

A Thesis Submitted for the Degree of PhD at the University of Warwick

Permanent WRAP URL:

<http://wrap.warwick.ac.uk/108798/>

Copyright and reuse:

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it.

Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk

**An Object Oriented approach to Automating the
Product Specification Concept in the Automotive
Industry.**

by

Panayiotis S. Markomichalis

Submitted in fulfilment of the degree of Ph.D.

Department of Engineering

University of Warwick

February 1991

VOLUME 1

I

grandchild: Grandad tell me how far shall I go?

grandfather: As far as you can.

grandchild: But this is easy.

grandfather: Then, go as far as you cannot.

Nikos Kazantzakis.

ABSTRACT

The research in this thesis is focused around the control of rapid automotive product specification changes which are due to multiple and unexpected factors ie. legal requirements, technological improvements, climate conditions.

Automotive companies use the Product Specification Concept which consists of a multidisciplinary theory using Boolean logic as the applications environment and a team of auditors - people who check the validity of such a theory - to control the complexity of the changes in its products.

Although the specifications data are stored electronically in data bases, the core of such business is dependent on the knowledge and experience of people within the automotive companies and still generally operates manually. Thus, human characteristics have an affect upon the business (ie. the inability of people to work with codes and many different data at once, people tend to forget or they lack proper training and skills, etc.) which makes it less efficient and consequently more costly.

In this thesis possible ways of computerising such an environment (specifically, Rover's Auditing function and Product Specification Concept) are investigated. The characteristics of the problem domain indicate the need to use knowledge based reasoning and Object Oriented Programming.

A system, **ROOVESP** (Rover's Object Oriented VEHICLE Specification) was developed as the "vehicle" to explore the area and it proved that knowledge and experience can be automatically acquired from the existing data and procedures. When these are coded into rules, computer intelligence can contribute to this traditionally human oriented environment and automate fully both the Auditing area and the Product Specification Concept in Rover.

The techniques adopted were proved applicable to other similar areas.

ACKNOWLEDGEMENTS

I would like to express my thanks to the following people for their encouragement and cooperation in writing my thesis.

My supervisor Dr. Amanda Dowd who patiently spent a great deal of time with me and helped me to express myself clearly.

The people in the ATC (Advanced Technology Centre) who have helped me: Dr. Peter Davies and Mrs. Wendy Crees who helped me in many various ways when I needed it.

People from Symbolics: Guy Footring, Karl LLOYD and Ian Banks who provided special courses and helped during the system development as well as entrusting me with confidential information.

People from Rover: Mike Hughes, Dez Quenan, John Layes and Rob McLeod who provided the basis of the expert knowledge used in this research and in testing the program.

My wife and my family who bore my absence with great patience and frustration and encouraged me to the completion of this research.

DECLARATION

Some of the figures in the thesis are missing because they are confidential.

ABBREVIATIONS

AFC	Additional Features Chart
AFGIR	Additional Features Group Index Report
APN	Automatin Part Numbering
AS	Applications System
BAFC	Base and Additional Features Chart
BFC	Base Features Chart
BFCIR	Base Features Code Index Report
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CIE	Computer Integrated Engineering
CIM	Computer Integrated Manufacture
DEP	Design Effect Point
IBOM	Illustrated Bill Of Materials
IPL	Illustrated Parts List
MRP	Material Resource Planning
OOP	Object Oriented Programming
OODB	Object Oriented Database
PDL	Product Development Letter
PIMS	Product Information Management System
PROMS	(PROspecs and piMS)
ROOVESP	Rover's Object Oriented Vehicle
SPecification	
TCIR	Territory Code Index Report

TGIR

Territory Group Index Report

VMEA

Variant Model Effect Analysis

TABLE OF CONTENTS

VOLUME 1

CHAPTER 1: INTRODUCTION TO THE PROBLEM.....	1
1.1 Historical development/increase in vehicle complexity..	2
1.2 The need for flexibility.....	5
1.3 The effect of flexibility.....	8
1.4 Additional problems created by complexity.....	10
1.5 Complexity control methodology.....	12
1.6 Problems associated with complexity control.....	13
1.7 Increase in costs due to errors.....	15
1.8 Research objectives.....	18
1.9 Structure of thesis.....	19
 CHAPTER 2: THE PRODUCT SPECIFICATION CONCEPT (PSC).....	20
 CHAPTER 3: HOW LAND ROVER TACKLES THE PSC.....	28
3.1 The Land Rover company.....	29
3.2 Summary on the chapter.....	34

CHAPTER 4: HOW ROVER TACKLES THE PSC.....	38
4.1 Concepts.....	38
4.2 Application tools.....	47
4.3 The Rover's Product Specification Concept.....	55
CHAPTER 5: THE AUDITING FUNCTION.....	66
5.1 An audit example.....	70
5.1.1 phase 1.....	72
5.1.2 phase 2.....	93
5.1.3 phase 3.....	96
5.2 Summary on the chapter.....	97
CHAPTER 6: CHARACTERISTICS OF THE PROBLEM.....	102
6.1 Complexities of the PSC.....	103
6.2 General characteristics of the PSC.....	105
6.3 Complexities of the Audit function.....	106
6.4 Characteristics of the Audit function.....	107
6.5 Why the PSC and the Audit function represent an appropriate field for computer application.....	109
CHAPTER 7: INVESTIGATION ON THE SOFTWARE ENVIRONMENT FOR THE DEVELOPMENT OF THE SYSTEM.....	112
7.1 Abstraction hierarchy.....	112
7.2 "Planning in a hierarchy of abstractions" with Object Oriented knowledge representation flavor.....	123
7.3 Meta knowledge.....	127

7.4	Meta knowledge with Object Oriented knowledge representation flavor.....	129
7.5	Rover's objectives and the project.....	131
7.6	Object Oriented Programming and comparison with the traditional programming and the Relational paradigm.....	141
7.7	Need for real Object Oriented Databases.....	158
7.8	Knowledge representation in ROOVESP	167
7.9	Summary on the chapter.....	171

CHAPTER 8: **ROOVESP** (Rover's Object Oriented VEHICLE Specification). THE FIRST PHASE OF THE AUDIT FUNCTION.....173

8.1	Introduction to the system design.....	174
8.2	Detailed description of the system's components.....	209
8.3	Further description of ROOVESP	255
8.4	Testing.....	264
8.5	Summary on the chapter.....	266

CHAPTER 9: FURTHER IMPLEMENTATION OF THE SYSTEM AND THE PSC (SECOND AND THIRD PHASES OF THE AUDIT FUNCTION).....269

9.1	Original thoughts on the further implementation of the system.....	271
9.2	Intelligent Networks.....	276

CHAPTER 10: FURTHER IMPLEMENTATION OF ROOVESP	306
10.1 Software issues.....	306
10.2 System Design issues.....	315
10.3 Summary on the chapter.....	325
CHAPTER 11: DISCUSSION ON ROOVESP	327
11.1 Comparison of IPL with ROOVESP	328
11.2 ROOVESP in relation to other existing systems in AI.....	351
11.3 How ROOVESP affected the existen environment of the PSC in Rover.....	358
11.4 The new conceptualism introduced in the PSC by ROOVESP	373
11.5 What can be learnt from ROOVESP	378
11.6 The new conceptualism and ROOVESP	385
CHAPTER 12: CONCLUSIONS.....	489
BIBLIOGRAPHY.....	391

VOLUME 2

APPENDIX 1: KNOWLEDGE AND OBJECT ORIENTED PROGRAMMING.

APPENDIX 2: STATICE.

APPENDIX 3: LISTING OF THE VAX TAPE DATA AND THE PARSING PROGRAM.

APPENDIX 4: LISTING OF THE VALIDATION ALGORITHM.

APPENDIX 5: LISTING OF THE STATISTICAL ANALYSIS PROGRAM.

APPENDIX 6: THE STATICE DATABASES DEFINITION.

APPENDIX 7: LISTING OF THE TRUTH MAINTENANCE MECHANISM.

APPENDIX 8: LISTING OF THE INTELLIGENT NETWORKS.

APPENDIX 9: SAMPLE OF THE INFORMATION CREATED DURING THE RUN OF THE VALIDATION ALGORITHM.

APPENDIX 10: SAMPLE RUN OF **ROOVESP**.

APPENDIX 11: INSTRUCTIONS TO RUN **ROOVESP**.

1. INTRODUCTION TO THE PROBLEM

The prime objective of any automotive company is to build vehicles from elementary manufactured parts and/or products assembled either in house or by outside suppliers, in the easiest, fastest and most cost effective way.

Each vehicle is assembled from at least 5,000 different parts [87], from the elementary components such as screws up to the more complicated assemblies such as engine, dashboard, suspension etc. All these parts are linked logically to each other with specific quantitative and engineering rules, creating an extremely dynamic environment. This means that a change in the specification of a part affects not only the assembly design, sourcing data and future implementations of the part itself, but also the overall specification of the particular area of the vehicle in which this part is fitted.

Part specifications do indeed change rapidly, both through the practical implementation of the vehicle from its conceptual phase through to the final production line and due to the pressures of the competitive market. Especially for the latter, the continuing expansion of more cost effective technological solutions have resulted in the increase of the vehicle luxuries which customers can get with the same amount of money. This gradually has led the automotive industry to loose the full share of decision making in the specification of the new vehicles as the customers can now decide on what characteristic of the vehicle to spend their money

dependent on their own taste. In other words, the part of decision making lost by the automotive industry has actually been gained by the customers themselves. As stated in [10] "the need for greater profitability or indeed survival has led volume motor manufacturers over the last quarter of the century, to pay greater attention to what the customer wants".

1.1 HISTORICAL DEVELOPMENT/ INCREASE IN VEHICLE COMPLEXITY

Up until the 1960's the only options available with a vehicle were a limited choice of engine specifications and/or body colours. For example, in the late 1960's Chevrolet produced approximately three quarters of a million Impalas in a year and the customer was happy with a single choice of exterior colour [10]. Today, in order for a manufacturer to achieve half this volume, a significant level of product complexity and consequent sophisticated product control is required.

Tables 1 and 2 show the historical development of factory fitted options for both the British Leyland (nowdays Rover) models and the top of the range European built Fords. Table 1 depicts the availability of variants according to the date whereas table 2 details the existent British Leyland models from 1964 to 1986.

Looking at a specific point in time, the year 1969, there were two British Leyland top of the line models, the P5 3.5 litre and

P6, they could be split to six base derivatives - ie. the coupe

No. of variants available excluding Paint/Trim options

YEAR	1960's	1970's	1980's
Rover	29	136	280
Ford	49	335	544

TABLE 2. Historical development from reference [10].

Rover		Ford	
Model	Year	Model	Year
95/110	1964	Zephyr Mk III	1962-66
P5 3 litre	1958-67	Zephyr Mk IV	1967-72
P5 3.5 litre → Coupe → Saloon	1968-75	Granada/Consul	1972-79
P6 → 2000 → 820i → 820e	1964-72	Granada	1979-85
→ 3500 → 820i → 820e	1977-86		
SD1 → 2.3 → 2.6 → 3.5			

TABLE 2: Model range to date from reference [10].

and saloon versions of the P5 3.5 litre, the 820i or 820e of 2000 litre version of P6 etc. - and each derivative had its own options, offered by Rover, giving more different variants. Excluding trim/paint colour options there were 80 off line

variants available in 1969.

In 1979 the British Leyland top of the line range model consisted of three basic derivatives (ie. SD1 2.3, SD1 2.6, SD1 3.5) fewer than in 1969, but with a much wider range of options on offer. In total there were over 400 off line variants (excluding trim/paint colour) offered to the public.

The current Rover 800 Series offers 14 base derivatives (table 1) and each of them has up to 8 different customer options. This provides over 2376 variants excluding the trim/paint colour options. If trim and paint choices are included the number of variants is over 54,000. Note that these calculations represent only the UK market. The number of variants increases even more when the world market is considered eg. there are 172 different 'base' derivatives for the Rover 800 alone sold to 34 countries [10].

The figures for the top of the European Ford range (table 1) show a similar trend - as the market has developed, the number of variants has tended to increase.

It is also worth noting that a number of vehicle features which were options in the 1960's and even some which were options in the 1980's are now fitted as standard, eg. power steering, central locking. This results in an increase in the complexity of the base vehicle. As a model range develops an increased range of options become available and a number of the original options become standard, so automotive companies have to further increase the offered options in order to maintain sales.

It can be concluded then, that when a new model is introduced, the overall complexity is higher than the previous model launch as the customer now expects a number of items as standard which previously were considered luxuries (figure 1).

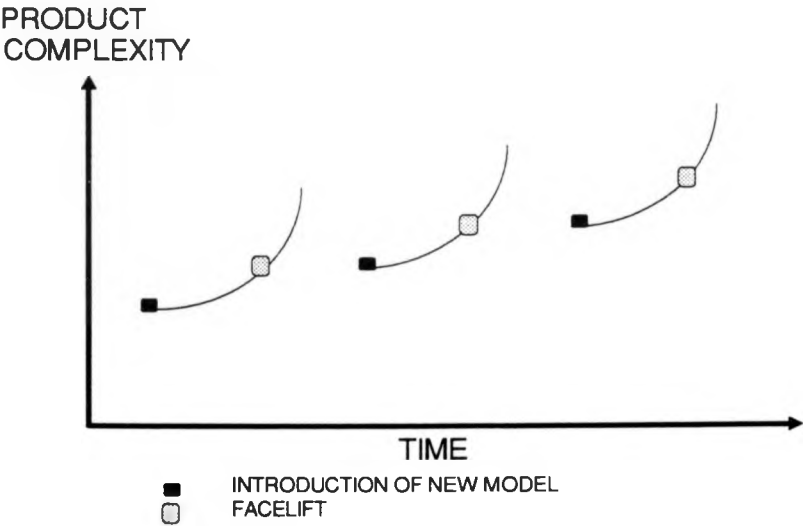


FIGURE 1: Increase of model complexity through time, from reference [10].

1.2 THE NEED FOR FLEXIBILITY

Nowadays people see the products of the motor industry as goods with a variety of affordable options which they can bargain for, rather than the expensive standard product of the past. They also

Category	Item ID	Performance Metrics (Q1-Q4)										Status	Notes
		Q1	Q2	Q3	Q4	YTD Avg	Target	Variance	Growth %	Efficiency	Reliability		
Electronics	Smartphone X1	95	92	98	90	93.75	95.0	-1.25	5.2%	98%	99%	Active	Latest OS update
	Laptop Pro 15	88	85	90	82	86.25	88.0	-1.75	3.1%	95%	97%	Active	Minor hardware repair
	Tablet Ultra	78	75	80	72	76.25	78.0	-1.75	4.5%	92%	94%	Active	Screen replacement
	Smartwatch S3	65	62	68	60	63.75	65.0	-1.25	6.8%	90%	91%	Active	Battery replacement
	Headset VR	55	52	58	50	53.75	55.0	-1.25	7.5%	88%	89%	Active	Software update
	Camera Z1	45	42	48	40	43.75	45.0	-1.25	8.2%	85%	86%	Active	Lens cleaning
	Smart Home Hub	35	32	38	30	33.75	35.0	-1.25	9.1%	82%	83%	Active	Reset and reconfigure
	Wireless Earbuds	25	22	28	20	23.75	25.0	-1.25	10.5%	80%	81%	Active	Charging case repair
	Smart TV 55"	15	12	18	10	14.25	15.0	-0.75	12.0%	78%	79%	Active	Remote control replacement
	Smartwatch S3	5	2	8	0	3.75	5.0	-1.25	15.0%	75%	76%	Active	Screen cracked
Software	Operating System	90	88	92	85	88.75	90.0	-1.25	2.5%	99%	99%	Active	Security patch
	Productivity Suite	85	82	88	80	83.75	85.0	-1.25	3.8%	97%	98%	Active	Feature update
	Cloud Storage	75	72	78	70	73.75	75.0	-1.25	4.5%	95%	96%	Active	Storage expansion
	Security Suite	65	62	68	60	63.75	65.0	-1.25	5.2%	93%	94%	Active	Virus scan
	Backup Software	55	52	58	50	53.75	55.0	-1.25	6.0%	91%	92%	Active	Configuration update
	Monitoring Tools	45	42	48	40	43.75	45.0	-1.25	7.0%	89%	90%	Active	Alert setup
	Collaboration Tools	35	32	38	30	33.75	35.0	-1.25	8.0%	87%	88%	Active	Integration update
	Analytics Platform	25	22	28	20	23.75	25.0	-1.25	9.0%	85%	86%	Active	Report generation
	CRM System	15	12	18	10	14.25	15.0	-0.75	10.0%	83%	84%	Active	Lead management
	ERP System	5	2	8	0	3.75	5.0	-1.25	11.0%	81%	82%	Active	Inventory tracking
Hardware	Server Rack	90	88	92	85	88.75	90.0	-1.25	1.5%	99%	99%	Active	Hardware refresh
	Storage Array	85	82	88	80	83.75	85.0	-1.25	2.0%	97%	98%	Active	Capacity expansion
	Network Switch	75	72	78	70	73.75	75.0	-1.25	2.5%	95%	96%	Active	Firmware update
	Router	65	62	68	60	63.75	65.0	-1.25	3.0%	93%	94%	Active	Configuration change
	Firewall	55	52	58	50	53.75	55.0	-1.25	3.5%	91%	92%	Active	Policy update
	UPS System	45	42	48	40	43.75	45.0	-1.25	4.0%	89%	90%	Active	Battery replacement
	Switch Stack	35	32	38	30	33.75	35.0	-1.25	4.5%	87%	88%	Active	Redundancy setup
	Access Point	25	22	28	20	23.75	25.0	-1.25	5.0%	85%	86%	Active	Signal strength
	Network Card	15	12	18	10	14.25	15.0	-0.75	5.5%	83%	84%	Active	Driver update
	Modem	5	2	8	0	3.75	5.0	-1.25	6.0%	81%	82%	Active	Service provider
Services	Cloud Migration	90	88	92	85	88.75	90.0	-1.25	1.0%	99%	99%	Active	Project completion
	IT Support	85	82	88	80	83.75	85.0	-1.25	1.5%	97%	98%	Active	SLA compliance
	Managed Network	75	72	78	70	73.75	75.0	-1.25	2.0%	95%	96%	Active	Proactive monitoring
	Security Audit	65	62	68	60	63.75	65.0	-1.25	2.5%	93%	94%	Active	Compliance report
	Disaster Recovery	55	52	58	50	53.75	55.0	-1.25	3.0%	91%	92%	Active	DR plan update
	Backup Service	45	42	48	40	43.75	45.0	-1.25	3.5%	89%	90%	Active	Retention policy
	Monitoring Service	35	32	38	30	33.75	35.0	-1.25	4.0%	87%	88%	Active	Alert management
	Collaboration Tools	25	22	28	20	23.75	25.0	-1.25	4.5%	85%	86%	Active	Integration testing
	CRM Integration	15	12	18	10	14.25	15.0	-0.75	5.0%	83%	84%	Active	Lead management
	ERP Integration	5	2	8	0	3.75	5.0	-1.25	5.5%	81%	82%	Active	Inventory tracking

Table 3: Rover's options demand forecast table for 1990.

tend to purchase vehicles "tailored" to their personal taste or job requirements. For example in a discussion in a showroom today the following statements might be heard:

- I want a Montego for Europe
- I want an estate
- I want a 1600 engine with an automatic gearbox
- I want a stereo radio cassette player, leather seats e.t.c.

It's actually the customer who "runs" the conversation and the motor manufacturers have to satisfy the majority of his preferences in order to survive the competition. They must also meet the different requirements of the world market ie. local legislation, climatic conditions, etc. For this reason, motor manufactures are today moving towards the maximisation of the **feature** options that they could supply to a new vehicle, together with the highest flexibility in their design.

It can be seen from the sales forecast table 3 that there are several model variants for which there will be very low requirement. Nevertheless, it is a marketing decision that all these options be offered to meet potential demand despite the complexity added to the product line. For example, Rover offers two different cassette players for all basic derivatives of Maestro and Montego models, although the average demand forecast is only 2% and for some derivatives there is not even that demand

(0%!).

1.3 THE EFFECT OF FLEXIBILITY

Thus, the *Product Derivative Complexity* (as the increase in variety of the product from a common base vehicle is termed) to a large extent enables companies to maintain sales and control their success in the market. The game of gaining the maximum benefits depends upon the skill in all areas of the company in managing complex derivative programmes and large numbers of combinational variants. Today for example, for only a single model, the right hand drive ESCORT L, the Ford motor company offers 3000 different **feature** options to its customers. This can be translated literally to 2.3 BILLION Ford-Escort-L variants for all the possible combinations of the feature options! Although, this number of Escort-Ls are not actually built in the production plant, 200,000 of different Ford Escort L variants have been offered worldwide and 70% of all option-combinations are used per annum [27]. In addition, almost 5000 different parts - assemblies and components - are required for the building of this car and nearly all of them are used for every variant, implementing each time a hard Specification (or Product) Control exercise.

