representation of the design process is more complete than the one offered by DDAM.

Indeed, the inference level allows the modelisation of general reasoning processes to be reused in several applications.

However, in our case, the inference structures obtained are not significantly superior compared to the DDAM modelisation.

The different problems modelised with KADS are very close to the designer's process. Thus they do not allow a new decomposition and structuration of the problem as DDAM does.

Some tasks identified with KADS do not appear in the DDAM process model and vice versa. The design of some parts of the product is dispatched in several tasks in DDAM, whereas it is grouped in one task in CommonKADS.

14.5. FUTURE EXPECTATIONS FOR THE DEVELOPMENT OF DESIGN AID TOOLS

14.5.1. A tool box for the development of new applications

The DDAM is the first part of a bigger project the purpose of which is to offer a complete design aid application development environment.

Indeed the model of the problem represents a great improvement in terms of

diffusion and capitalisation of the know-how of a company.

Nevertheless, the final aim is very often to develop and use a computerised application using the modelisation of the problem (obtained with the method). [THO 90]; [YVA 92]; [COU 93].

To do that, apart from the methodologies, the followings are required:

1 a modelisation language: DDL (Design Description Language).
This language is used to write a computer program based on the results of the DDAM application to the given design problem.

2 a development tool: DAS (Design Advisory System)
This tool allows the user to produce a runable application based on the DDL code.

14.5.2. The knowledge representation language: DDL

In DDL, we find the set of models used in DDAM in our operational form. Each element of the model is represented by an object Class with its associated methods.

1 Product model
- 1.1 Functional model (concept, technical solution classes)
- 1.2 Physical model (assembly, parts classes)
- 1.3 Geometrical model object classes with their methods.

These allow the connection to traditional CAD system.

2 Design process model
- 2.1 hierarchy of task objects
- 2.2 description of the solving methods as objects
- 2.3 expression of basic problem solving mechanisms (heuristics, backtrack, etc..)

14.5.3. Implementation

The implementation of DDL as a knowledge description language implies that the host language has the following capacities:

1 object oriented hierarchy with classes and methods
2 constraints expression
3 usual control structure (loops, task)
4 dynamic memory management
5 possibility of integration within CAD systems

For the technical implementation choices, the C++ language has been selected. It offers the object oriented capacities as well as the classical programming language facilities.

We have also selected a constraint programming library called Solver. This library offers a set of predefined constraints (on variables, objects, classes, sets, ...), as well as extension possibilities (specific constraints programming). The system is being specified.

14.6. CONCLUSION

We have seen how the DDAM and KADS methodologies answer the design problem of the cylinder head.

The DDAM method allows the designer to decompose in detail the product and the process .

The process can be different from the one usually used by the designer. Improvements are expected from this reorganisation of the product structure and process.

The DDAM development is to be continued during the other phases of the project. Indeed, the structures of the models described in DDAM are to be reused as data structures in the DDL.

Furthermore, the specifications of the data required for the implementation of a design problem in DDL enable the identification of missing though necessary information.

The development of the cylinder Head design application will validate the works led within the Deklare project.

14.7. BIBLIOGRAPHY

[Alb 92] P. Albert et al., "KADS-TOOL, a case tool for the CommonKADS methodology", in conference of International Association of Knowledge Engineers, 1992.

[Bou 93] Bousquet I., Marty C., Terrier H., "KADS-TOOL, guide méthodologique", ILOG, France, 1993.

[Cha 90] Chandrasekaran B., "Design Problem Solving: A Task Analysis", AI Magazine, winter 1990.

[Cou 93] Court P., Valois G., Yvars P-A., "Systèmes à base de règles d'ingénierie et CAO, leur utilisation dans le groupe PSA", Proceedings of MICAD 93, Hermès, 1993.

[Dek 93a] Deklare, WP1 Leader & DEKLARE Consortium, "Conceptualisation of Design Analysis Methodology Rev B4", DEKLARE - ESPRIT Project 6522, 9 September 1993.

[Dek 93b] Deklare, WP1 Leader & Deklare Consortium, "Test-Case Reports R1.3/R1.4", Deklare - ESPRIT Project 6522, 9 September 1993.

[Dek 93c] Deklare, WP1 Leader & Deklare Consortium, "Survey of Design Analysis Methodologies", Deklare - ESPRIT Project 6522, February 1993.

[Ede 90] Eder E.W., "Design science - Meta-science to engineering design", Department of Mechanical Engineering, Royal Military college of Canada, Kingston, Ontario, Canada, 1990.

[Fin 89a] Finger, S. and Dixon, J. "A review of research in mechanical engineering design. Part I: Descriptive, prescriptive, and computer-based models of desing processes". Research in Engineering Design 1 (1) , 1989.

[Fin 89b] Finger, S. and Dixon, J. "A review of research in mechanical engineering design. Part II: Representations, analysis and design for the lives cycles". Research in Engineering Design 1 (2) , 1989.

[Gru 91] Gruber T., Al, "Design Rationale Capture as Knowledge Acquisition: Tradeoffs in the Design of Interactive Tools", Machine Learning, Proceedings of the Eighth International Workshop San Mateo, CA USA, 1991.

[Ilo 92] "KADS-TOOL User Manual", Ilog, 1992

[Ilo 92] "SMECI Manuel de référence", Ilog, 1992

[Pah 88] Pahl, G. and Beitz, W. "Engineering Design", Spinger Verlag, 1988.

[Shr 92] Shreiber, G. and Wielinga, B. and Breuker, J., "KADS a Principled Approach to Knowledge-Based System Developpement", Academic Press, 1992

[Tho 91] Thoraval P., "Systèmes Intelligents d'Aide à la conception: Archix & Archipel", PhD Thesis dissertation, University of Technology of Compiègne, France, 1991.

[Tro 90] Trousse B., "Architecture réflexive pour la coopération entre systèmes à base de connaissances et outils de CAO", Proceedings of MICAD 90, Hermès, 1990.

[Wie 92] Wielinga, B. and Van de Velde, J. and Shreiber, G. and Akkermans, D., "The CommonKADS framework for Knowledge Modelling", AAAI Knowledge Acquisition Workshop 1992, Banff, Canada, 1992

[Yva 92] Yvars P-A., Court P., "Techniques d'intelligence artificielle et CFAO: CLEAN, un système d'aide à la conception d'essuie-glace de véhicule", Proceedings of MICAD 92,Hermès, 1992.