Table 4 compares the current option availability of Rover 800 with two of its direct competitors, the Ford Granada and the BMW

Option Availability

BMW 5 SERIES

1.	520i	520iSE	525i	525iMi	530i	530iSE	535i	535iSE	Sport
2.	1.15x10 ⁹	1.42x10 ⁷	3.44x10 ⁹	4.25x10 ⁷	5.73x10 ⁹	2.35x10 ⁶	1.13x10 ⁹	4.72x10 ⁶	2.95x10 ⁵
	95	95	95	95	95	95	95	95	95

(3) = 5.07 x 10¹⁰

ROVER 800 SERIES

1.	020i	020e	020i	020e	020i	020e	020i	020e	027Si
2.	32	256	120	256	120	256	120	256	256
	24	24	24	24	24	24	24	24	24

(3) = 54160

FORD GRANADA/SCORPIO

1.	Taxi	LX 2.0	LX 2.0EFi	LX D	GL 20EFi	GL D	GLia 2.0i	GLia 2.9i	GLia D
2.	2	0	0	2	4	1	16	16	0
	9	9	9	9	20	20	20	20	20

(J) = 2371

1. = No. of Variants Available excluding Paint/Trim Colour Options
2. = No. of Paint/Trim Colour Options
3. = Total No. of Variants

Table 4: Availability of options in Rover 800 in comparison with Ford Granada and BMW Series.

5 Series, as was mentioned earlier the number of Rover 800 variants including colour and trim options is approximately 54,000 for the domestic British market. This compares to 2,371 for the Granada and 5.07×10^{11} for the BMW 5 Series [10].

The large difference in figures is due to the various marketing policies followed by the companies. Although Ford offers as many as 3000 options for the Ford Escort, it only offers a few options for the higher range Granada model and maintains a large number of the remaining 3000 options fitted as standard to the vehicle. BMW offer two trim levels on each BMW 5 Series model (ie. 520i and 520iSE for the BMW 520 model) plus a top of the range sport model. It is then possible for many more features to be fitted in the vehicle because of the relatively high number of optional alternatives. Consequently, BMW follows the same marketing policy for its products irrespective of the model range.

Rover falls between the two policies. If the top of the model range there are few options but still more than Ford offers and more choices in the middle of the range though they are much less than 3000. There are 300 options offered for the Rover R8 model, the Rover equivalent of Ford Escort [95].

From these data can be seen the problem of complexity faced. It is the large number of options which leads to the large number of variants - only with 25 options available, there are over 30 million variants available, provided the options are mutually exclusive.

1.4 ADDITIONAL PROBLEMS CREATED BY COMPLEXITY

As the overall product complexity is increasing it applies an incremental workload and cost not only in Specification Services but to all associated support functions of the engineering design and development of the product itself. The situation becomes more difficult when one considers the additional 1980's automotive needs to integrate Specification/Bill Of Materials (BOM) data with the existing Computed Integrated Engineering (C.I.E.) and/or Computer Integrated Design (C.I.D) databases in order to control complexity.

Additional pressure for correct specification, so that Rover together with all the other European and North American Automobile companies are able to compete with the Japanese motor manufacturers, means that a high proportion of time of their Design Engineers is spent in Auditing instead of Design (Auditing in the contexts of this thesis means the validation of the various documents within Rover which specify the vehicles of the company).

	<u>Europe</u>	<u>Japan</u>	<u>USA</u>
DESIGN TIME - MONTHS	60	47	60
DESIGN EFFORT - MAN HOURS (millions)	3.2	1.7	3.0
MODELS IN PRODUCTION	48	72	36
AVERAGE REPLACEMENT PERIOD - YEARS	9.2	4.2	9.2
AVERAGE ANNUAL PRODUCTION OF EACH MODEL (000's)	200	120	230

TABLE 5: Design time and effort in Vehicle Design from reference [10].

The Japanese follow the philosophy that Manufacturing should move upstream to have an input in Product Strategy instead of waiting until the product is practically engineered and in that way they have achieved a reduction in Design lead time (table 5).

In summary, flexible and efficient control in vehicle specification changes, thus more efficient control in product complexity, is the core concept and reality for any automobile company.

1.5 COMPLEXITY CONTROL METHODOLOGY

Automotive companies have implemented a methodology for controlling and communicating the complexity of their products. A 'language of discourse' of expressing and manipulating vehicle characteristics with its own semantics and syntax. This is called the **Product Specification Concept** (PSC or Product Definition). The term Product Specification Concept (PSC) is used by Rover and it will be used in here for any automobile company. PSC is discussed in more detail in the following chapter. The way of issuing information within the company on the principle of the Product Specification Concept is the **release system**. A release system is simply a method of communication. Basically, it says make or buy this part to this shape, in these colours, in this material. It represents a final statement from the engineer that

he has completed his part of the design. In small companies the release may be merely the handover of the drawing, together with some verbal information regarding timing, volumes, materials etc. As companies get bigger, the method of design transmittal becomes more and more complex. Responsibilities are fragmented, and the engineer becomes increasingly remote from the actual manufacturing process.

As product offerings expand as well, the complexity of product combinations impacts upon the design engineers's need to communicate their intent and further implementations, combinationally increasing the amount of data and the actual circulation of their release.

1.6 PROBLEMS ASSOCIATED WITH COMPLEXITY CONTROL

Figure 2 shows the percentage shares of changes introduced to the specification of a vehicle by each of Manufacturing, Administrative and Engineering department in Rover. These changes occur during its conceptual stage right through to the final production line, which can take from three to five years, as two statistical surveys revealed at May 1987 and June 1988 [16]. It is worth mentioning that many of these changes may be the result of human error.

From early times Rover and other motor manufacturers realised the impact that such mistakes could have on cost and time in the

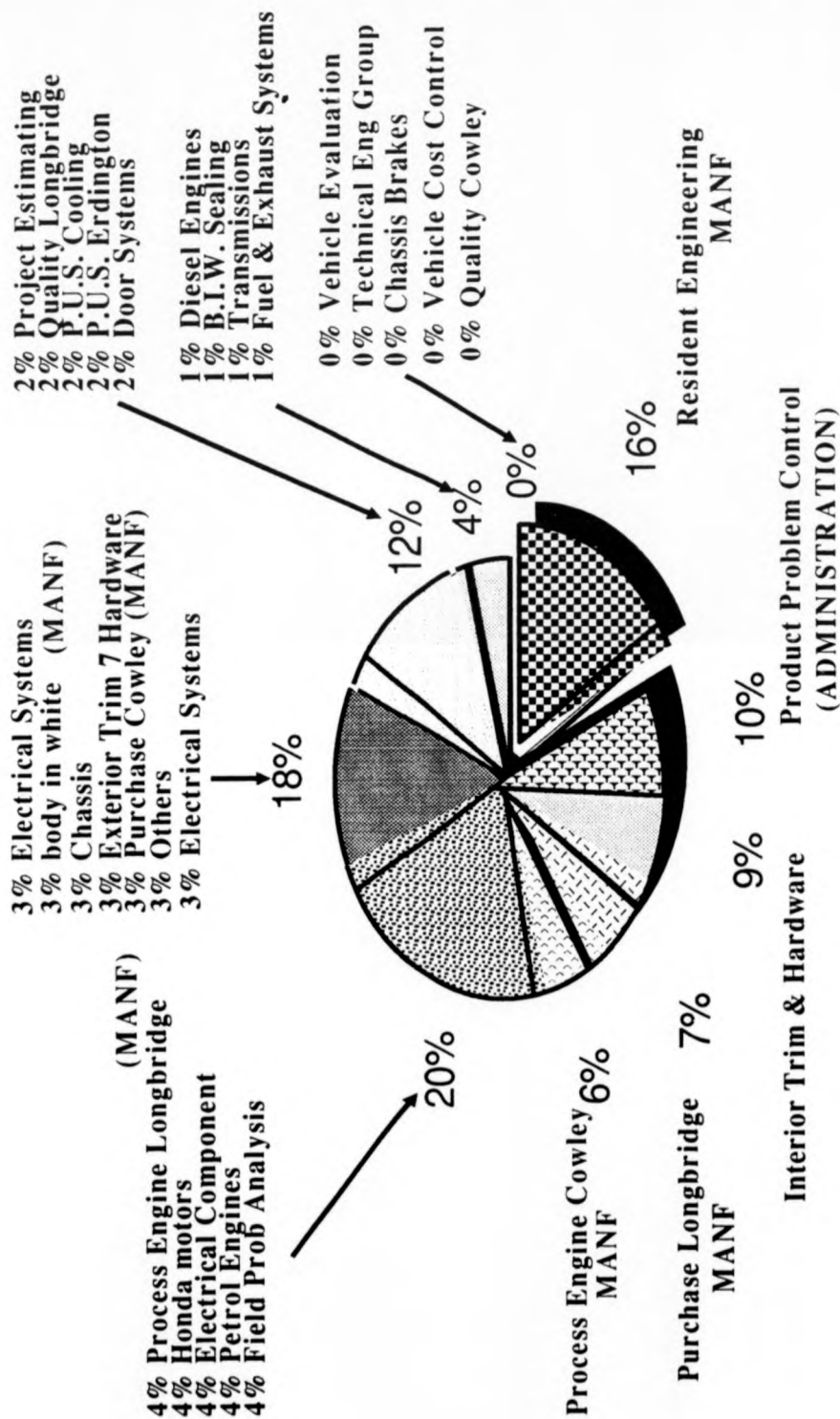


figure 2: Percentages of changes - due mainly to mistakes - in Manufacturing, Administration and Design engineering, from reference [16].

company and launched a new department within the specification services in order to maximise the control upon such inconcinstencies; the Auditing department. Also, it is the extremely delicate and dynamic nature of Motor industry that makes the Product Specification control and/or auditing problem recurrent.

1.7 INCREASE IN COSTS DUE TO ERRORS

The values in figure 2 represent in reality the result after the application of a well experienced Auditing function in Rover. The question which arises is how such a function could be improved in order to pick up the human error at the earliest possible development stages of the vehicle rather than being discovered later in the production circle where they could cost the company dearly.

It has been recorded that 5,587 changes occurred during the 18 months development of the Rover model R8, though there were 13 circulations (build phases) and re-examinations of the model's release package and an auditing function was in operation for each circulation (table 6) [11].

More precisely, a major number of the 3,521 changes before the Methods build phase could be avoided and save the company money and time as the prototype vehicles which are built in phases D01 and D02 tend to be extremely expensive and time consuming. This is because all the parts and tools used to make the them were

Methods phase, ie. Purchasing the wrong tools and/or parts, updating the data base, liaising with the Design Engineers, etc. After the Methods build phase mistakes happen but it is questionable whether the Auditing function could prevent most of them. It has been estimated [11] that after the Methods phase the administrative cost associated with a single mistake is £15,000. This is for changes in purchasing, scheduling, engineering work, update of the data, parts and accessories, paper work, reissuing of the drawings, inventory and spacing, labour disposal of the old parts and palleting the new ones, publishing of new brochures etc. In the Rover R8 model example, the total cost of changes after the Methods build phase came up to many million pounds and people within the company believe that this cost could be reduced drastically with a more thorough audit appliance and control on the whole scope of the release system. Some of them could be picked up. However, there are examples of mistakes which were discovered by customers and the Auditing function was in the main responsible.

Land Rover, as well, though representing a less complex manufacturing environment by tradition, has faced similar problems in the past.

As it is expected, the major automobile companies - Rover, Ford, Honda, Peugeot etc. - have already implemented their own audit based release systems in one form or another based on their own particular Product Specification Concept. Also, the problem of overlooked human mistakes which always appear in motor industry, forces Management to investigate alternatives for a more

efficient Auditing process.

1.8 RESEARCH OBJECTIVES

The original objective of this research was to investigate the means of automating, as far as possible, Rovers's existing engineering audit function. However, during the initial phases of research, it became apparent that it is not feasible to merely automate the audit function, the whole Product Specification Concept needs to be considered. This lead to a revised objective of:

Investigate possible changes to the Product Specification Concept which would enable this and the Engineering Audit function to be automated.

This then involved the following sub-objectives:

1. Fully investigate the current Product Specification Concept and the associated Audit procedures.
2. Characterise the problem.
3. Determine alternative approaches which could be

adopted.

4. Build a working system to explore the feasibility of these alternatives.

1.9 STRUCTURE OF THESIS

In chapter 2, the PSC is introduced in more detail in the way it currently applies within Rover. The Rover paradigm represents the general way of the application of the PSC in the automotive industry. Chapter 3, investigates another 'variant' of the PSC, that of Land Rover.

Chapter 4 tackles, more specifically, the Rover's PSC - the 'sub-concepts' on which the general PSC is built and its application tools used by Rover - and some of the reasons that such an existing manual process impacts on specification mistakes.

Chapter 5 introduces a typical manual audit example in the way it currently operates in Rover. The audit function is investigated first, in order to put the basis for the better understanding of the PSC in the company. Chapter 6 analyses the characteristics of the problem - both the audit function and the PSC - and why such a problem might be suitable for automation.

Chapter 7 investigates the computer literature, both system

analysis techniques and knowledge representation, which would be more appropriate to automate both the Audit function and the PSC, according to their characteristics.

Chapter 8 introduces the design of the **ROOVESP** system - standing for the Rover's Object Oriented VEHICLE SPecification - and its implementation in three successive phases. Chapter 8 is concerned mainly with the automation of the Audit function. Chapter 9 completes **ROOVESP** by introducing the Intelligent Networks Prototype which tackles the PSC in the company as a whole. Chapter 10 suggests the further developments of the existing system, which could fully automate the Rover's PSC around **ROOVESP**. Chapter 11 discusses both software and design issues which were met during the implementation of **ROOVESP**. Additionally, the general knowledge that has been achieved from this research is pointed out which can help to solve problems of similar characteristics.

Finally, the chapter 12 summarises the state of the problem before this project started and the objectives achieved after this research, as well as the objectives which can be achieved by following the conceptualism of **ROOVESP** and further implement the system.

The appendices 1 and 2 introduce the **FLAVORS** and **Static** environments which describe further the software tools used in **ROOVESP**. The rest of the appendices (appendix 3 to 11) list the program which implements **ROOVESP** and run samples of the system.

2. THE PRODUCT SPECIFICATION CONCEPT

The literature of major automobile companies expresses concepts and attributes of existing specification systems in different ways, although the majority of terms which they use it reflects a similar, generally adopted "logic" for vehicle specification. For the consistency and clarity of the terminology that will be used, the terminology used at Rover has been adopted, except in the situation where the source of the terminology is clearly defined somewhere within the document.

The automobile literature shows that the Product Specification Concept (PSC) has been considered by many automobile companies to be the most efficient means by which they can tackle the product complexity problem.

The PSC in its general form was introduced by the American automobile companies early in 1960's [17]. It was the first time that vehicles started to be designed for longer marketing life in order to meet flexibly the newcoming expansion of the motor industry and the expected increase in customer requirements.

The automobile companies handled the increasing product complexity problem at first manually up to the time that the amount of data processing became overwhelming, at which point they switched to information technology solutions. However, although the engineering data are manipulated electronically

nowdays, the automation of such a process is concerned only with their storage retrieval and maintenance. The general flow of information within the Product Specification has not been changed, especially the manual audit of the Product.

In order to introduce the Product Specification Concept, an example will be used to show how this concept can specify a product and reduce the workload in a company. The Product Specification Concept organises old information and uses it to set up a new model variant.

<u>Trim Level</u>		<u>Trim Level</u>
<u>Part</u>	5	5.1
A	✓	✓
B	✓	✓
C	✓	✓
W	✓	
✓		✓

As many as 90% of parts needed re-specifying in trim level 5.1 as these were COMMON PARTS

This part remains UNCHANGED and does not go on the new model variant.

This part is a NEW part that has been introduced for the new Trim Level only.

Table 7: The "bulk" of re-specification is shown, from reference [5].

The example assumes that an M.G Metro variant is already in existence, and Rover decides to introduce a *Turbo* version, the M.G. Metro Turbo.

METRO, Trim Level 5, 3 DOOR, M.G 1300 - **OLD** VARIANT

METRO, Trim Level 5.1, 3 DOOR, M.G 1300 Turbo - **NEW** VARIANT

When the new model variant is introduced, a new trim level - ie. trim level 5.1 - is allocated by Rover to identify the small number of parts that differ from the old variant. Without the PSC the effect of creating the new trim level means, that all the existing parts of the variant must be re-specified to the trim level 5.1 (a major task).

However, PSC does not allocate all the existing parts to the new trim level, instead it specifies changes in the general definition of the original Metro derivative M.G 1300 and generates a new model variant. In that way only those parts which differ between the two model variants (modified and new) are specified. This saves considerable time and reduces potential sources of error as the 90% common parts continue their existence in the vehicle unchanged simply as 'carry-over' parts.

The PSC has to refer to some criteria to work with in order to perform the required design changes and consequently control the overall specification function. Such criteria/attributes developed from system analysis in Vehicle Design are known as product **features**. In reality they represent the 'semantics' of the PSC.

"A feature is a characteristic of a vehicle that causes Parts to be fitted to that vehicle and which may cause other parts not to be fitted to the vehicle" (Rover definition).

The PSC in Rover recognises two categories of features:

a. **BASE FEATURES** representing the majority of parts on a vehicle (70-80 per cent).

- marque
- model
- trim level
- body style
- engine
 - family
 - capacity
 - performance
- drive
- transmission

b. **ADDITIONAL FEATURES.** These define the remaining feature content of the vehicle:

- features required to make a complete vehicle
 - eg. headlamps ,instrument packs,
 - emission or legislative requirements.
- features relating to options
 - eg. alloy wheels, sunroof
 - glass
 - steel
 - air-conditioning. etc.

The general idea is that by using the specification concept and especially using its features (its semantics) people within the company can identify any vehicle variant both unambiguously and yet flexibly through manufacturing codes. These are data strings (with the PSC's syntax eg. '+', '-', etc.) which contain the Engineering information about the product and are called the **usage statements** of the individual parts or the **part usages** in an assembly. So a manufacturing code would be something like "A10X+F47K" where each such code refers to the specification of an individual part which is going to be assembled together with other part to make up the car.

For example, for the following Rover 800 model variant, the sunroof part of this car is specified as sliding glass to be fitted:

Part in concern:

sunroof

	BASE FEATURES	ADDITIONAL FEATURES
CODE:	A10X + A20S	+ F47K
DESCRIPTION:	Rover 800 Series	Glass sliding sunroof
	(a positive Usage Statement)	
	(A10X is translated to Rover, A20S means 800 series whereas F47K means Glass sliding sunroof)	

For the following statement of a car heater, an air condition is not fitted to the Metro model variant:

Part in concern:

heater

	BASE FEATURES	ADDITIONAL FEATURES
CODE:	A10X + A20F + A50A	- C88A
DESCRIPTION:	Rover Metro Right-Hand-Drive (RHD)	Air cond.
(a negative Usage Statement)		

The (-) minus sign denotes that the feature is *not* fitted, whereas the (+) plus sign denotes that the feature should always fitted.

Appropriate management at the usage statements level of the parts of a vehicle, provides only the new feature options or their combinations which are required for the model variant and decreases rapidly the volume and cost of redundant re-specification. Hence, 'thinking' with the Product Specification Concept, design engineers can refer to the usage statements of the part and not the part itself. This makes things ever easier for the company as it not only responds flexibly to the various phases at the conceptual design of the part but also helps Manufacturing to avoid unnecessary changes at the assembly levels.

In summary, the Product Specification Concept has been adopted worldwide in automobile industry. Its well specified syntax and semantics, represent a common language of reference in use within the company which helps departments with different knowledge,

skills and objectives to communicate without distortion of the original information; the customer's requirements. It enables people from Vehicle Directorate/Line to "compile" the documentation from Marketing as features and engineers to conceive and design the new requirements based on those features combination and supply Manufacturing with appropriate assembly structures. The PSC as a common language of reference needs a way of transmitting its information through the various departments of the company. This is the release system, discussed in the previous chapter, that every automobile company has adopted to support the concept.

It must become clear that the term PSC in the contexts of this thesis will mean both the common language of reference (the concept methodology) and the flow of its information within the company (the release system).

3. HOW LAND ROVER TACKLES THE P.S.C.

In order to proceed into the actual mechanics which drive the Product Specification Concept (and release system) within Rover and understand the major need for a thorough audit function to make the PSC valuable we will overview first the surrounding literature of some other automobile companies which face the same problem and more specifically for Rover we will discuss the basic concepts which supported such a theory and the tools which the company has implemented through time for its appliance.

The confidentiality that covers such a highly competitive environment as the motor industry, makes it extremely hard to perform a comparison of the way in which different companies approach the common Product Specification problem based on the existing automobile literature. Motor companies become even less informative when the interested party represents one of their competitors. Besides, the relevant academic literature in Change Control, M.R.P.1&2 (Material Resource Planning and Manufacturing Requirements Planning), C.I.M. (Computer Integrated Manufacturing), C.A.D. (Computer Aided Design) and B.O.M. (Bill of Materials), covers only the general domain of similar problems in Engineering and Manufacture.

For those reasons the scope of the investigation for other motor manufacturers is limited only to Land Rover.

3.1 THE LAND-ROVER COMPANY

At this point, it must become clear that when reference to the Rover specification system has been made the author has been considering the specification system of the division previously known as Austin Rover.

The part of the Rover previously called Land Rover has implemented its own specification/release system to provide management and control in the specification complexity of its products. So, it makes use of the Product Definition concept as it is known within Land Rover.

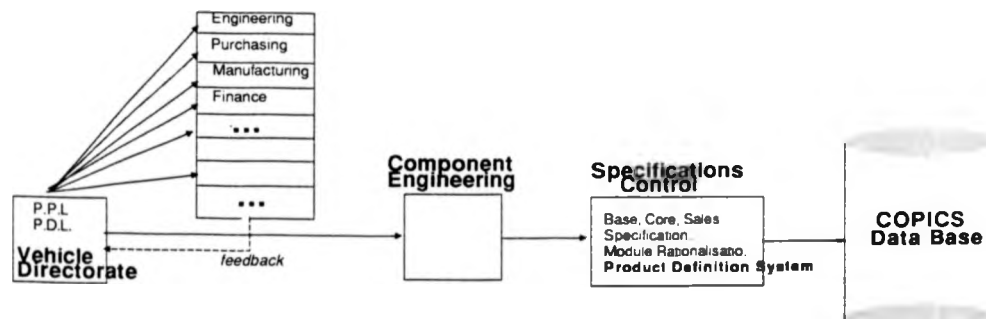


Figure 11: Land Rover's PSC.

The Vehicle Directorate within Land Rover originates the Product Development and Product Policy Letters which establish the

design intent of the new Vehicle. Those documents are passed to all departments concerned with the new product information ie. Manufacturing, Purchasing, Engineering, Finance etc. When all feedback of the new product is collected by Vehicle Directorate then Component Engineering is authorized to create the design and documentation of the product. The source document of the vehicle is compiled and audited from the Specification Control department and loaded to COPICS (Computer Oriented Product Information and Control System), the IBM mainframe data base product used by Land Rover. The Specification control department makes use of the Product Definition concept in order to specify and audit the vehicle.

The Product Definition Concept defines a basic vehicle *framework* independent of product changes. The approach that the concept follows is termed as modular approach and is achieved by dividing every product into 53 groups and working out the right 53 groups to be fitted into the original framework. Each group is called the Module Family and represents a section of the vehicle framework (figure 12). Some examples of module family are:

WT = Wheels & Tyres

GL = Glazing

DR = Doors

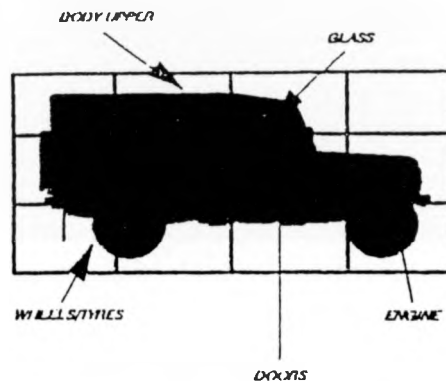
EN = Engines.

Within each module Family there are a number of specific Modules.

SPECIFICATION

(LAND ROVERS INTERNAL PROCESSING)

- INITIALLY WE DIVIDE THE PRODUCT INTO 53 SECTIONS CALLED "MODULE FAMILIES"
- EACH MODULE FAMILY REPRESENTS A UNIT OF PRODUCT BUILD REQUIREMENT



- A COMPLETE PRODUCT MUST HAVE ONE ELEMENT FROM EACH MODULE FAMILY. THESE ARE CALLED MODULES.
- CONNECTED TO EACH MODULE IS THE TOTAL PART REQUIREMENT

Figure 12: The framework of modules in Land Rover's Product Definition Concept, from reference [94].

For example the EN family would collect all the Engine modules.
An example is :

EN003 = a 4 cylinder petrol engine with an oil cooler
and evaporated Loss control for Gulf.

At the beginning of 1989 there existed approximately 1500 modules, each one allocated to a specific family and while some of them may have been deleted or new ones created, it was always within the same family. For a vehicle to be completely specified ONE module of each module family is chosen. The process occurred at 3 levels:

- BASE SPECIFICATION (70% of the definition)

The left hand steering vehicles represent the largest percentage of the company's sales worldwide. The company, at the beginning of the specification of the vehicle, tries to cover the wider range of customer requirements, hence all the Modules in this first stage of the specification of the vehicle are allocated to create products which are left hand drive.

eg. M_1 module + M_2 module + M_k module = *Base Specification*

- CORE SPECIFICATION (10% of the definition)

Land Rover has divided the 170 customer countries around the world into 25 areas called "Build to Regions" depending on legal, marketing or Engineering requirements with common specification. The Core Specification links the product to the "Build to Region" area and in that way adds all the legal, marketing or engineering requirements. More modules are added to the specification of the product. eg.

$M_1 + M_2 + \dots M_k + M_{k+1} + \dots M_1 = (Base + Core) \text{ specification.}$

At that stage the vehicle is almost fully specified.

- SALES SPECIFICATION

The third and final stage completes the specification by adding the rest of optional features :

- alternative features eg. trim level and colour. (for example Range Rover has 15 colours and 4 trim levels)
- additional features eg. sunroof, antilock brakes etc.
(ie. Land Rovers have a choice of up to 67.)

So, the specification of the vehicle at that stage it would like:
 $M_1 + M_2 + \dots M_k + M_{k+1} + \dots M_1 + M_{1+1} + \dots M_n$ (modules) =
(Base + Core + Sales) specification.

At this stage of the vehicle Specification, experienced auditors apply *module rationalisation* which means that they run the **Product Definition** system in liaison with the most technically experienced engineers to generate the final "usage statement" of the vehicle. Notice, the usage statement of a Land Rover's product it will be quite different from what it has been said till now with reference to Ford and Chrysler. It refers to *Modules* rather *features combinations*, though the notation remains the same. ie. "+", "-" etc. The modules, however, implicitly refer to vehicle features. Even further, each module is connected to its total part requirements, every last nut, bolt and screw, which takes the vehicle from the Product Definition system direct into the COPICS Bill Of Materials.

The rationalisation of modules is done with the application of rules that determine the correct combination of modules. There exist two types of rules within Land Rover the *resolution* and *feasibility* rules.

A resolution rule, for example, may stand as :

IF module M1 appears into the final vehicle "usage statement"

(ie. left hand drive car, with automatic transmission ..

... which probably belongs to module family MF1)

and

module M2 appears into the final vehicle "usage statement"

(ie. right hand drive car, with automatic transmission

... which probably belongs to the same module family MF1)

THEN

choose module M2 (*the most specific*) or

IF (M1 + M2) THEN M45 (where M45 is a totally different module).

The feasibility rules code the "common sense" in the design of the vehicle such as that you can't have 8 or 7 doors in a vehicle, or 6 wheels.

3.2 SUMMARY ON THE CHAPTER

In this chapter the PSC of Land Rover was discussed. The major concern in this thesis is the enhancement of the way which **conditionally** specifies the vehicles in a company, as this is the case with Rover. Conditional specification means the specification of the parts by using feature combinations and

boolean algebra (see chapter 2).

3.2.1 Modules and conditional specification

The use of Modules by Land Rover to specify vehicles is apparently simpler than the conditional way of combining vehicle features. This is because a specific Module includes a specific combination of features rather than a huge number of all the possible combinations of those features. For instance, the EN003 Module, which was discussed earlier, uniquely identifies a part for the vehicle:

a 4 cylinder petrol engine with
an oil cooler and
evaporated loss control, for Gulf.

In the conditional way this would represent at least $2^3 = 8$ combinations of the three vehicle features which are implicitly expressed within the Module such as the type of the engine, the cooler and its control. If the type of the engine fuel is concerned as well, such as diesel or petrol, this would create $2^4 = 16$ possible combinations of the features mentioned in the Module instead of a single one. The multiple feature combinations naturally affect the assembly structures which have to be re-organised each time the combination of features in the top assembly changes. Finally, in a Module, the territory

restriction (ie. for Gulf) is included whereas in the conditional case, the specification of the vehicle becomes even more difficult if it is considered the multiple territory restrictions usually for each feature combination.

Unfortunately, although the Modules specification is simpler than the conditional way of specification, it can only be used efficiently for vehicles whose customer requirements are limited. For example, the use of Modules in Rover each one connected to its own specific assembly structure would create Module Families, each one consisting, of literally millions of Modules, each Module representing a possible feature combination. Land Rover uses the Modules specification because of the only limited number of options which are required to satisfy the part of the market in which it operates.

The other point which emerged from the study of the automotive literature is that the specification of the vehicle, regardless if it is conditional or based on Modules, is done manually.

Regardless of the different audit philosophies invented to alleviate the load of specification (ie. module rationalisation, or the traditional Boolean algebra way) the thinking process for the validation of the usage conditions of the parts of a vehicle is done from humans. Consequently, mistakes are inevitable to happen because of human nature and the problem of the specification of the vehicle is still open for investigation.

The next chapter is concerned in particular with the PSC in Rover and discusses typical tools used by the company and other concepts which constitute it. The idea is that by understanding the PSC better, parts of its application may be found which may be automated or supported by a computer system and consequently the human error could be reduced.

The following two sections provide a gradual introduction to the basic concepts which are needed in the conceptual design of the Rover's Product Specification Concept and the tools which Rover uses for its application.

4.1 CONCEPTS

4.1.1 *Vehicle Part Grouping (V.P.G) (funtionality)*

An automobile company, in order to control the thousands of parts which assembly each of its products, needs a way of discriminating their function and fitment into the Vehicle. Rover uses the Vehicle Part Grouping "logic"; a standard grouping of parts by their function. At the highest level of Vehicle grouping Rover uses the *V.P.G.-groups* such as:

<i>DESCRIPTION</i>	<i>CODE</i>
Complete Vehicle	10
Body System	11
Frame and Mounting System	12
.....
Exhaust System	19
Fuel System	20
Steering System	21
.....

Each V.P.G-group can be then broken down into *V.P.G.-sub-groups*. For example, the V.P.G.-group "Steering System" is broken to the VPG-sub groups:

DESCRIPTION	CODE
Vehicle Gears	2101
Steering Column and Wheel	2102
Power steering pump and fluid	2103
.....

At the next level down, *V.P.G.-Sub-Sub groups* can be defined for each V.P.G. Sub-group. ie.

DESCRIPTION	CODE
Column, Shafts, Coupling & Tube	2102AA
Shrouds & Locking Mechanism	2102BA
Steering Wheel	2102CA
.....

A VPG-Sub-Sub group can then be broken to different assembly "*families*" with unique functionality in the vehicle area that the group represents. So, the "Column, Shafts, Coupling & Tube" group includes the assembly areas in Manufacturing with unique codes (QTB, QTC, QTD, QTG, ... QTN, QTP). The discrimination of the Vehicle parts can continue even more with the actual 'meaning' of the part itself within each assembly family. For example, as shown below an assembly family within the "Column, Shafts, Coupling & Tube" *V.P.G.-Sub-Sub-Group* can include more

than one different **Part Descriptions**:

DESCRIPTION:	CODE:
Pad assy Cruise Control	QTC
Pad assy non Cruise	QTC
Pad assy strg wheel	QTC
Support moulding assy Cruise control	QTD
Support moulding assy Non Cruise	QTD
Badge Steering wheel	QTG
Badge & carr assy countryman L/E	QTG

A part description, ie. 'Badge Steering wheel', as the term indicates, represents only the higher level of description of probably a group of different physical Vehicle parts with significant resemblance. ie. more than one types of 'Badge Steering wheels' may exist, similarly there are many different types of screws under the Part Description "Screw-flanged head". Hence, more than one **Part Number** may be allocated under the same Part Description to identify uniquely a physical part.

At the last stage of the Vehicle Part Grouping, physical parts (part numbers) are linked logically to various **fitment** conditions, which means that a single part may be applied in different ways on the vehicle. For example, for the part description "Screw-flanged head" more than one fitment is available, and applies accordingly to the way the part is specified in its usage statement.

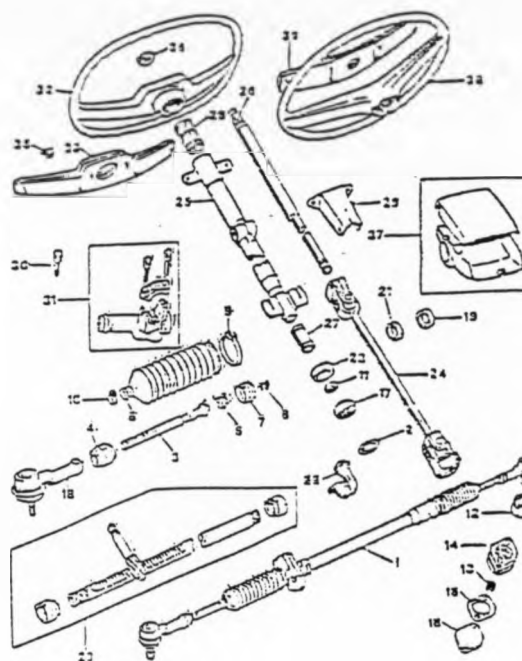


Figure 16: VPG group level - Sub Groups
STEERING COLUMN & WHEEL.

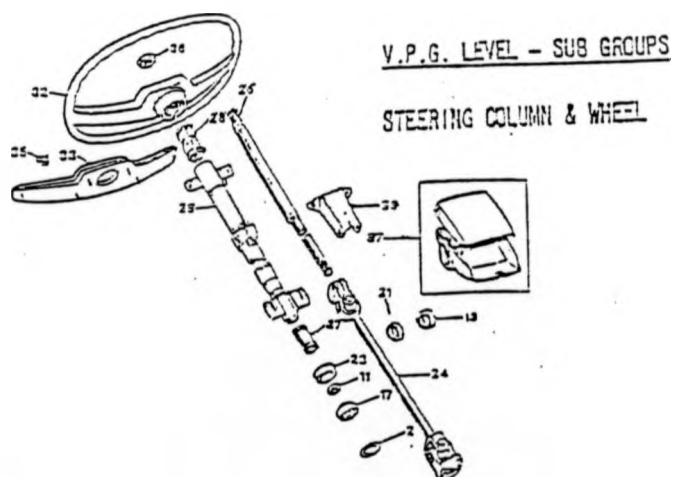
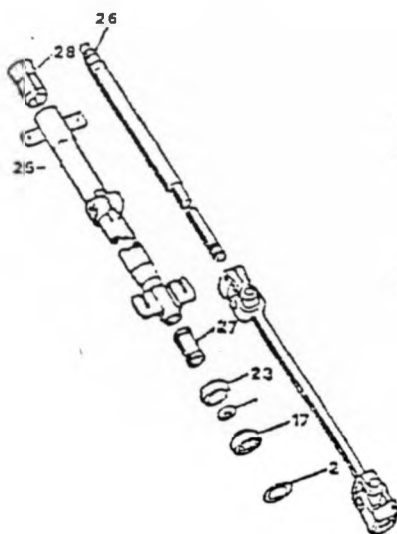
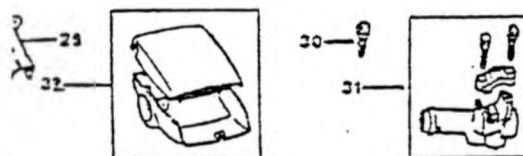


Figure 17: Sub-VPG level - STEERING COLUMN.

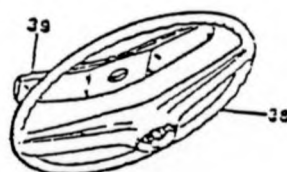


V.P.G. LEVEL - SUB SUB GROUPS

COLUMN, SHAFTS, COUPLING & TUBE



SHROUDS & LOCKING MECHANISM



STEERING WHEEL

Figure 18: Sub-Sub-VPG level, from reference [23].

<i>Part Description</i>	<i>Physical Parts - Part Numbers</i>	<i>Fitments</i>
-------------------------	--	-----------------

"Screw-flang head"	JTR123419xxx	- door handle fitment
		- boot assembly
		- interior lighting

Diagrams 16, 17 and 18 show a subassembly explosion of the steering columns VPG of the Rover vehicle which have been discussed.

4.1.2 *Parts TO Features TO Model concept (functionality)*

4.1.2.1 *Features to Model concept*

As mentioned earlier a vehicle is a combination of many different multilevel assemblies. eg. engine assemblies, glass assemblies, wiring, seat assemblies (plastic, leather, etc), vehicle carpets, electrics, etc. In that sense, it seemed logical for any automobile company to orientate the breakdown of its product in the assemblies direction. Vehicle Part Grouping as discussed earlier represents the methodology of this idea. Furthermore, such an analysis of the vehicle to different parts by their functionality has enabled Rover to refer to vehicle characteristics such as adjustable seat, central door locking, electric mirrors, manual gearbox etc. Those are the features of the vehicle which were referred during the introduction of the

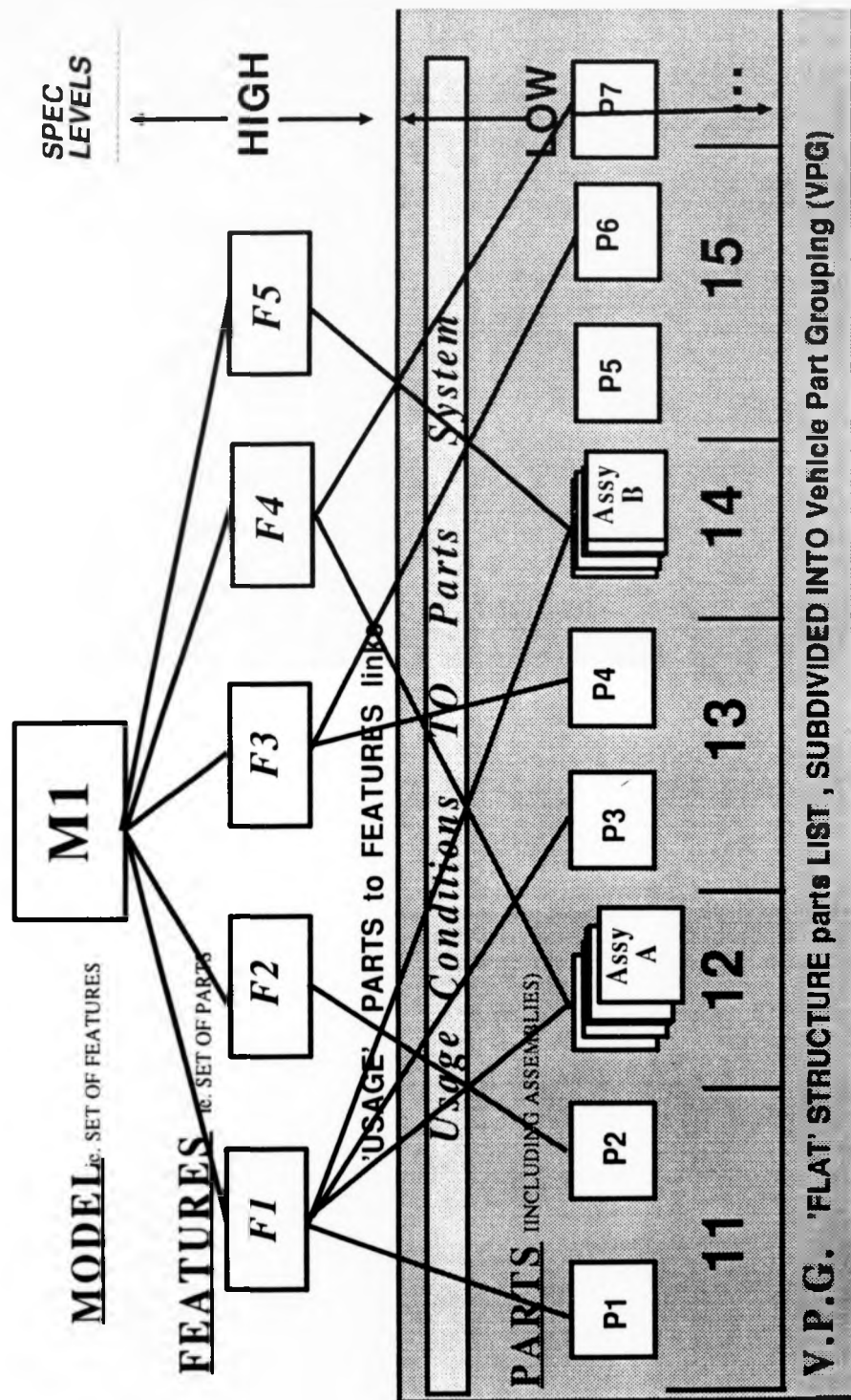


Figure 19: Parts to Features to Model Concept.

Product Specification Concept. Features as used within Rover represent different **Sets of Parts** going for manufacturing and subassembly and then on to the vehicle and top assembly. For example, at figure 19, the use of feature F1 in the model M1 will indicate to manufacturing that model M1 'gets' the parts P1, P3, and the assemblies ASSY-A and ASSY-B.

4.1.2.2 **Usage Conditions TO Parts Concept** (physicality)

The Sets-of-Parts as they are carried over from the feature statements, are initially meaningless to manufacturing and a transposition from 'functionality' to 'physicality' is required. ie. the "compilation" of all the parts usage statements to parts fitments.

In an environment like manufacturing which everything must be unambiguous and accurate, it is the fitment of each part which drives the assembly structures of the final production line, rather than the part itself.

4.1.2.3 **Assembly Structure Concept** (physicality)

When all the features of a model variant have been defined and the usage statements of all the parts have been specified against those features, Manufacturing can thoroughly document the fitment of each part and consequently generate the Bill Of Materials (B.O.M) for the specific vehicle (figure 20).

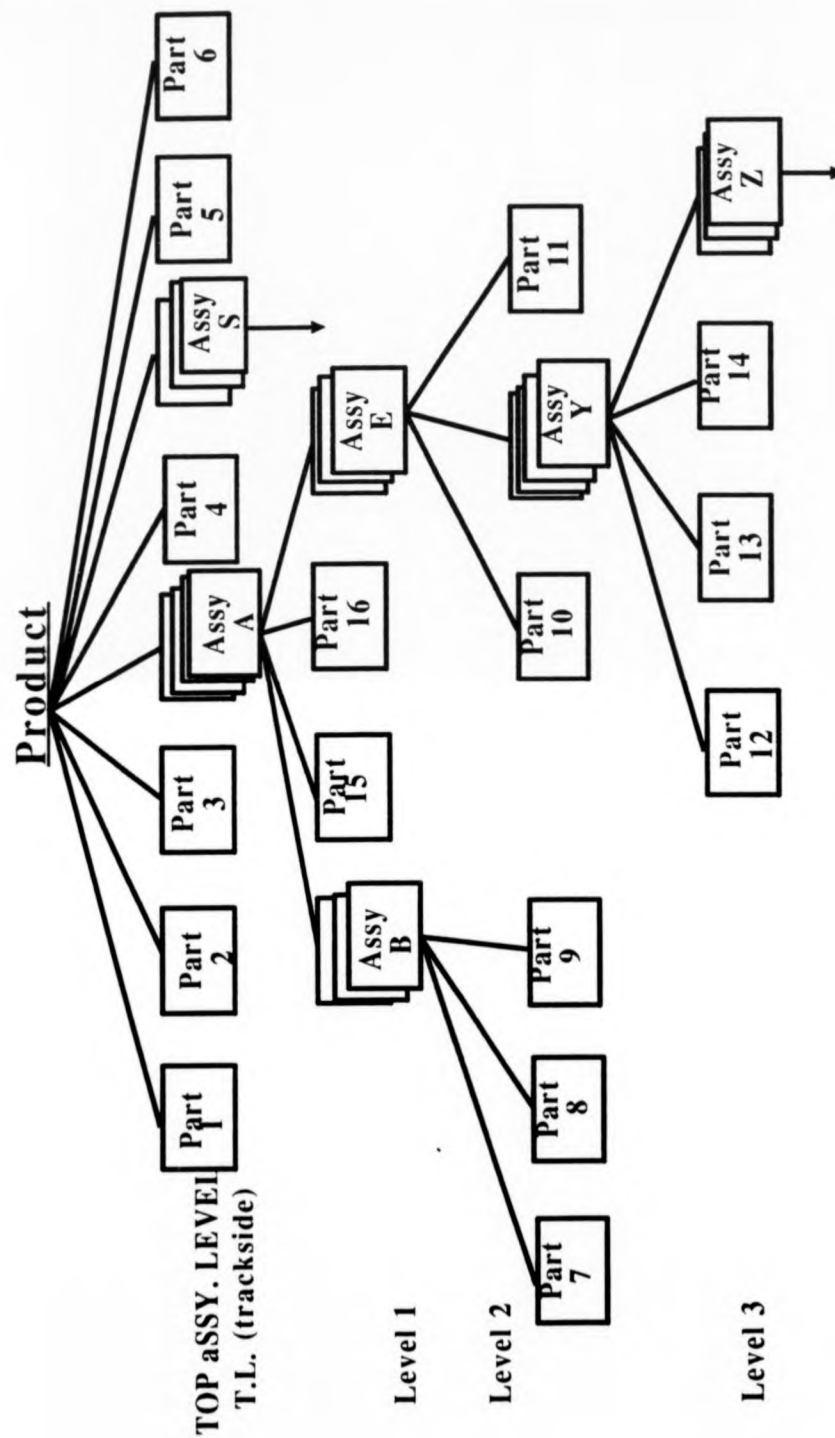


Figure 20: The assembly structure concept.
This is a hierarchical structure B.O.M (ie. using levels within Part Numbers)

In summary, the combination of the above 3 basic concepts provides the theoretical background for the design of the Product Specification Concept within Rover. Figure 21 shows how all these concepts are linked together for its implementation.

4.2 APPLICATION TOOLS

In this section the different tools - electronic or manual - that Rover uses to either implement individually or link together each of those concepts is introduced.

4.2.1 **Product Information Management System**

The first thoughts for the physical development of a Product Specification System were focused on the prime requirement that the system should be capable of handling huge amounts of data and both unambiguously and flexibly mix and match the data with the newcoming feature combinations. It also should allow for extension. In order to get the maximum benefits of such a system Rover introduced in 1981 a computerised specification system called the PRODUCT INFORMATION MANAGEMENT SYSTEM (PIMS) to replace the manual Engineering Release System which had been in operation since 1972. PIMS was initially designed for British Leyland cars. It was the first attempt by Rover to support electronically its Specification system and it used the database technology available then, namely the hierarchical structure.

Hence, PIMS is a complex Mainframe-based system of multilevel hierarchical Information Management System (IMS) databases both stand alone and physically linked. Although PIMS is generally referred to as a single database, at least five major databases can be categorized and several smaller subsidiary ones, as well. Database programs run exercises for the control of these data in the form of Rover's needs, eg. a program may use the PIMS databases to generate the Automatic Part Numbering (A.P.N.) of a new part. The 5 major databases carry information relevant to Parts, Model ranges, Classifications, Sources and Control data. The reader for further information can refer to [17].

Although PIMS is the prime author for the specification of a part, several other documents are created and circulated inside the company to support the general operation for the specification of the product and are discussed in the following.

These are:

- Features List
- Base and Additional Features Charts
 - Base Features Summary Chart
 - Model Summary Chart
 - Additional Features Chart
- Territory Group Index Report
- Territory Code Index Report
- Base Features Code Index Report
- Additional Features Group Index Report
- K87 or Company Features Directory
- Automatic Part Numbering (APN)

4.2.2 **Features List** (fig 27, 28)

The Features list is issued by the Vehicle Directorate department and appears at the very first phase of the life of the new product. It defines in an English-like language the form the new product should take. It actually lists the various derivatives - different versions - of the product that will be produced in terms of their trim levels, badging, body type and engine capacity. The availability of "features" that the product may have is presented as a *matrix* map of X's (standard availability), O's (optional availability) and empty spaces (not available) (appendix 1).

4.2.3 **Base and Additional Feature Charts.** (fig 37, 38 and 31, 32, 33)

The Specification Services department uses the Features List documentation to compile the Base and Additional Features Charts (BAFC). The BAFC provide the availabilities of the vehicle features in Boolean code which Manufacturing should receive from Design Engineering.

More specifically, BAFC list all the *Features Groups* which experience and practice within Specification Services have invented to tackle the areas of the car which show a tendency for regular multioptional availability, hence maximum complexity in

their specification. Within a Feature Group the availability of each feature or vehicle characteristic is more specifically defined according to Combinations, Territory and Base features restrictions issued from the Vehicle Directorate.

Combinations restrictions establish the availability of a vehicle feature in combination with other features of the product whereas territory restrictions control the availability of each feature to the various territorial options such as Europe, UK only, etc.

For example, the Base and Additional Feature Charts of Metro include all feature families that made up this model; bumpers, floor coverings, radio speakers, exterior mirrors finish etc. The Feature Group description "Exterior Mirrors Finish" has Feature Group code "B12", and more specifically for this model includes the features:

DESCRIPTION	CODE
"BLACK Mirror - standard head"	"B12A"
"BODY COLOUR MIRROR - standard head"	"B12C"
"BLACK Mirror - large head"	"B12D"
"NIMBUS mirror - large head"	"B12G".

Within the BAFC the restrictions of each feature is defined with the same Boolean notation, discussed earlier ie. (+), (-), etc.

The BAFC consists of three documents:

(i) Base Features Summary Chart (fig. 37)

(ii) Model Summary Chart (fig. 38)

These two documents keep similar vehicle information in different format; the Base features of the particular model, and are mentioned in more detail in chapter 5.

(iii) Additional Features Chart (fig. 31, 32, 33)

This is the largest and most important document in AFC. It lists all the Additional features offerings from Rover for a particular model, restricted to various territories and design combinations.

The BAFC though are documents with high discipline, the presented vehicle information using manufacturing codes and boolean statements make it unfriendly for design engineers.

4.2.4 Territory Group Index Report Territory Code Index Report (fig. 36, 35)

The first of these two documents groups territories in the following sequence: *Supergroups*, *Groups* and *Countries* and the latter works the other way around, from the countries to supergroups. eg. the *supergroup* with description "HOME GROUPS" and code

"M20*" includes the groups :

"U.K. EXPORT GROUP" (M20E) including the countries:

Cyprus, Germany-RHD, Malta, New Zealand, and

"UNITED KINGDOM GROUP" (M20E) with countries :

Irish Republic and United Kingdom.

4.2.5 Base Features Code Index Report (fig. 34)

This translates the Base Features coding system into English.

4.2.6 Additional Features Group Index Report (fig. 29, 30)

This document lists the availability of all Feature Groups in relation to all Rover vehicles.

4.2.7 K87 or Company Feature Directory

It records all the Base and Additional features available for any model, from the time that BAFC started to be applied within Rover.

4.2.8 Automatic Part Numbering (A.P.N.) (see fig 28, 54)

A.P.N. has been developed in response to a requirement outlined by Rover who wished to add significance into the part number and have the facility of on-line creation and recording of the parts. The Part Description Catalogue contains all the descriptions of

the parts which have been used by Rover until now and represents the higher level for the search of a part and assign to its part number.

4.2.9 Usage Conditions to Parts System

This Usage Conditions to Parts system was developed by ISTEEL, as was PIMS, and uses the information from PIMS - ie. the usage statements of all parts of the new vehicle - to calculate the quantity of individual components required on the production line. As mentioned earlier, features in reality represent sets of parts to be assembled. Thus, the Usage Conditions to Parts system can broadcast the quantity of parts required by 'pulling out' from the data base parts whose usage condition matches the order specification. This is discussed in more detail in chapter 9.

In summary, the first section of this chapter analysed the subsidiary concepts of the Rover's PSC. Typically, these represent the common ideas which have modelled the PSC in automobile through the years. The top-down specification of the vehicle, from features to the actual physical parts, forced automobile to create tools which would support such a process at each level.

The Features List and the BAFC are among the major tools in the specification of a vehicle. They respectively document and code the parts of a vehicle in the form of usage statements. They are typical examples of tools which are used when the vehicle is

specified conditionally.

PIMS is another important tool for the specification of the vehicle as it keeps all the usage statements of all the parts of the vehicles. Notice, however, such databases of usage statements accumulate information rather than elaborate it. APN which is a subsidiary database of PIMS and hierarchically groups the parts of the vehicle according to the Vehicle Part Grouping concept is another major tool for the specification of the vehicle and it will be used later.

The Usage Conditions to Parts system, concerns mainly manufacturing and it not used in the specification of the parts. However, the knowledge of the way its principle works to collect the parts usages related to a company order can help for higher quality in the specification of the parts.

The rest of this chapter, shows the Rover's PSC with only brief indications to the tools it uses.

4.3 THE ROVER'S PRODUCT SPECIFICATION CONCEPT

4.3.1 a 'high level' simplified approach

The Rover's release system which issues the information to support the Product Specification Concept is shown simplified in

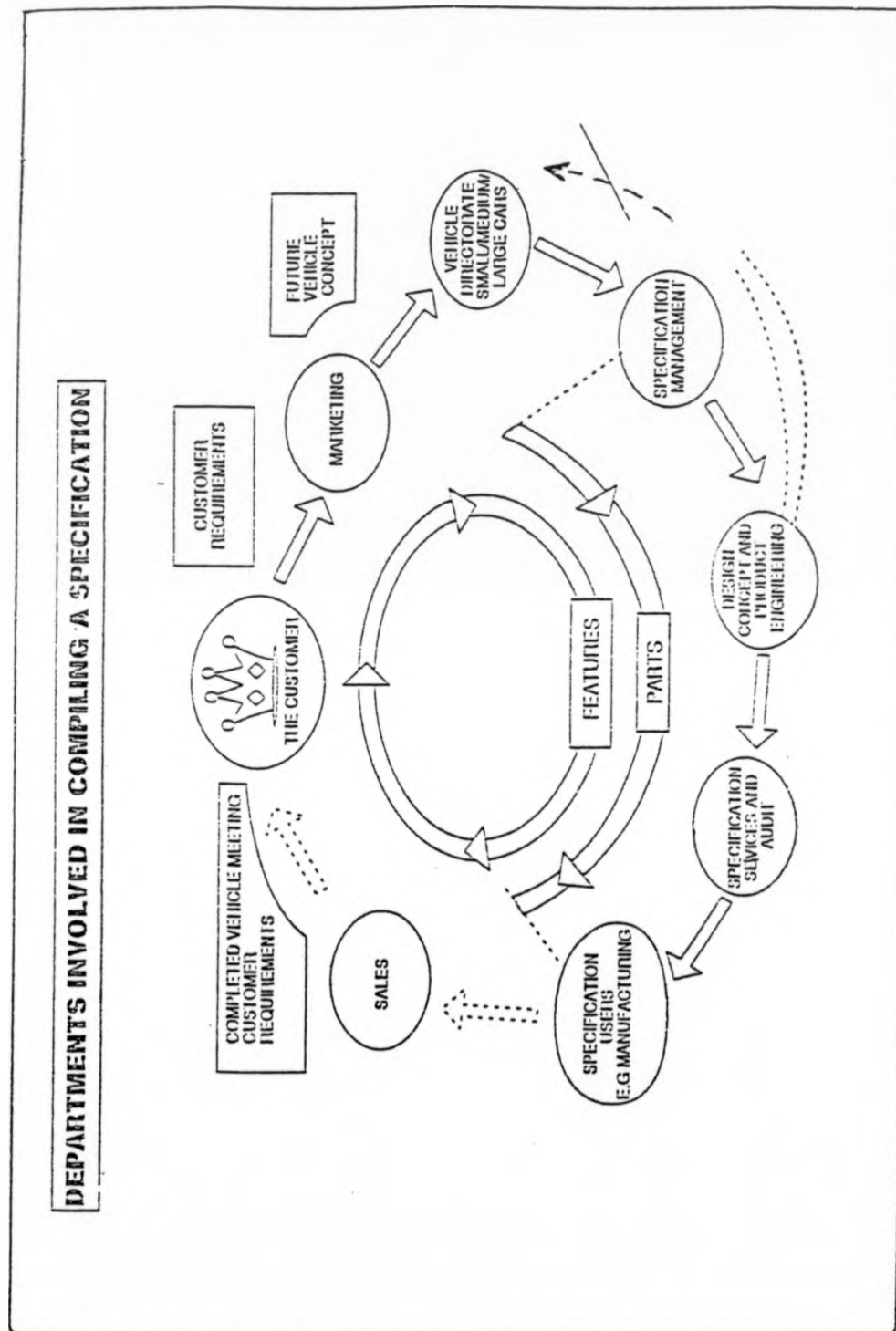


Figure 22a: A simplistic view of the PSC (and release system) in Rover.

figure 22a. In reality it represents a very complicated procedure with nested subtasks. Further details can be found in [19].

The preparation of any specification starts with the Customer. The customer's requirements are established by the Marketing department via extensive market research. Marketing must then relay these to the Vehicle Directorate/Line so that they can be translated into a specific vehicle strategy which accounts for the needs of the customer. Vehicle Directorate compile a detailed proposal of the actions required to complete the project which is then issued as the Product Development Letter (PDL). This authorises work and development to be carried out for the new product and supports it by compiling the Features List of the vehicle.

Features Lists outline in "feature" terms the make up of the total vehicle. Specification Management/Specification Services then use the Features List to compile the Base and Additional Feature Charts. The Base and Additional Feature Charts form the high level of the specification. Together with the next assembly relationships (the next higher level in the assembly structure), they represent the two basic types of information which combine to form the complete product specification. The Base and Additional Feature Charts are then used by the Design, Concept and Product Engineering areas to develop the parts that will comprise the model variants.

At present, Specification and/or Audit services receive documents from Product Engineering at any form ie. in boolean expressions or english statements and either check the validation

of each part's usage statement or work out those statements for themselves. This operation is performed manually and auditors liaise with Component Engineers in the case of queries or raise questions both verbally and using Release Query Notes.

Finally, the audited engineering packages (features & parts) are loaded to PIMS from where they can be used by other departments of the company that needs them such as Inventory, Purchasing, Manufacturing etc.

4.3.2 the Rover's PSC/release system as related to build phases.

Figure 22b depicts (still simplified) a closer look to the processes happen during each build phase of the vehicle.

The fundamental of the procedure is to start the programme by establishing on PIMS a complete product specification (ie. all vehicle/unit derivatives intended for final production) during the concept phase (D01). Then at each build phase, first validate the complete PIMS specification, and from that, derive electronically the specification of the products chosen to be built in that phase. The 'build specification' is then sealed and transferred to Application System (AS), the IBM's database handler, and subsequently kept up to date for the duration of the build phase. AS holds the Master Part Status (parts to be manufactured or bought from outside), BAFC, timing information (lead times for manufacturing and purchasing of parts), long lead

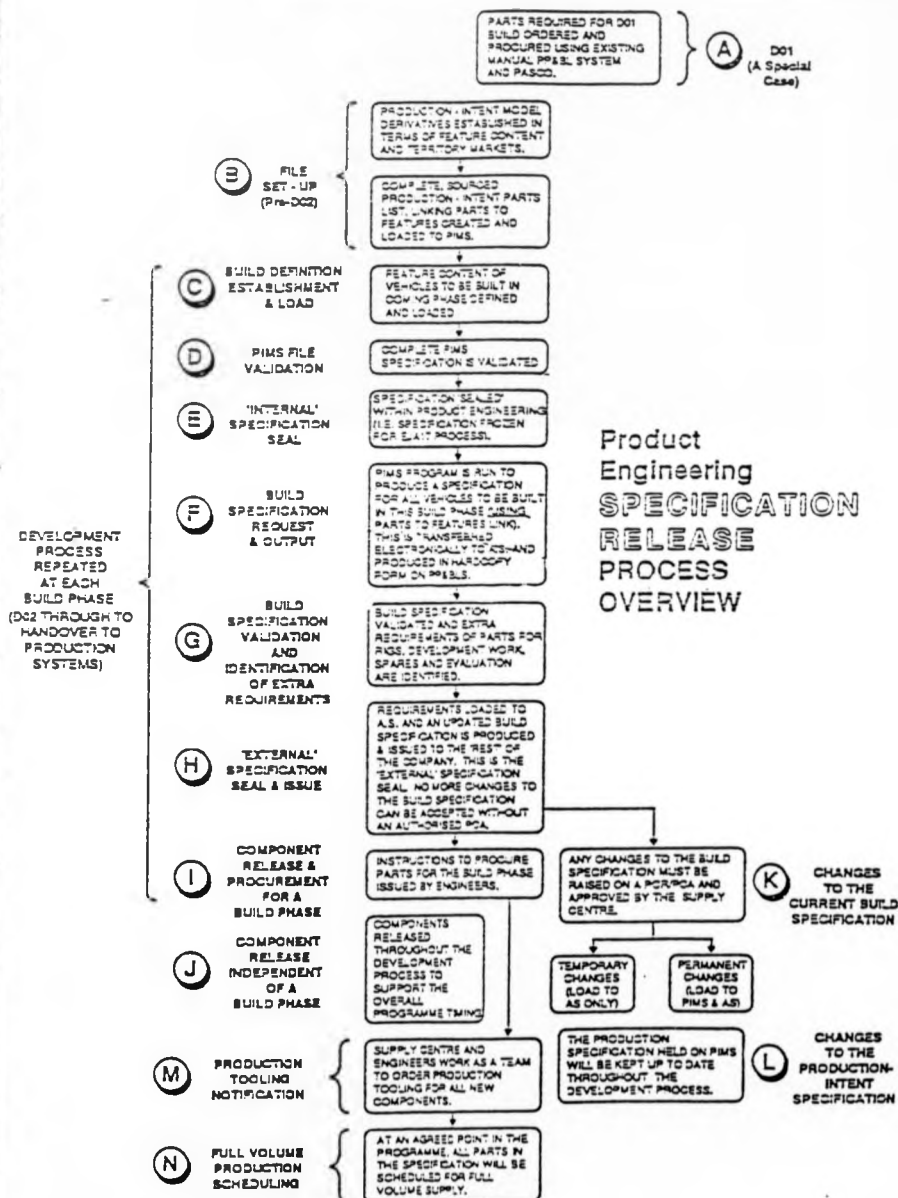


Figure 22b: A closer look at Rover's PSC circulated around the company.

items etc. In reality there is more information in AS than in PIMS.

This process is repeated at each build phase starting with the re-validation of the complete PIMS specification. The timing of each event will be detailed on the appropriate Timing Plan.

Thus, throughout the whole development process, PIMS is the prime source of data, but the unique build specifications, although initially derived from PIMS, are held separately on AS.

Note that during the PSC development process there are, in effect, two 'specifications':

- 1) The complete product-intent specification, held on PIMS, and
- 2) The unique build phase specifications, held on AS,

and throughout the process they must BOTH be kept up to date and consistent with each other [19].

4.3.3 PROBLEMS OF THE ROVER'S PSC / RELEASE SYSTEM

The first main problem that came to light to prevent the correct application of the Product Specification Concept is the **re-formatting** of the vehicle information through the route of its release within the various departments of the company.

It was mentioned that the Product Specification Concept and more specifically its features semantics were developed on the idea that people within the company could communicate with each other

regardless of their own specialisation. However, things are not as simple as they seem in theory. Contradictory motives arise with the actual application of the concept in the company. Because of the different needs of the various departments within Rover, the features of a vehicle must both be defined unambiguously to understate the risk of false specification but also ambiguously concerning flexibility in the market to the legal or technological changes. This comes as result of the apparent confusion in Rover of what people really mean by the word **feature**.

Vehicle Directorate, for example, implements the Features List having its main input from Marketing. Marketing as a cost analysis oriented department details the slightest modification in the vehicle that affects cost as a new "feature" of the product. Thus, a major number of such "features" are compiled in the Features List of the vehicle. The following example clarifies the situation.

The Features List of the Rover R8 model includes the "feature" 'BOOT LID FITTINGS' which is detailed further to other "features" such as

- key operated boot lock
- Plastic sleeves over boot latch
- Rear spoiler, black
- Moulded plastic boot lock cover etc.

For Vehicle Directorate/Line, the above are considered to be "features" of the product as the responsibility of this

department is to point out all the project details of cost feedback.

On the other hand, such boot details do not mean anything as far as the Base and Additional Feature Charts document of the vehicle is concerned. The BAFC describes only the major features. That is, features for which experience within Specification Services has revealed that they represent the main cause of changes in the design of new derivatives or new models. These are the Base and Additional features which were discussed earlier during the introduction of the Product Specification Concept. In the the Base and Additional Feature Charts for the model boot details are not referred to at all but are linked implicitly through the BAFC's feature 'REAR BADGE'.

For this reason, the features of a vehicle which are defined in the Features List document are marked by upper quotes. This convention is used to indicate that such "features" do not represent the features concept in this research.

The terminology "Features List" used by Vehicle Directorate for the document that supports the development of the new product is misleading as there are not only features inside this document but also parts. Specification Services translates the Features List into the logic of the Base and Additional Feature Charts.

The problems of the Rover's Product Specification Concept start with this "change of gears" in the features ideology.

4.3.4 DIFFERENCES BETWEEN THE TWO DOCUMENTS

Firstly, the Features List is written in an english like language whereas the Base and Additional Feature Charts code unambiguously features with their restrictions.

Secondly, the Features List and Base and Additional Feature Charts implement similar information but from totally different angles. Not only in the way that the features meaning differs but also in the design of the information itself within the documents.

The Features List specifies the vehicle - features and restrictions - on the basis of the UK market vehicle specification (main document) with the rest of the world derivatives being variations on this. The reference specification point is geographical e.g. the UK market [9].

On the other hand, the Base and Additional Feature Charts are oriented to the characteristics of the vehicle which control its design update. The specification reference point is the vehicle feature and all the territorial information is grouped around it.

The Base and Additional Feature Charts of the model, once they are compiled from Specification Services, never go back to Vehicle Directorate for in line re-specification with the Features List although they are released to various departments

within the company (figure 22a).

4.3.5 WHY SUCH A DIFFERENCE IMPACTS ON THE AUDITING FUNCTION
AND FURTHER TO THE PSC IN THE COMPANY

The problem this gives rise to is that the re-formatting of vehicle information during its release to the various departments of the company creates confusion in the Product Specification Concept. That is because engineers, having the choice of the Features List document, they prefer to work with it, as it is written in English. However, they must specify their design intents on the basis of the BAFC's coding system. The meaning of the vehicle feature becomes vague and auditors have to work with both documents.

Furthermore, as the two documents differ so much in their compilation, inconsistencies appear with regard to the same information existing in both documents resulting from compilation error in Specification Services. Auditors have to check for their in-line compatibility, as well.

In summary, a closer look to the Rover's PSC, showed that the specification of a vehicle does not happen in one circulation of the relevant information through the various departments. Instead, information for the new product is gathered gradually,

fed-back from the various departments, quality tests and market situations. This process can last even one year or more and usually implies more than 11 circulations of the vehicle information (build phases). At each build phase the old information is updated in PIMS and AS having previously validated from the Auditing department.

In addition, studying further some of its tools, it became apparent that the re-formatting of the original vehicle specification is a major cause of human error. The PSC loses its original clarity because of the created features definition confusion as it expressed in the various documents which are compiled from different angles. Such a confusion impacts upon the various departments of the company and especially Auditing. From the literature survey it was found that such a phenomenon is common among all the automobile companies which justifies why up till now the Specification Services and/or Auditing have been highly dependent on 'common sense' and 'experience' and not yet automated.

Having, discussed the Rover's PSC the following chapter continues further the investigation, more specifically, with the Rover's Audit function itself. For this reason, the following chapter introduces the way which auditors use to validate the documents coming from Component Engineering manually.

5. The auditing function

The need for the existence of the auditing department within Rover arises from the inability of the engineers to understand and work with the complexity of the Product Specification Concept. That is, though they are well trained in component design, they are concerned only with their areas of specialisation rather with the complexity of the vehicle in a whole. In addition, they find the coding systems most unfriendly to work with.

Training of these engineers in the techniques used by Auditing services would not be a solution as it would require at least 3 months of course attendance and up to one year working with the system on a daily basis, also, because of the need to keep design lead time down (see discussion associated with table 5). Designers need to concentrate on their basic function rather learn other skills.

For this, the auditing function exists independently as a separate department within the company; it is applied exhaustively within Rover, in all vehicles, in all phases of their design, at any new changes in their specification, throughout their life.

More specifically, the job of the Auditing department is to validate the specification packages - as they are called - compiled by Component Engineering. Figures 23, 24 represent two

pages (or physical parts) of a specification package which refer to two different parts which are linked in assembly.

At the top right corner of figure 23 which represents the "SEAT-FRONT COMPLETE manual", the 8 digits code number under the heading "PART NO." represents the part number of the physical part which uniquely identifies it. The "USAGE CONDITION" section describes the usage statements of the part and the number under the heading "USAGE QUANTITY" shows the quantity of such physical parts needed in one car. Two more pieces of information are of great importance; the "NEXT ASSY NO." which indicates the higher level assembly (part number) in which this part is going to be assembled and "QUANTITY" which records the number of such parts needed at each assembly. Figure 24 which represents the "COVER-ASSY-SEAT-FRONT-SQUAB" shows that one such a part is fitted to the "SEAT-FRONT COMPLETE manual" assembly.

A typical specification package would usually consist of about 30 of these pages each representing a physical part linked together by engineering design and/or manufacturing rules. Often the drawings of the parts accompany the specification package and show the relationships between these parts.

In actual fact, specification packages arrive in the Auditing department in various forms, in addition to the official one (figures 23, 24). This is because many engineers cannot understand or work with the auditing coding system and express their design in ways they find more suitable. Some usual cases are the submission of documents in a eg. matrix form or even the

68

Figure 23: The specification document for the physical part HAD1000€

description of the characteristics of the design of the part in English. Rover tends to accept such alternatives in order to save design time even though it is likely to increase the possibility of mistakes in specification. This, then, creates even more pressures on the Auditing department as now the auditors not only have to validate the usage statements coming from Component Engineering but also to create them from scratch, assuming they fully understand from the ambiguous English statements the reasons for which the component was designed.

In the next section a typical audit exercise is discussed in order to clarify the tools that the audit function uses and also to provide a basis of the understanding of the design of a computerised system which may automate it.

Notice, the existing manual system has grown over many years in a pragmatic manner, extra procedures being invented as required by individual auditors. The analysis which was essential for the development of an automated system had to lead to a reconfiguration of the system in a structural manner.

Contribution to knowledge:

- the identification and sequencing of three phases in the Auditing function
- the identification of areas in which computer algorithms could possibly be implemented.

5.1 An audit example.

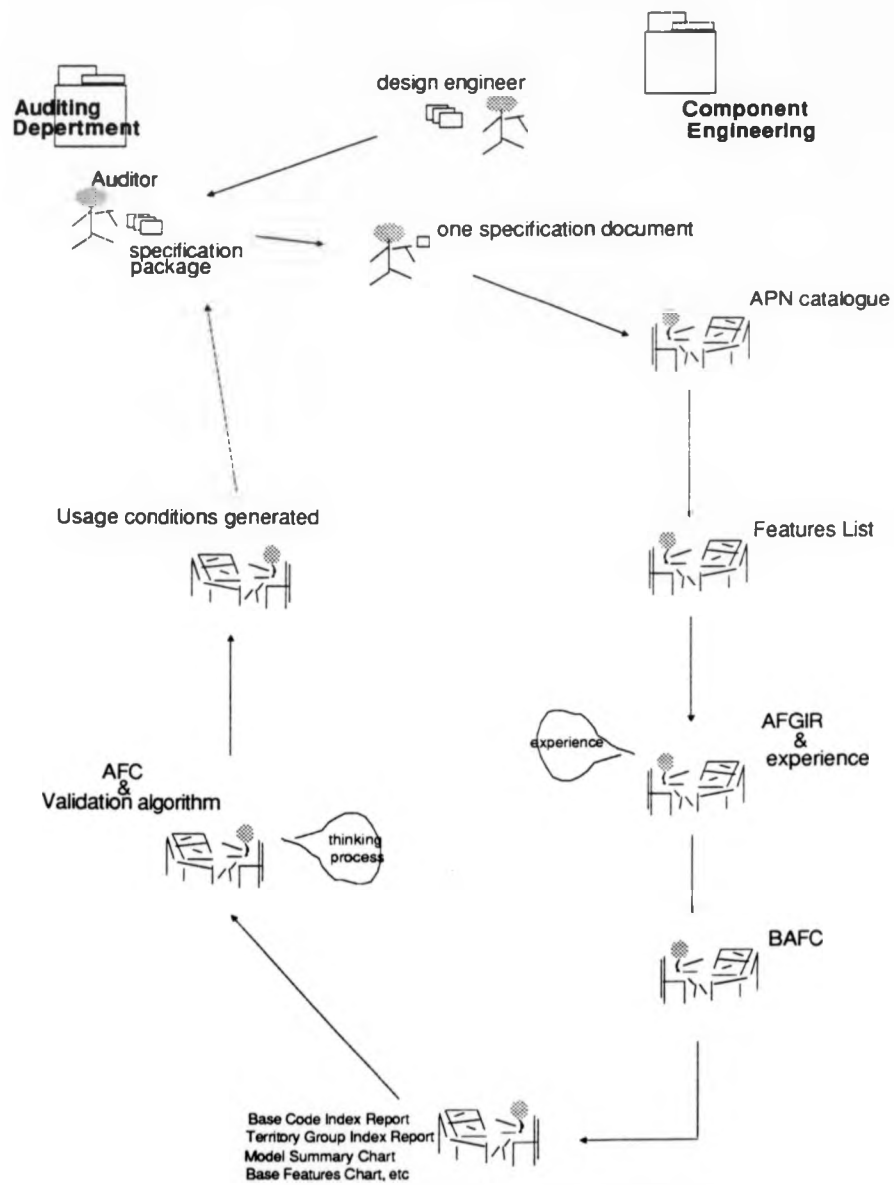


Figure 25: The first phase of the Audit process

5.1.1 PHASE 1

The auditor uses all the application tools discussed in the previous chapter in order to validate or create the usage statements of the parts of the vehicle (Features List, BAFC, K87 etc.).

The process that the auditor follows in the first phase of the audit function is shown in figure 25.

The Features List represents the "bible" - using the auditors' terminology - for the auditing process. It is the first document relating to the product compiled by the company and describes the vehicle in english. Figures 26, 27 depict two consecutive pages from the Features List of the Rover R8 model and its counterpart, the Honda Concerto.

In them, the side-headings at the top of the figure "CLASS" and "TR. LEVEL" represent the various derivatives of the Rover R8 which were offered by Rover until the middle of 1989. The "DESCRIPTION" section details the characteristics of the vehicle, with their availability indicated with Xs (standard fitment) or Os (optional fitment) across the row of derivatives. Finally, the "REMARKS AND EFFECTIVITY" section on the right hand side of the figure comments upon such characteristics and shows the human orientation of the document.

In the example which follows the auditor is required to validate (or create) the usage statements of the front seats of the Rover

R8 model. In reality, the description of the part which is going to be audited is chosen from the Automating Part Numbering Catalogue (APN) and it will be very specific; for example, "SEAT-FRONT COMPLETE manual" (figure 28).

In this case, the information relevant to this seat in the Features List of the Rover R8, exists under the sub-heading *front seats*. Thus the auditor has his first view of the characteristics ("features") of the Rover's R8 seats which helps him to identify the possible new additional features which have been offered by the company for the product in the past and which may condition the usage of the part at a later stage. The auditor, also, knows from experience some other features which a seat part of the vehicle usually uses in its specification such as the material from which it is made (leather, plastic seat), the style of its operation (manual, automatic) etc. Combining both of these inputs he can select only those features - actually feature groups - which are closely related to the application.

Not all of them, however, may be available to the specific model. In this case, the Additional Feature Group Index Report (AFGIR) will help the auditor to identify only those feature groups applicable to the specific model. For example, for the lowest specification car, the Rover Mini, it would not be expected that luxurious options be offered on its seats (such as heated, or remote control adjustable seats).

Figures 29, 30 represent two pages from the Additional Features Group Index Report. This document maps in a matrix form the availability of all the additional feature groups which have

Figure 26: The Features List section for the seats of the Rover R8 model.

[illegible]

Figure 27: The Features List section for the seats of the Rover R8 model, continued.

PART NUMBER LAP FAC	PRIMARY DESCRIPTION	SECONDARY DESCRIPTION	IN STOCK IN SP
VPG 1109AA FRONT LIGHTING (FAC LIGHTING IN 2115)			
110	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
111	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
112	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
113	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
114	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
115	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
116	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
117	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
118	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
119	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
120	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
VPG 1109AA INTERIOR LIGHTING (FAC LIGHTING IN 2115)			
121	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
122	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
123	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
124	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
125	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
126	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
127	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
128	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
129	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
130	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
VPG 1109AA SEAT COMPLETE (FAC LIGHTING IN 2115)			
131	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
132	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
133	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
134	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
135	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
136	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
137	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
138	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
139	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
140	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
VPG 1109AA SEAT COMPLETE (FAC LIGHTING IN 2115)			
141	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
142	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
143	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
144	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
145	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
146	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
147	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
148	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
149	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA
150	FRONT LIGHTING LAMP	FRONT LIGHT	AMTCA

Figure 28: The 1109AA VPG from the APN catalogue which includes "seat-front complete manual".

been compiled by the Specification Services, against all the current Rover models (the code XW represents the Rover R8 model, XF the Metro, XS the Rover 800 etc.).

In the AFGIR it can be inferred that the feature groups which affect the front seats of any Rover model would be: "B38: SEAT BELTS FRONT", "B40 or B51: SEAT MATERIAL FINISH", "B44: SEATS - RECLINING", "B69: SEAT HEATING" etc. It might be assumed that the AFGIR would provide the information on which feature groups affect the front seats of any Rover model. However, this is only partly true as shown in figure 29. The "B38: SEAT BELTS FRONT" feature group is not available to the Rover R8 model, thus it cannot condition the usage statement of its front seats. On the other hand, the rest of the feature groups are available for the Rover R8 model and may condition them.

Up to this point, the auditor has clarified the area of the vehicle information needed for his exercise. The following feature groups are available for specifying the front seat of the car (R8):

"B51: SEAT MATERIAL FINISH" (the "B44" feature group does not account as it existed only in the past),

"B44: SEATS -RECLINING" and "B69: SEAT HEATING" feature groups.

Thus, he can access the details of this information for each feature in the feature group in particular, by using the hardcopy indexes (like "Table Of Contents") which are implemented within the Additional Features Charts document.

ILJ	INDIDA	DAI	LAD	UK	BLD	XE	HOHTFGO	XS	AUS	HOV	XX	UK	BLD
05	0117R10					XC	ME110 2	X1	AUS	HOV	XX	JAPAN	BLD
09	01000A	XX	JAPAN	BLD		XC			00				
100	INDIDA	UK	UK	BLD		200	HOVER 200	XV	AUS	HOV	XX	UK	BLD
100	S. D. 1.					200	AIL 16	XV	10005.1.	0011	-	SALLES	
200	000					XC	01111	X2	ADULT	PRINCESS	1901		
XC	DAE.5.110					XC	016						

Figure 29: A page from the AFGIR.

Figures 31, 32, 33 depict three typical pages each representing a feature group from the Additional Features Charts.

The second column in figure 31, for example, "FEATURE DESCRIPTION" describes in English the feature as it is compiled by Rover's Specification services in order to be offered as a new option to the new product. The third column codes the type of the feature availability ie. standard fitment (s), or optional (o), or a legal requirement (l) etc. The fourth column specifies the base features of the vehicle. The last column records the effectivity of the feature or Design Effect Point (DEP). That is the time the feature compiled first time from Rover (DEP IN) and the time which the Specification Services people withdrawn it (DEP OUT). The fifth, sixth and seventh columns are concerned with the combinations and territory restrictions of the feature.

The boolean algebra codes which appear in the Additional Features Charts represent Rover's notation to express restrictions on its products. For example, the group territory restriction "- M32S" in figure 32 means that the Rover R8 model can get an autolux leather seat, provided that the car is not going to the Spain group (M32S). In other words it is not going to the countries: Canary Isles, Gibraltar, Greece, Portugal and Spain. When more than one boolean expression is involved in the restriction of the feature, the specification area becomes the logical combination of these statements. For example, the top combination of the restrictions in figure 33 "+ M32B M32D M34A" and "M32B +DK" is interpreted as : "the front seat for the passenger of a Rover R8 model can be heated at the trim level 51 (X), for a left hand

drive car (B), if such a requirement is coming from the countries of the Belgium group (M32B), Austria group (M34A) but only from the Germany (DK) in the Germany group (B32D).

The Additional Features Charts therefore describe all the applicable options (features) of each feature group individually, together with the restrictions which control their availability. The auditors use three hardcopy files in order to interpret the codes of the Additional Feature Charts into English: the Base Feature Code Index Report, the Territory Code Index Report, and the Territory Code Index Report (figures 34, 35, 36). In addition to these, they use two more documents for quick references for the application they working on, rather than thoroughly search the complete Additional Features Chart of the product. These are the Base Feature Chart and the Model Summary Chart (figures 37, 38).

The Base Features Chart lists all the possible base feature combinations of the product with reference to their territory restrictions, whereas the Model Summary Chart expresses similar information - the base features of the product - from a different angle; it provides a synopsis of the TRANSMISION and DRIVE specifications with the rest of the combinations of the base features of the vehicle.

The existence of so many different documents corresponds to the the auditor's need to be able to handle vehicle information from many different perspectives, as no clear line of audit guidance exists. However, once the auditor has identified the information

Page	I	BASE FEATURE CODE	BASE FEATURE WORD	BASE FEATURE DESCRIPTION	INDEX	Q'PHH	OCT 89	ABBREVIATION FOR VPG GRID
		A30A	20SAL	2 DOOR SALOON				20SAL
		A30C	20C00P	2 DOOR COUPE				20C00
		A30E	10YVAM	2 DOOR VAN				1/VAM
		A30J	10YVAM	2 DOOR VAN				1/VAM
		A30K	10P00P	2 DOOR PICKUP				P/CUP
		A30L	40SAL	4 DOOR SALOON				40SAL
		A30M	40EST	4 DOOR STATE				40EST
		A30N	30SAL	3 DOOR SALOON				30SAL
		A30O	BUCKRD	BUCKBOARD				BUCKRD
		A30S	GPETLC	5 DOOR TUGHR				E/VAM
		A30W	50SAL	5 DOOR SALOON				50SAL
		A40A	MAH	MAH				MAH
		A40B	AUTOMATC	AUTOMATIC				AUTOMATC
		A50A	RIGHT HAND	RIGHT HAND				RIGHT HAND
		A50B	LEFT HAND	LEFT HAND				LEFT HAND
		A60F	IN LINE	IN LINE				IN LINE
		A60H	SPECL	SPECIAL				SPECL
		A60W	U.I.S	U.I.S				U.I.S
		A60Y	CITY	CITY				CITY
		A63A	LEVEL 49	TRIM LEVEL 49				LEVEL 49
		A63B	LEVEL 40	TRIM LEVEL 40				LEVEL 40
		A63C	LEVEL 1	TRIM LEVEL 1				LEVEL 1
		A63D	LEVEL 1	TRIM LEVEL 1				LEVEL 1
		A63E	LEVEL 59	TRIM LEVEL 59				LEVEL 59
		A63F	LEVEL 19	TRIM LEVEL 19				LEVEL 19
		A63H	LEVEL 2	TRIM LEVEL 2				LEVEL 2
		A63I	LEVEL 3	TRIM LEVEL 3				LEVEL 3
		A63J	LEVEL 3	TRIM LEVEL 3				LEVEL 3
		A63K	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63L	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63M	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63N	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63P	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63Q	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63R	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63S	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63T	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63U	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63V	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63W	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63X	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63Y	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A63Z	LEVEL 4	TRIM LEVEL 4				LEVEL 4
		A70A	850	850 (800)				850
		A70B	1000	1000 CC				1000
		A70C	1100	1100 CC				1100
		A70D	1275	1275 CC (1300)				1275
		A70E	1300	1300 CC				1300
		A70F	1400	1400 CC				1400
		A70G	1500	1500 CC				1500
		A70H	1600	1600 CC				1600
		A70I	1700	1700 CC				1700
		A70J	1800	1800 CC				1800
		A70K	1900	1900 CC				1900
		A70L	2000	2000 CC				2000
		A70M	2100	2100 CC				2100
		A70N	2200	2200 CC				2200
		A70O	2300	2300 CC				2300
		A70P	2400	2400 CC				2400
		A70Q	2500	2500 CC				2500
		A70R	2600	2600 CC				2600
		A70S	2700	2700 CC				2700
		A70T	2800	2800 CC				2800
		A70U	2900	2900 CC				2900
		A70V	3000	3000 CC				3000
		A70W	3100	3100 CC				3100
		A70X	3200	3200 CC				3200
		A70Y	3300	3300 CC				3300
		A70Z	3400	3400 CC				3400
		A70A	3500	3500 CC				3500
		A70B	3600	3600 CC				3600
		A70C	3700	3700 CC				3700
		A70D	3800	3800 CC				3800
		A70E	3900	3900 CC				3900
		A70F	4000	4000 CC				4000
		A70G	4100	4100 CC				4100
		A70H	4200	4200 CC				4200
		A70I	4300	4300 CC				4300
		A70J	4400	4400 CC				4400
		A70K	4500	4500 CC				4500
		A70L	4600	4600 CC				4600
		A70M	4700	4700 CC				4700
		A70N	4800	4800 CC				4800
		A70O	4900	4900 CC				4900
		A70P	5000	5000 CC				5000
		A70Q	5100	5100 CC				5100
		A70R	5200	5200 CC				5200
		A70S	5300	5300 CC				5300
		A70T	5400	5400 CC				5400
		A70U	5500	5500 CC				5500
		A70V	5600	5600 CC				5600
		A70W	5700	5700 CC				5700
		A70X	5800	5800 CC				5800
		A70Y	5900	5900 CC				5900
		A70Z	6000	6000 CC				6000
		A70A	6100	6100 CC				6100
		A70B	6200	6200 CC				6200
		A70C	6300	6300 CC				6300
		A70D	6400	6400 CC				6400
		A70E	6500	6500 CC				6500
		A70F	6600	6600 CC				6600
		A70G	6700	6700 CC				6700
		A70H	6800	6800 CC				6800
		A70I	6900	6900 CC				6900
		A70J	7000	7000 CC				7000
		A70K	7100	7100 CC				7100
		A70L	7200	7200 CC				7200
		A70M	7300	7300 CC				7300
		A70N	7400	7400 CC				7400
		A70O	7500	7500 CC				7500
		A70P	7600	7600 CC				7600
		A70Q	7700	7700 CC				7700
		A70R	7800	7800 CC				7800
		A70S	7900	7900 CC				7900
		A70T	8000	8000 CC				8000
		A70U	8100	8100 CC				8100
		A70V	8200	8200 CC				8200
		A70W	8300	8300 CC				8300
		A70X	8400	8400 CC				8400
		A70Y	8500	8500 CC				8500
		A70Z	8600	8600 CC				8600
		A70A	8700	8700 CC				8700
		A70B	8800	8800 CC				8800
		A70C	8900	8900 CC				8900
		A70D	9000	9000 CC				9000
		A70E	9100	9100 CC				9100
		A70F	9200	9200 CC				9200
		A70G	9300	9300 CC				9300
		A70H	9400	9400 CC				9400
		A70I	9500	9500 CC				9500
		A70J	9600	9600 CC				9600
		A70K	9700	9700 CC				9700
		A70L	9800	9800 CC				9800
		A70M	9900	9900 CC				9900
		A70N	10000	10000 CC				10000
		A70O	10100	10100 CC				10100
		A70P	10200	10200 CC				10200
		A70Q	10300	10300 CC				10300
		A70R	10400	10400 CC				10400
		A70S	10500	10500 CC				10500
		A70T	10600	10600 CC				10600
		A70U	10700	10700 CC				10700
		A70V	10800	10800 CC				10800
		A70W	10900	10900 CC				10900
		A70X	11000	11000 CC				11000
		A70Y	11100	11100 CC				11100
		A70Z	11200	11200 CC				11200
		A70A	11300	11300 CC				11300
		A70B	11400	11400 CC				11400
		A70C	11500	11500 CC				11500
		A70D	11600	11600 CC				11600
		A70E	11700	11700 CC				11700
		A70F	11800	11800 CC				11800
		A70G	11900	11900 CC				11900
		A70H	12000	12000 CC				12000
		A70I	12100	12100 CC				12100
		A70J	12200	12200 CC				12200
		A70K	12300	12300 CC				12300
		A70L	12400	12400 CC				12400
		A70M	12500	12500 CC				12500
		A70N	12600	12600 CC				12600
		A70O	12700	12700 CC				12700
		A70P	12800	12800 CC				12800
		A70Q	12900	12900 CC				12900
		A70R	13000	13000 CC				13000
		A70S	13100	13100 CC				13100
		A70T	13200	13200 CC				13200
		A70U	13300	13300 CC				13300
		A70V	13400	13400 CC				13400
		A70W	13500	13500 CC				13500
		A70X	13600	13600 CC				13600
		A70Y	13700	13700 CC				13700
		A70Z	13800	13800 CC				13800
		A70A	13900	13900 CC				13900
		A70B	14000	14000 CC				14000
		A70C	14100	14100 CC				14100
		A70D	14200	14200 CC				14200
		A70E	14300	14300 CC				14300
		A70F	14400	14400 CC				14400
		A70G	14500	14500 CC				14500
		A70H	14600	14600 CC				14600
		A70I	14700	14700 CC				14700
		A70J	14800	14800 CC				14800
		A70K	14900	14900 CC				14900
		A70L	15000	15000 CC				15000
		A70M	15100	15100 CC				15100
		A70N	15200	15200 CC				15200
		A70O	15300	15300 CC				15300
		A70P	15400	15400 CC				15400
		A70Q	15500	15500 CC				15500
		A70R	15600	15600 CC				15600
		A70S	15700	15700 CC				15700
		A70T	15800	15800 CC				15800
		A70U	15900	15900 CC				15900
		A70V	16000	16000 CC				16000
		A70W	16100	16100 CC				16100
		A70X	1620					

[illegible]

113	HONDA	BALLAD	UK	1010	XL	MOTITEGO	X5	AUS	HUV	XX	UK	010
115	ROYALTY				XF	METRO 2	X1	AUS	HUV	XX	JAPAN	010
116	HONDA	XX	JAPAN	010	YG		XV					
119	HONDA	HB	UK	010	ZH	HOVER 2000	XW	AUS	HUV	HB	UK	010
120	S. D. 1				XA	AC 16	XY	10005.1	0011	-	SALTS	
121	XX	010			XB	01011	XZ	ADU/1	PHHLS	1981		
122	XX	MAT 5100			XP	IG	X2	AUS	HUV	HB	JAPAN	010

Figure 36: A page from the TGIR.

Figure 35: A page from the TCIR.

relating to his application, the Additional Feature Charts will be the main document which will describe this information at its lowest level.

In the example cited, the auditor through the Rover R8's Additional Features Chart can view any detail concerning the specifications of the seats and their restrictions and his job is then to match them with the specification and restrictions given by the engineer. In order to do that he has to understand the rationale behind of the new design of the seat and accordingly mix the features which condition it. As stated in [9], "a correct usage statement should only include those features which condition the fitment of the part in the vehicle".

It should be noted that although the engineer is the prime author for the design of his part, it is the experience and knowledge of the auditor for the product and in particular for the part itself which ensures the correct usage condition. That is, it is the auditor who has the responsibility for actually loading the usage statement in Rover's database. He, also, has a better understanding than the engineer for the part, as he has coded its usage condition in the past. It can be concluded then that it is his function to code the usage statements of the parts in the most optimum way; ie. to avoid duplication of the information and make the design intentions of all the parts obvious to its users.

Thus, in the example, if the front seat of the Rover R8 has been designed with a new heating mechanism, perhaps because the direct competitor of this product, the Ford Escort, offers such

BASE FEATURE CHART									
MODEL RANGE ADS NOV 00 00 00 00									
DATE 04 Oct 09									
DESIGN DET. AUTHORITY									
EFFECT STA REFERENCE									
RESTRICTIONS									
GROUPS									
TERRITORY									
THREAT CODE									
THREAT TYPE									
V A R I A N T									
LUGGAGE									
CAPACITY									
PERF									
R/T									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									
A/B									

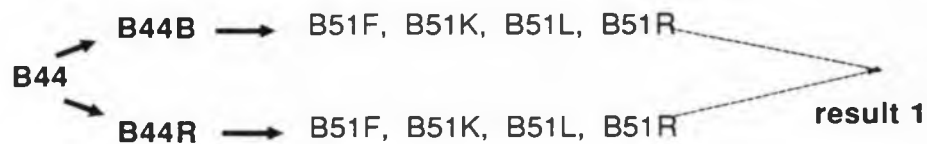
option, then the feature group which controls the heating of the seats of the vehicles must be incorporated in the usage statement of the seat.

For the purpose of the demonstration, we assume that all the three feature groups, valid according to the AFGIR, "B51: SEAT MATERIAL FINISH", "B44: SEATS -RECLINING", "B69: SEAT HEATING", are needed to condition the usage statement of the seat in the example. Then, the valid combinations of all the features within a feature group will determine all the different seats which can be made for this three feature group specification area.

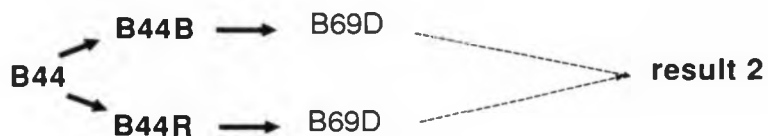
The auditor generates all the usage statements for the seat, ie. all the possible newly designed seats, using a mix and match method applying to the above participating feature groups. The term **validation algorithm** has been adopted in this research to formalise the procedures used.

The algorithm is an iterative process which combines at each step two feature groups (parent groups). The result of their combination (child group) is recorded in order to be combined later with the rest of the feature groups. The depth of the algorithm (ie. sequential steps to the final result) can be zero if the part was offered unconditionally to all derivatives of the model, one if only one or two feature groups condition the fitment of the part into the vehicle, three if three feature groups are involved in the usage statement or more generally $(n - 3) + n$, where n is the number of features (see section 8.2.1.2.1). The root of the search of the **validation algorithm** is the trim levels of the product and it performs a three

Step 1



Step 2



Step 3

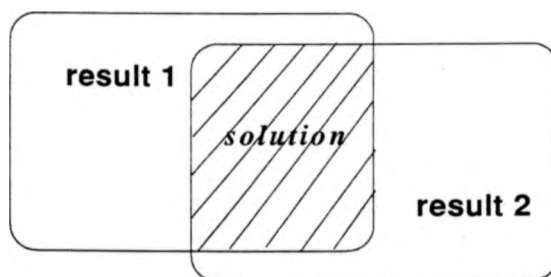


Figure 39: Diagramatic representation of the validation algorithm's process

dimensional check at each step ie. the the base features of the car, the combinations restrictions and the territory restrictions of the features.

The validation algorithm can use any feature group as the

reference point of its combinatorial search. Assuming that the master (pivot) feature group in the example is the "B44: SEATS - RECLINING" group. Then, each of its features will be combined with all the other features of the rest of the feature groups as the figure 39 shows. (A detailed analysis of the validation's algorithm process is discussed in the next chapter).

In summary, the first phase of the audit process represents the core. The auditor, individually, validates each page of the specification package representing a physical part, based in his experience with the parts and the information he collects through a series of steps and documents (Features List, Additional Features Group Index Report, Base features etc.). At the end, the validation algorithm uses this information to identify the appropriate sections of the Additional Feature Charts from which it extracts further data and compiles them into the boolean codes of the usage conditions of all the physical parts in the specification package.

5.1.2 PHASE 2

Figure 40 depicts the last two phases of the auditing function.

In the last two phases, the auditor examines the specification package as a single document of information rather than independent physical parts.

Each specification package, as mentioned earlier, usually represents an area of the car and the physical parts which it contains, ie. the assembly structure of that area.

In the second phase, the links between the physical parts within the specification package are investigated in order to ensure that their relationships correspond with the physical assembly structures of that area of the car. Those links are expressed, through the part number reference under the "NEXT ASSY. NO" heading, in the document of every physical part (fig. 23, 24). For example in figure 24 the physical part "HGB100007" with part description "COVER-ASSY-SEAT-FRONT-SQUAB" is assembled to the higher level assembly with part number "HAD10006". In this case the top assembly is "SEAT-FRONT COMPLETE MANUAL" and the relationship is correct as can be seen in figure 41. Figure 41 depicts the assembly structure of the "SEAT FRONT COMPLETE MANUAL". The equivalent drawings are shown in figure 42.

The most common errors in this case are the transcription errors, for example, the incorrect copying of a part number. Despite being though simple mistakes, these can distort the Bill Of Materials which the specification package represents.

The most important issue of the second phase is the validation of all the quantities for every physical part in the assembly. This process uses the "USAGE QUANTITY" and "QUANTITY" sections of the documents of the physical parts (figures 23, 24) to complete missing information. The equation for two physical parts X and Y, where X is fitted only to Y, is:

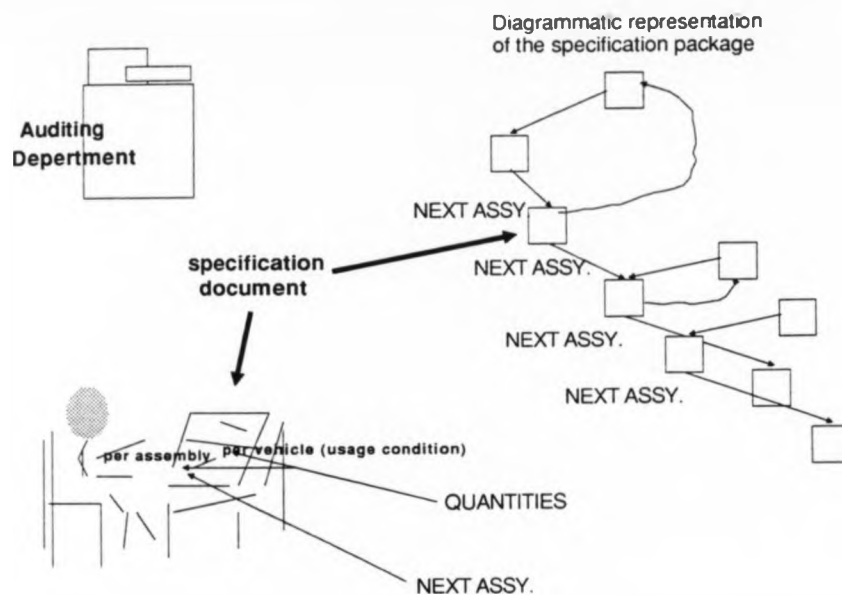


Figure 40a: The second phase of the Audit process

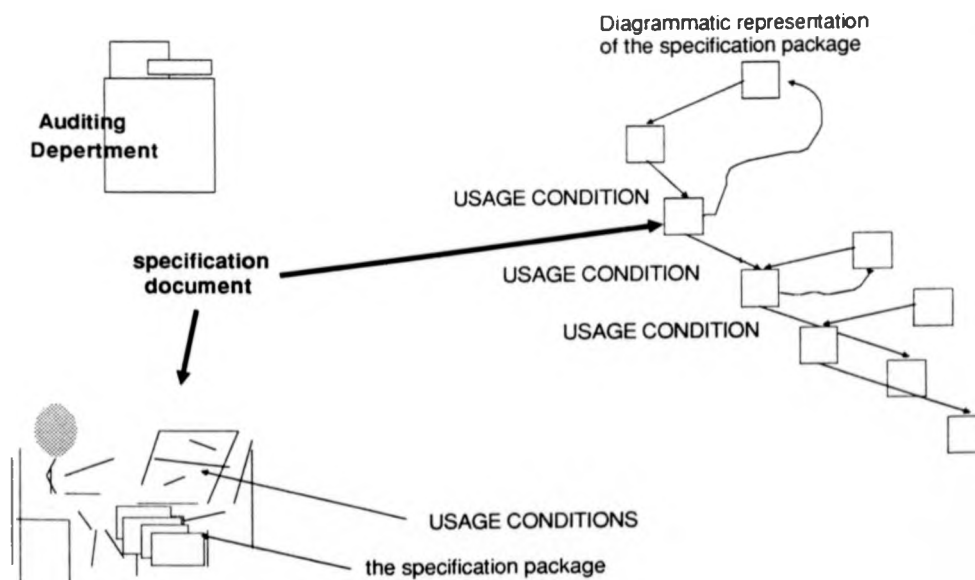


Figure 40b: The third phase of the Audit process

$$USAGE-QUANTITY_y = USAGE-QUANTITY_x * QUANTITY_x$$

The equation becomes more complicated when the part X is fitted to more than one assembly. The audit process of the parts's quantities is discussed in detail in chapter 9.

In the example, the missing quantity of the part "HGB100007" is 2 as results from the equation : $2 = 1 * ?X$. The question mark ("?",) corresponds to the question mark in figure 24.

5.1.3 PHASE 3

The correct validation of the usage statements in phase one represents the major part of the work of the auditing process (almost 90%). It is a difficult task because it requires the knowledge and experience of both the general philosophy of the automobile manufacture and also a good understanding of the specific product (ie. Rover R8) and the vehicle parts (ie. seat). Because this wide-ranging knowledge is required the risk of error in the usage statements of the parts is always present. The third phase, consequently, double checks the usage statements of the parts by using inputs from phases one and two. That is, it compares the usage conditions, derived at phase one, among themselves based on the relationships of the parts they represent which are audited at phase two.

Phase three, in other words, completes the original audit of the usage conditions especially those lower in the assembly level

hierarchy parts. It helps to diffuse consideration of the parts which are affected by the change. The general rule is that the logical summation of the usage statements of the parent parts should equal the usage statement of the child part.

In the example shown in figure 41 the assembly of the heated front seat is cascaded down to the elementary part which makes the difference to any other top level front seat assembly. This is the heating element ("ELEMENT-HEATER-ST/F SQUAB" HEATING") which added in the SQUAB's OVERLAY subassembly.

The top level front seat assembly "SEAT-FRONT COMPLETE manual" is dependent on the feature group "B69: SEAT HEATING" and similarly the SQUAB, its OVERLAY and the HEATING ELEMENT must dependent on the "B69: SEAT HEATING" feature group (figure 41). (This cascade of all the usage statements of the parts in an assembly, starting from the top level, is discussed in more detail in chapter 9).

If the feature group "B69: SEAT HEATING" does not appear in either of the usage conditions of the above parts then such a part is wrongly specified as inferred from its assembly relationship. In that way, the auditor picks up missing specifications at the lowest level by cascading the specification package from its top assemblies downwards to the elementary components.

5.2 SUMMARY

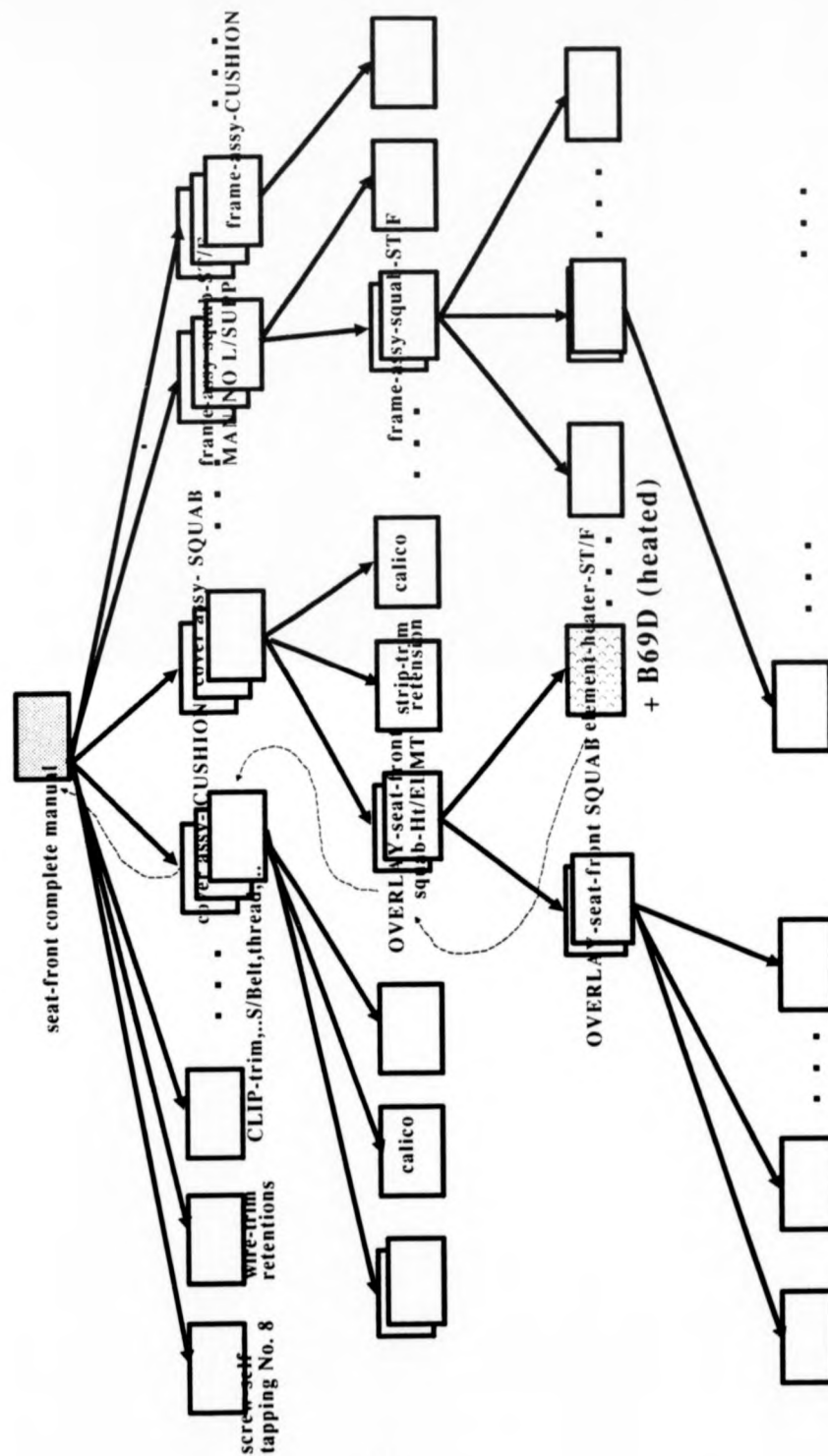


Figure 41: Part of the "SEAT-FRONT COMPLETE manual" assembly

The auditing function, though mundane in nature, represents the backbone of the correct application of the Product Specification Concept. It operates as an extension to Component Engineering in order to correctly specify the new design intentions for the parts of a vehicle. It is a difficult exercise because of firstly the use of codes which creates the possibility of many errors and secondly the engineering design knowledge it requires, such as that a front car seat becomes heated by fitting heating elements on either or both of its CUSHION and SQUAB overlays or that sport seats take this style by fitting a lumbar subassembly to the basic seat frame.

In addition, the analysis of the audit process can be only vaguely specified sequentially as the known information relevant to the application is literally dependent on the experience and knowledge of the auditor. For example, the auditor may know the reason why a new part is specified if he happens to have discussed it with the engineer, together with the changes in its assembly structure. In this case he would start with the third phase. In the case that he has no knowledge of the new design, then he would start with the first phase by consulting the Features List and the BAFC of the model.

This research, has therefore identified three distinct phases in the auditing function:

- (i) audit of the usage statements of each physical part
- (ii) assembly and quantity relationships and finally
- (iii) usage conditions relationships.

A combinatorial algorithm can, also, be derived, from the existing manual operation that auditors apply to the Additional Features Charts of the model in order to specify the parts. This algorithm can be designed and proved to be practically correct and some of its properties postulated; the depth of its search, the time of its process etc.

Finally, using both the analysis of the audit phases and the new algorithm, a computerized expert system could be developed to behave *knowledgeably* within the problem domain and experiment with the automation of the auditing function.

6. CHARACTERISTICS OF THE PROBLEM

This chapter summarises the characteristics of the problem which the investigation has revealed. Firstly, the PSC, giving a general picture of the rationales for which the Rover's specification concept is still in the area of research and experimentation. Then, most specifically the problems met within the Auditing function and how it affects and is affected by the overall PSC in the company and how such a function could be automated or supported by a computer system.

The general idea is that experimenting with the specific application of the PSC, which is the Auditing function, problems should become clearer as they would be of an applied nature rather than just theory. Their solutions, then, could be more clearly defined and investigated. If a computer system could be developed to support the Audit function, then the software development of the whole PSC in Rover could grow more naturally by extending further the existing system. In that respect, the automation of the audit function could become "vehicle" for clarifying the various concepts within the PSC and prototyping its automation.

The first section of the chapter outlines the characteristics and complexities of both the Product Specification problem and the Audit function, which lead to them still being a subject for research.

From the investigation two types of problems with the PSC within Rover can be distinguished: complexities, specific to its application in the company and general characteristics of such a problem which make its solution difficult.

6.1 complexities of the PSC

6.1.1 *duplication and re-formatting of information*

It was discussed in chapter 4 how the PSC becomes obscure with the reformatting of the original vehicle information from Features List to the AFC data.

6.1.2 *unclear interrelation of the various functions of the PSC.*

It was mentioned that the Audit function was invented by Rover as an 'artificial' mechanism to tackle the mistakes occurring in Component Engineering. Such a mechanism represents a classic example of how very large organisations, including Rover, usually tackle problem in complex system such as the PSC: new functions are invented and fitted somewhere inside the problem framework as autonomous entities rather than being integrated with the rest of the problem. This, however, creates additional problems such as:

- (i) the interrelation of the problem components becomes

looser. It creates duplication in the information and overlapping of the tasks of the new function with some of the existing ones.

(ii) the overall PSC becomes more vague in the way that part of the additional detail is entered into the problem framework.

Furthermore, because of the high complexity in product specification, the various departments involved (Purchasing, Auditing, Specification Services, Manufacturing etc.) tend to "tailor" their functions to their own needs which in return leads to views of the same problem from many different (and increasingly narrow) perspectives. Rover's PSC thus suffers from a proliferation of autonomous functions instead of a single integrated one. Clarity which would be the result of integration of the various functions has been sacrificed in favour of high specialisation and handling the whole complexity each function at once.

Because of the lack of clarity in the whole procedure of the Rover's PSC, it is difficult to study the Audit function in isolation. It is equally difficult to clearly specify the PSC's functions upon which the audit function is dependent or the functions which auditing directly supports.

6.1.3 *fear of the new technology*

It has become apparent through the investigation of the problem

PAGINATION ERROR

p104

that some people in various departments do not wish to co-operate in computer implementation of their job as they are afraid for their job security because of computer automation. In another situation, people have worked in the Auditing department manually throughout the period of their employment and have met the difficulties of the process in a daily basis such that they do not believe it is possible to develop computer solutions and consequently they are less than co-operative.

6.1.4 *confidentiality*

Literature in the subject is very limited if non existent. That is because of the confidentiality of such business in the industrial sector. Even when literature is found it is so highly specialised to the needs of the individual company that it is of little use.

6.2 general characteristics of the PSC

6.2.1 *knowledge and experience*

As the PSC tends to high specialisation in each of its functions, its application and maintenance are even more dependent on the knowledge of the experts scattered around the various departments of the company. For example, component engineers know the

characteristics of the design of the parts with which they deal together with the update peculiarities of those parts. The Specification Services people can recognise features which are likely to change through the lifetime of the vehicle. Manufacturing and Purchasing know the lead times for building or purchasing the parts.

Such types of knowledge of the PSC are not fully documented anywhere within the company and the amount of labour required for it to be consolidated is great because of the amount of data involved. More specifically for the auditing function, such a knowledge is hard to code because of its qualitative factors such as "common sense" and "experience".

6.2.2 *dynamic environment*

The automobile environment is highly dynamic. The design conditions of a product can change according to legal, climate, technological, or market requirements. Rules which may used to be reliable in the past may not have any value in the future.

There follows a discussion of the difficulties of the auditing function, more specifically.

6.3 complexities of the Audit function

6.3.1 use of codes

The auditing function is closely linked to codes. In reality auditors work so heavily with codes that they tend to think and express themselves in codes rather with the meanings of the codes. As humans are not generally good at remembering and manipulating codes, mistakes in the auditing process are likely to happen.

6.3.2 use of many different hardcopy files

Auditors sometimes have to work with ten hardcopy files simultaneously. In general, the least number of files that auditors work with at the same time is three plus their own notebook for notes and calculations. Actually, as simultaneously working with so many files is such an everyday procedure in the Auditing department, practical methods of manipulating the paperwork manually better have been developed through experience and these are taught to the trainees during the auditing course.

6.4 characteristics of the Audit function

6.4.1 unclear procedures /subjective information

Each auditor follows his own way of usage condition validation dependent on his own knowledge and experience. This means that he may have alternative ways of reaching the result other than the three audit phases discussed in the previous chapter, or he may use them but in a different order.

Although the audit function has grown satisfactory during the last 10 years, it is still subject to "heavy functionalism" - Rover's terminology [20]. Heavy functionalism means that it is possible for auditors to interpret the design intentions of the engineer differently and consequently code similar information in the data base differently. Besides each one could exhibit equally valid arguments to support the correctness of his usage statement ie. "my usage is better than yours".

6.4.2 *need for multidisciplinary knowledge*

It is quite important for a person (or a system) to have knowledge of many different aspects of the product in order to do the auditing job. That is manufacturing knowledge, design engineering knowledge, product specification understanding. Most important is that all this knowledge is accumulated only through experience.

In the rest of the chapter the reasons for which the PSC and the Audit function could possibly benefit from a computer system is discussed, based on the complexities and characteristics

discussed previously.

6.5 Why the PSC and the auditing function represent an appropriate field for computer application.

Although the vehicle specification and/or auditing area in automobile are still done manually, as the intelligence required in such a business is dependent on the "common sense" and "experience" of experts, the analysis of the complexities and characteristics of the problem show that this could be computerised, in the following.

The PSC handles huge amounts of data of a dynamic nature (ie. update, deletion, insertion). Huge amounts of data, updated regularly can be more efficiently manipulated by computers than manually by humans.

The study of the PSC has shown that as the vehicle information is released through the various departments of the company, the same auditing procedure occurs at each build phase of the vehicle (at least 11 times): sourcing, update, specification, validation, loading in the data base. The only variable is the time. Computers are good for iterative (usually mundane) type of work as it is only necessary to implement the procedure once.

The auditing function in particular makes heavy use of codes with which computers are better designed to work than humans.

The use of many hardcopy files in the simplest situation could be

replaced by many computer screen windows and user friendly interfaces.

The semantics (features) and syntax ("+" "-" ..) of the PSC are implemented in boolean algebra. Consequently algebraic logic could be used to infer rules. Computers handle algebraic problems very efficiently.

The creation of the usage conditions of the parts is done through the application of combinations of features. Computers are much faster at combinatorial problems rather than humans.

It was mentioned that a huge amount of labour and time is required to collect knowledge from experts. In addition, the type of such knowledge is heavily dependent on experience.

Software research, nowadays, seeks to develop programs which could assist in knowledge acquisition and learn from experience. In that respect a computer could interactively consult the auditor or engineer with probably a highly refined knowledge derived from statistical analysis of existent data.

In the PSC parts are linked together by engineering and quantitative links; such relationships could be built directly within the computer memory, rather than using the flat structure of hardcopy files. In this way the representation of the knowledge of the problem could represent the problem domain itself resulting to higher naturality, understanding and performance.

Finally, computer programs could replace the subjectiveness

caused by humans in any aspects of the PSC or the auditing function.

The following section introduces to the research of the software tool required to satisfy the objectives cited above.

7. INVESTIGATION ON THE SOFTWARE ENVIRONMENT FOR THE DEVELOPMENT OF THE SYSTEM

This chapter investigates the software environment needed for the automation of both the Audit function and the PSC, based on their characteristics discussed in the previous chapter. The research extended into the subjects of abstraction hierarchy, Meta-knowledge, programming and database technology.

The audit function has been designed by Rover to control the correct application of the PSC. Thus, it is bound to maintain, even vaguely, the information needed to tackle the product specification problem in Rover. By understanding better the Auditing function, it was thought that the Product Specification Concept should become clearer and more easily automated. For this, the design of the overall system was firstly focused on the automation of the auditing function. Besides, this was the original objective of the project.

Things that may help the automation of the PSC, as well, are pointed out during the discussion which follows.

7.1 *abstraction hierarchy.*

The automation of the Auditing function within Rover could not immediately be understood in its entirety, therefore it was

necessary to design it in easily understood components.

The general idea was first to solve the problem in an *abstraction level*, a simplified representation of the problem in which not all the details are involved. When a solution to the problem at the *abstraction level* was discovered all that remains would be to account for the details of the linkup between the steps of solution. This approach should also help the automation of the PSC, as such a solution at an abstract level should closely reflect the solution required for the automation of the whole PSC. That is, as the audit function represents the PSC's application, its solution at the abstract level should be the same as the theory which it follows, ie. the PSC in Rover.

The term for this approach, gradually inserting details in the design of the problem, has been borrowed from [48] in the AI field and it is used in this thesis as "planning in a hierarchy of abstractions". In some cases it may be referred as "hierarchical planning" or "abstraction hierarchy", as well.

Planning in a hierarchy of abstractions, in the context of this thesis has been used to ease naturalness and understanding in the system design and facilitate rapid prototyping. These reasons are of great importance in the implementation of the system as

- no work relevant to this project has till now been carried out in automobile industry, to which one could refer for literature and/or evaluation of alternative approaches

- there seemed to exist multiple alternatives for the implementation of the system and it was difficult to judge the

best one when no results of a prototype program exist.

Multiple alternatives arised mainly for two reasons:

Firstly, because of the different objectives which were drawn during the development phases of the system. The first audit phase, for instance, was implemented with the view to re-configuring the Audit process in a manner such that it could be mechanised and guarantee correctness in the creation of the usage conditions of the parts. That required the manipulation of **part descriptions** at an abstract level.

The development of the last two phases of the Audit function, however, had different objectives. One needed to be concerned with the overall Product Specification Concept and study the actual **physical parts** and assembly structures in the company.

Secondly, there is no single department within Rover that possesses the total knowledge for the Product Specification Concept. The Auditing department did not have sufficient information - it eventually proved necessary for the designer of the system to understand the PSC rather than just the audit function. The required knowledge is scattered through the various departments of the company such as Component Engineering, Specification Services, Auditing, Manufacturing, Purchasing etc. which approach the problem from different angles, as well, subject to their profession.

In this perspective, it seemed vital that a planning technique be used which would allow the design of each phase to be tackled

independently but also allow the adjustments that may affect to the whole system, be easily implemented.

"Planning in a hierarchy of abstractions" provides such characteristics and the whole design can be updated when needed by simply modifying the interactions protocol of the system's subcomponents. This is obtained by using *satisfaction posting* and making use of the *almost hierarchical decompositions* conceptualism (discussed in the following).

Polya has discussed quite early on the concept of decomposing a problem to subproblems in order for one to study it, referring actually to the way which the human problem solving mechanism operates [96].

In the followings the benefits of such an approach in the design of the system are discussed.

7.1.1 CLEARER DEFINITION OF THE PROBLEM DOMAIN

With "planning in a hierarchy of abstractions" the apparent complexity of the design problem is reduced by understanding the better defined subsystems and hence the global exercise becomes clearer. An additional advantage of this approach is that the partitioning of the exercise can be done before the specifications of the subsystems and the implementation of the subsystems can be launched in parallel. This helped in the

initial stages of the research into the auditing function: the distinguishing of three audit phases although their internal processes were not clear at the beginning.

Furthermore, "planning in a hierarchy of abstractions" helps the knowledge engineer to distribute the acquisition of the system's knowledge evenly to the several specialists. A vital need for the computerisation of the PSC, in particular, where different departments are involved in the process and view similar information from different angles.

7.1.2 CONSTRAINT POSTING

(using constraints to interact the subsystems)

The **MOLGEN** system [54] is probably the best known paper in AI literature covering the means of clarifying terminology in "planning in a hierarchy of abstractions". *Constraint posting* is defined in **MOLGEN** to "represent the approach to hierarchical planning which uses constraints to represent the interactions between the subproblems. Constraints are dynamically formulated and propagated during hierarchical planning, and used to coordinate the solutions of nearly independent subproblems".

Constraint formulation, *Constraint propagation*, *Constraint satisfaction* are defined clearly in the **MOLGEN** as well and the reader can refer to [54] for more details.

The **MOLGEN**'s clear definition of the terms in the "planning in a hierarchy of abstractions" has been studied and used during the design of the computer system discussed in this thesis. However, *constraint posting*, as is defined in **MOLGEN**, represents the most important issue used in this thesis. That is, both consideration of the phases of the auditing function and how the whole auditing function fits into the PSC have been examined as independent subsystems without constant attention to their interactions. However, during the design of the audit function or the gradual learning of the PSC, interactions were always anticipated and were implemented later by the means of constraints when the knowledge of the problem had grown sufficiently.

7.1.3 MAKE USE OF ALMOST HIERARCHICAL DECOMPOSITIONS

Hierarchical planning by introducing constraints (or details) in the various steps of the planning process mainly comes from the work of synthesis and analysis of electrical networks. One of the first papers in this subject was **CONSTRAINTS** [47] which tackled the apparent synergy in the design of a circuit by *propagating* constraints in the sense of algebraic dependency analysis among the circuit's various components ie. adders, multipliers etc. **CONSTRAINTS**, especially, encourages the abstract analysis of a system to a hierarchical (tree-like) decompositions for simplicity but it also, maintains the idea that a strictly hierarchical description of the problem can only be an

approximation to its true structure of an object. Consequently, systems can only be decomposed to almost-hierarchical structures.

Figure 43 shows the functional operation of a timepiece - that CONSTRAINTS uses to illustrate the concept - which is not hierarchical but almost is.



Figure 43: The hierarchical decomposition of both the functionality and physical structure of a watch. From reference [47].

In the auditing function in particular, the manual example when structured, showed that the components which most probably would constitute a computerised system are: APN (part descriptions), EXPERIENCE LINK (design engineering knowledge), AFC (features), VALIDATION ALGORITHM (features combination) and a USER INTERFACE. At the beginning only a basic relationship to associate them together, a sequential reference, is adequate (figure 44).



figure 44: A linear (ie. a basic hierarchy) approach of designing the automation of the audit function

However, when the details of each subsystems entered and constraint posting used to interact the subsystems the design of the whole system did not look like as a hierarchy but rather as a network (figure 69).

7.1.4 EXHAUSTIVE SEARCH OF ALTERNATIVES

By using the abstraction hierarchy approach, it is possible to investigate several alternatives to the solution of each subsystem, individually. This is, because the different subsystems of the problem are considered independently, one can approach them from totally different angles. This means that even the philosophies of such approaches can be totally different from each another and still all of them interface together naturally in the whole design. This is the case with the audit function where the first phase is tackled from a totally different design perspective than that of the last two phases.

7.1.5 PREVENTING COMBINATORIAL EXPLOSION

MOLGEN [54] uses hierarchical planning with *constraint posting* to plan gene-cloning experiments in molecular genetics. One apparent advantage of the technique is the prevention of combinatorial explosion of the solution routes. The effectiveness of this is shown in the following rat-insulin example.

Elimination of Solutions

Antibiotic
 Saccharin
 Enzyme
 Linker
 Vector

Constraint	Combinations						
	Total	Considered					
	3456	0	9	3	32	1	4
Compatible	1152	4	9	1	32	1	4
Carries Resistance Gene	160	5	3	1	32	1	4
Has Sites	21	21	3	1	5	1	4
Doesn't cut Resistance Gene	10	10	2	1	5	1	4
Has Sites	4	4	2	1	1	1	3

Figure 44: Elimination of solutions using hierarchical planning, from reference [54].

If the variables were counted independently, the columns would be powers of the numbers shown. However, the numbers in the "Total Combinations" column show how the combinations in the MOLGEN

example reduced gradually from 3456 to 4 detailing at the various steps of the planning process genetics knowledge constraints.

Generally, when hierarchical and non-hierarchical approaches for the implementation of a computer system have been systematically compared, the former has dominated.

PLANNING IN A HIERARCHY OF ABSTRACTION SPACES

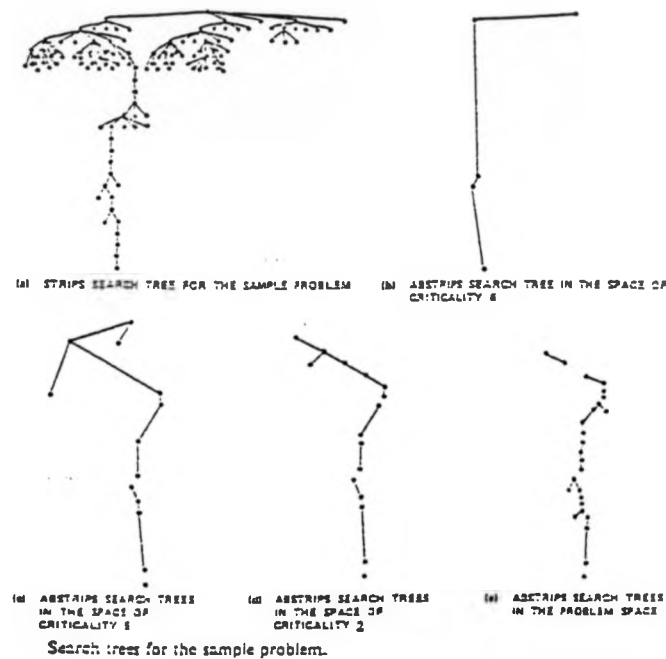


Figure 46: The search space trees a and b, using STRIPS and ABSTRIPS, respectively.

For example, the **ABSTRIPS** program [48] was a version of the non-hierarchical **STRIPS** planning program [49], retro-fitted with a scheme for abstract reasoning. **ABSTRIPS** demonstrated having implemented in a higher abstract level only the important information of the problem, the power of the heuristics of the program are augmented, by the means of directing the search more efficiently and avoiding backtracking. Figure 46 shows the difference in performance of the two systems **STRIPS** and **ABSTRIPS** in the robot application.

In summary, the manual audit example has shown that the whole part specification problem is governed by the correct use of constraints in the appropriate stages towards the route of the solution (ie. the vehicle model, design feature groups, territorial, combinational and base features restrictions).

Naturally the search leads towards the study of already existing systems which are built both on hierarchical planning and gradual constraint refinement.

"Planning in a hierarchy of abstractions" can help not only for the clearer definition of the problems discussed in this thesis, the audit function and the PSC, but if those problems are going to be computerised as well, combinatorial explosion in the search routes can be avoided by refining the level of the detail in their search space. A vital need for the computerisation of the audit function especially, as the usage conditions of the vehicle parts are obtained through feature combinations.

Many papers in AI literature refer to hierarchical planning, the most important are [50], [52]. A paper that approaches hierarchical planning from a quantitative point of view claims that "abstraction hierarchies can reduce time exponential problems to linear ones [51].

Having adopted hierarchical planning in the design of the system, the next section looks at the best way of representing the knowledge of the problem and the best software tools for its manipulation.

7.2 *"Planning in a hierarchy of abstractions" with Object Oriented knowledge representation flavor.*

Investigation in the computer development of systems in AI which encapsulate the concept of hierarchical planning has shown that Object (or frame) Oriented knowledge representation was almost universally adopted in such a system design.

The CONSTRAINT's system, for example, represents the various data types in a kind of 'constraints frames', whose initialisation depends on the values of their slots. For example a resistor is represented as a 'frame of constraints' in the followings:

```

(constraint resistor
  ((v1 number)
   (v2 number)
   ...
   (resistance number)
   ...
   (m multiplier)
   (== v1 (>> sum av))
   (== (>> a2 av) (>> product m))
   ...
   (== Resistance (>> m2 m))
   (== i1 (>> a1 ai))
   ...
  ))

```

In the above resistor definition, an expression such as "(>> a b)" means the "a of b". The notation "==" expresses equality. Thus, for the above definition, the keyword "constraint" is followed by a name, a list of component names and types, and a set of linkages. The resistor definition, in fact, represents a real object which enforces numerical constraints among two node potentials (ie. v1, v2), two currents (ie. i1, i2) and a resistance such as:

$$(v_1 - v_2) = i_1 \text{Resistance, where } i_1 + i_2 = 0.$$

For more details refer to [47].

In the **MOLGEN** system, as well, the different data types are represented in an object oriented way which distinguishes different laboratory objects such as a bacterium, a rat-insulin gene, a DNA-structure etc.

NASL [55] is a program for designing circuits hierarchically by combining and instantiating schemata representing functional subcircuits. Most of the "raw" information in the system is stored in "packets" defining known circuits, such as common emitters, amplifiers, voltage dividers etc. Additionally, decomposition rules of a circuit type are expressed as *plan schemata*, as well. Those are abstract objects, *instances* of which may be thought of as hanging as little subnets off nodes in the task (overall plan) network. The author of the paper, McDermott, claims that "Plans" or "packets" in NASL correspond directly to Minskian frames.

R1 [56] is a program that configures VAX-11/780 computer systems. The configuration task in R1 has been implemented as a hierarchy of subtasks that have strong temporal independencies. Given a customer's order, it determines, what, if any, modifications have to be made to the orders for reasons of system functionality.

In that respect it is closely related to the research of this thesis not only for its hierarchical planning design but for the automatic configuration of parts in particular, although it is concerned with computer instead of vehicle parts.

Each component in the data base consists of its name and a set of attribute/values pairs (*component information*). For example, the value of the attribute "type" for the component (object) "RK711-

EA", is *DISK DRIVE*. The system implements rules which indicate what components can be associated and what constraints must be satisfied in order for these associations to be acceptable (*constraint knowledge*). Currently there are 420 components supported from R1 and 480 rules of configuration manually extracted from the experts.

The following shows how information is stored in R1.

RK711-EA

CLASS: Bundle

TYPE: Disc drive

SUPPORTED: yes

COMPONENT LIST: 1 070-12292-25

1 RK07-EA*

1 RK611

RK611

CLASS: UNIBUS MODULE

TYPE: DISK DRIVE

SUPPORTED: YES

...

TRANSFER RATE: 212

BOARD LIST: (HEX A M7904) (HEX A M&903) (HEX A M&901)

...

CABLE TYPE REQUIRED: 1 070-12292 FROM A DISK DRIVE

UNIBUS DEVICE

Every component (object) in the database has a type attribute and a class attribute. There are 15 classes: bundle, cabinet, backplane, unibus module etc. Each component is connected with the other through specific object reference. For example, the "RK711-EA" component consists of several subcomponents including the "RK611" subcomponent disk drive, whose definition in the database is defined with another object datatype. For more details refer to [56].

7.3 *Meta knowledge* (automatic transfer of expertise)

During the investigation of the audit function it became apparent that the development of a computerised system to automate or support it would need *design engineering knowledge*. Such knowledge, however, is not documented anywhere in Rover. For this, research in this thesis moved to another topic in AI; the development of programs which could transfer expertise knowledge into the system.

Notice, in the context of this thesis, research of the existent systems in AI which implement Meta knowledge programs concentrated on those which analyse *empirical* data, as this is the case with Rover; already a huge amount of empirical data of the past exists. Additionally, the research is mainly concerned

with automatic transfer of expertise programs. This is because the huge amount of knowledge which is required to support both the audit function and the PSC makes an interactive approach impractical.

Similar problems of lack of knowledge in specialised fields has shifted AI search during the last decade to ways of automating knowledge acquisition. The theoretical basis of the implementation of such learning systems comes mainly from the fields of mathematics: fuzzy logic or probabilistic theory. (Bayer's rule, statistics etc.) [43].

Systems which already exist in AI field often re-present themselves with an additional *learning element* in their original implementation. **META-DENDRAL** [42] is a such example.

META-DENDRAL designed as the learning procedure of the already existing heuristic **DENDRAL** system [41]. The **DENDRAL** system implemented a "smart assistant" to help organic chemists determine the molecular structure of unknown compounds. This is required when no X-ray crystallography is possible and chemists have to resort to structure elucidation based on data obtained from a variety of physical, chemical or spectroscopic methods. The **META-DENDRAL** program refines further the analysis in mass spectrometry.

The result of the spectrum is depicted in a histogram. The **META-DENDRAL** program decides which data points in the histogram are important (among 100 to 300) and looks for fragmentation processes that will explain them. It attempts to form general

rules by correlating the plausible fragmentation processes of the various data points chosen. The production rules of the program are implemented in the way which the left hand side of the rule represents a description of the structure of a molecule and the right hand side of the rule expresses possible fragmentation processes of a such molecule. More specifically, bond cleavages and atom migrations.

For example, the below R1 rule:



indicates that such a molecule description on the left hand side it can be transformed to the right hand side where the asterisk (*) indicates breaking the bond at that position and recording the mass of the fragment (the molecule weight). If the most of the data in the histogram shows a similar molecule weight then the molecule structure can be found with high approximation. The **META-DENDRAL** program is actually made of two programs **INTSUM** (data Interpretation and Summary) and **RULEGEN** (Rule Generator).

7.4 *Meta knowledge with Object Oriented knowledge representation flavor.*

PROSPECTOR [40], **CENTAUR** [38] and **PIP** [39] assist experts in the areas of geology, pulmonary diagnosis and medicine generally (ie. glomerulonephritis) by analysing empirical data, as well.

The knowledge of these systems is represented with frame-like structures and production rules to perform the problem solving tasks. These frames (or prototypes) represent stereotype situations of the problem domain and they are used as a basis for comparison of the actual situation with that given by the data.

In summary, it has been discussed that "planning in a hierarchy of abstractions" can ease the understanding of the problem, the search for many alternatives, prevent combinatorial explosion and increase performance. Research in the AI field of systems which have been designed on this principle revealed that they implicitly or explicitly support the Object Oriented Paradigm in one form or another (specific frame definition such as R1 or MOLGEN, or constraint schemata CONSTRAINTS, NASL, ABSTRIPS). So, it can be seen, Object Oriented Programming, to be the most preferable environment for such a class of problems. From a pragmatic perspective, as well, this can be justified; OOP is concerned with the behaviour of objects, which can be expressed as constraints, and their relationships which can be expressed as almost-hierarchical decompositions. The objects themselves can even represent subgoals or plans in the system design exercise.

At the lowest contribution the Object Oriented Paradigm would promote *experimentation* in the design of the system as the behaviour of the objects could change much more flexibly than by using the conventional programming.

The need of both problems under investigation, the audit function and the PSC, to somehow enable knowledge to be transferred

automatically from the experts, led the search to the investigation of the existing systems in the AI field which implement Meta knowledge mechanisms. More specifically, those which analyse empirical data. As a result META-DENDRAL, PIP, CENTAUR, PROSPECTOR were studied. Surprisingly, the most of them showed a preference to the Object Oriented Paradigm, as well.

In conclusion, the Object Oriented Paradigm represents a suitable environment for the implementation of complex or unclearly specified problems and/or systems which need automatic transfer of expertise. However, still more search would need to be carried out at technical level for the choice of the software environment needed to automate both the audit function and the PSC in Rover. That came as a result of the situations occurred in Rover on the time and discussed in the followings.

7.5 *Rover's objectives and the project*

When the project was launched (Nov 1988), its objective was to automate or - if this did not prove feasible - to ease the audit function by introducing electronic equipment in the department. This project was a part of a more general CIE (Computer Integrated Engineering) project in Rover to interface vehicle specification data with the BOM (Bill Of Materials) [18], [93] (confidential).

The prime objective of the CIE (or BOM) project was to **replace** the hierarchical PIMS data base with another more flexible one. The alternative idea of the time was ORACLE. This was for two reasons:

firstly, ORACLE was available on the VAX platforms of the Business System and Product Planning departments in Rover.

secondly, ORACLE represented one of the best application databases of the relational model, a very popular database design model recently in industry.

Rover launched, almost simultaneously, another project with objectives very similar to those of the project discussed in this document. The objectives matched because, as was discussed earlier, the scope of this search extended from not only automating the audit function but the PSC, as well, as it became apparent that one cannot automate the audit function without firstly understanding the whole PSC in Rover.

The system discussed in this thesis has been named **ROOVESP** standing for Rover's Object Oriented VEHICLE Specification). A project was launched in parallel to **ROOVESP** which was named Illustrated Parts List (**IPL**). It was intended (and still is) that **ROOVESP** and **IPL** interface with each other in the future in order to support the whole Product Specification Concept of the company.

IPL, apparently has been designed with the idea that the host database will be ORACLE. In reality it needed to use an additional relational database, as well, 4th Dimension, in order to both fulfil this idea and also meet the overall business

objectives of the project.

It is appropriate to make an exhaustive comparison of the two paradigms as IPL uses the relational database model whereas the research in this thesis has indicated that the OOP is one of the most suitable environments to automate both the audit function and the PSC.

In the following section a clearer view of the relation of IPL with ROOVESP is discussed.

7.5.1 IPL and its relation with ROOVESP

Historically, IPL was launched by Business System and Component Engineering some months after of the ROOVESP project (Jan 1990). The two projects are linked functionally together in the Rover's original intention to merge them in a single new system which would support the existent Product Specification Concept.

Figure 47 illustrates the intended output of the new system, which was designed to replace the original format of the input documents in the specification packages (figures 23, 24).

Graphically, the relation of the two projects can be illustrated by the columns in which figure 47 is divided. The first two columns are of the main concern in the IPL project. That is, the integration of images and part descriptions, part fitments and

Figure 47: The new format of the specification document in IPL.

part numbers in a single document. The third column represents the intention of **IPL** to enter intelligence, besides the user interface facilities, in the system. That is, the ability of the system to validate the usage statements of the parts appearing in the screen, which naturally coincides with the the objectives of **ROOVESP**.

With the output objective of figure 47 Rover had a sequence of demonstrations from three companies: XEROX [85], APPLE [86] and SYMBOLICS [3], [4] in order to choose the most appropriate software tool which would combine efficiently such texts and images. XEROX was rejected because it offered only a publishing document, whereas APPLE and SYMBOLICS offered two databases to tackle the problem: 4th Dimension and Statice (see Appendix 2), respectively.

Feasibility studies were carried out on the two products which resulted in Rover using 4th Dimension for the **IPL** project and the project covered in this thesis using Object Oriented programming and later the Statice Object Oriented Data Base using the **GENERA** environment of the SYMBOLICS Lisp Machines [4]. G. Sussman and G Steele [47] suggest that Lisp is a suitable environment for expressing constraints (such as the ones which are required to specify a vehicle ie. the vehicle model, Base features, Additional features restrictions etc.) by using the lambda abstraction (For more details see [62]).

The availability of McIvory [5] in particular, a SYMBOLICS machine which incorporates both Lisp and Apple Macintosh environments, made such a choice even more preferable:

- if the computer implementation of the audit function and/or the PSC were to fail in using the OOP approach, Rover could still utilise the option of the 4th Dimension without the need to buy new hardware.

- even if ROOVESR and IPL, were implemented in different software environments such as GENERA and Apple Macintosh, they could still interface in the SYMBOLICS McIvory machine.

The major obstacle for Rover in choosing an OODB was the confidence which the people in the company had to work with the traditional relational data bases rather with a new software technology. Another reason was that the prime idea of ORACLE as the host database of the whole CIE project was becoming obsolete by the use of Statice.

In the following two sections, IPL is described and the feasibility study which direct compares the relational databases (using as example ORACLE) with the OODBs (using as example Statice) is given.

IPL

IPL uses the Oracle and 4th Dimension relational databases to interface texts with images. The way Oracle and 4th Dimension are linked in IPL is shown in figure 48.

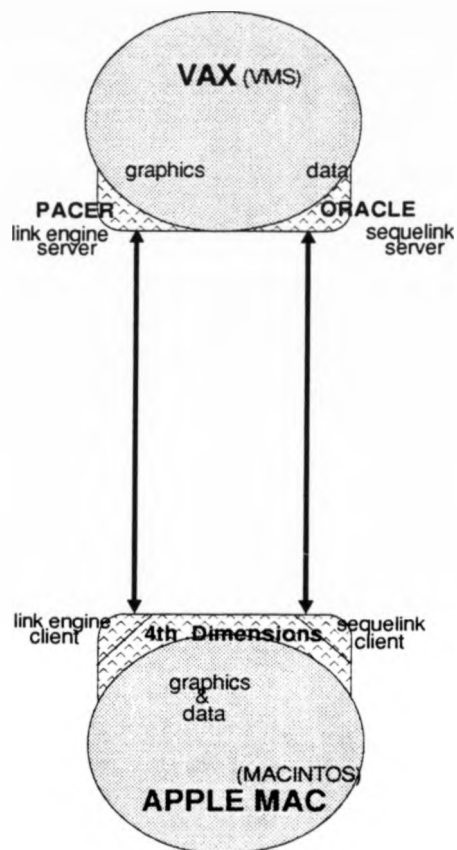


Figure 48: The Illustrated Parts List system

Data coming from PIMS pass from a C language based program (pro*C) which fills already existing data tables in ORACLE. This translation of the IBM mainframe data to ORACLE data is required in order for **PIMS** data to communicate with 4th Dimension. ORACLE represents the host server DBMS of the **IPL** system whereas 4th Dimension plays the role of "gluing" images to the PIMS data (text).

Text interface between ORACLE and 4th Dimension is implemented through the standard relational databases SQL language which both data bases support. The images of the IPL system are kept in the micro Vax in the form of scanned input. 4th Dimension which operates in an Apple Macintosh machine accesses the images from the host graphics library manager through a software package called PACER. Finally both text and images data are presented to the Engineer under 4th Dimension platform where now he can use its front end facilities: the WIMP environment [86] ie. Windows, Icons, Mouse, Pop up menus.

The engineer operates in the following sequence:

Firstly, he specifies the new parts or changes which he applies to the old parts output on the screen eg. deletion of parts, or creation of new ones, swap one part for another.

Secondly, he changes or specifies new usage statements for the parts. IPL guides the engineer's choice for only a specific collection of features groups that correspond to the VPG group he is working on. The implementation of such relationships between VPG groups and features groups is currently operated manually on a prototype basis using knowledge gathered from engineers.

A part of the IPL's logical E-R (Entity-Relationship) diagram shown in figure 49 gives a flavor of such a mapping of design features groups to VPG groups.

The part which shows similarity with ROOVESP is illustrated on the top left hand corner of the figure. The "Products" table represents the set of part descriptions which appear on the

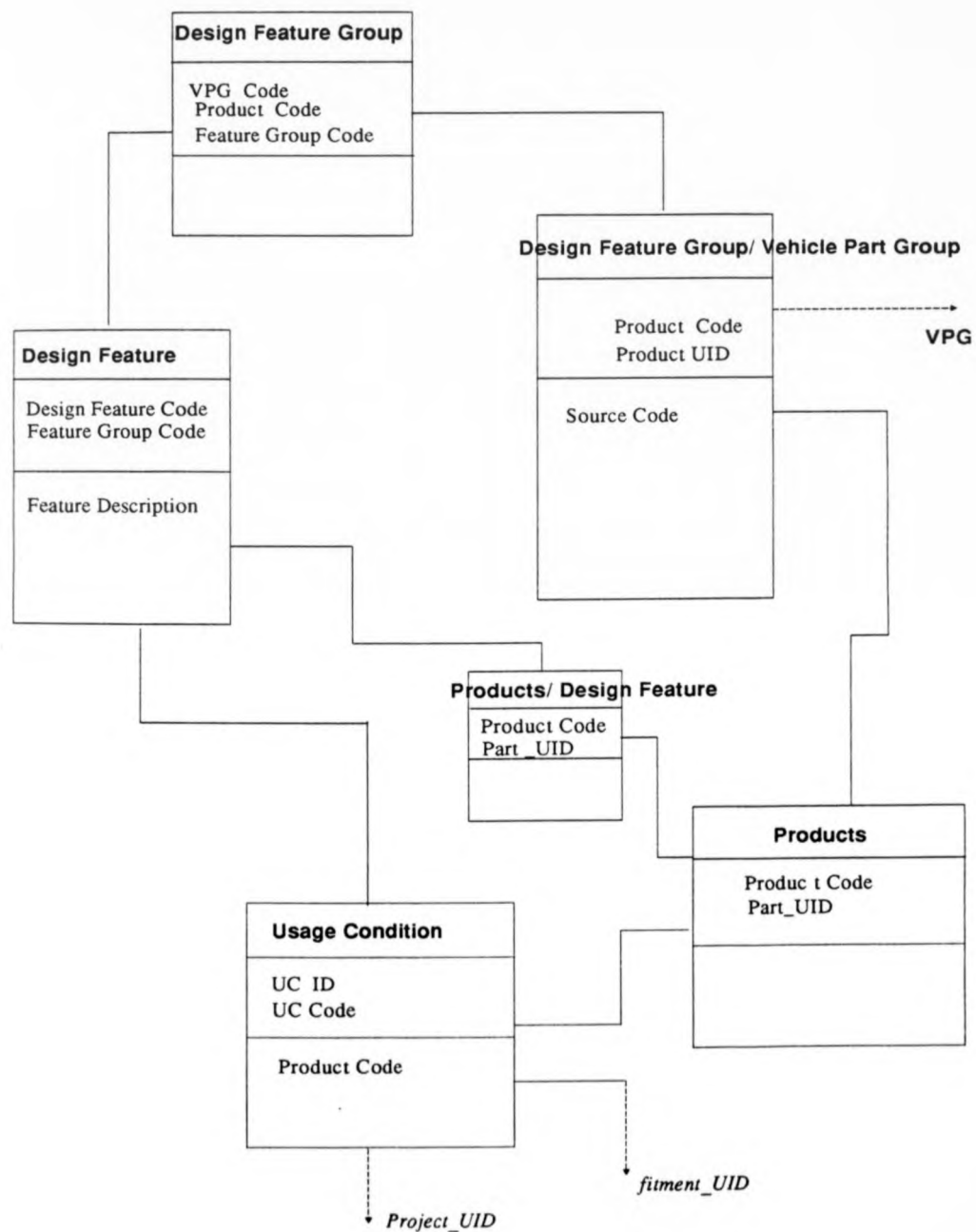


Figure 49: A part of the IPL's E-R logical diagram.

screen ie. "seat-front-complete manual". This set of part descriptions - in reality a VPG group or a part of a VPG group - is linked to the "Design Feature Group/Vehicle Part Group" table. This table maps Rover's design feature groups on to the VPG group under consideration. In reality, this table is the result of the questioning of human experts for the specific VPG, "filtered" from the Additional Feature Group Index Report for the availability of some features groups to the specific model. (Product/Design Feature table).

In order to change an image the engineer must have of the 4th Dimension software and make his changes with a scanner or a drawing package.

Finally, when the specification is completed, the changes are hardcopied and transferred back to Specification Services where people manually update PIMS, and the cycle continues.

Notice, that in order IPL to meet the original objective of the project, ie. to replace PIMS, the data from PIMS which filled the ORACLE tables at the beginning of the exercise are kept in the ORACLE data base. The general idea is that all data from PIMS will eventually populate ORACLE. Consequently, there is duplication of the information in PIMS and ORACLE. In result an additional load of work has moved to IPL, to keep 'in line' the two databases.

Having outlined the functionality of the IPL project which is based on the relational database model, this can now be used as an example of the relational database model in a direct

comparison of the Object Oriented databases and more specifically Statice.

7.6 Object Oriented Programming and comparison with the traditional programming and the Relational paradigm.

This chapter compares the relational database model, the most popular traditional database technology, with the OODB model, recently emerging on a commercial basis from the development stage. It was mentioned previously that such a comparison is done by using as examples real databases: ORACLE and Statice. It is not felt appropriate to describe ORACLE here as it is a well established product and it has been used widely in universities and industry. A description of Statice (and FLAVORS) however can be found in appendix 2 (and appendix 1) which uses examples of the ROOVESP implementation.

The comparison is done at a general rather than the more narrow industrial requirements of Rover and the following aspects are discussed:

- naturalness in prototyping
- data abstraction

- modularity, extendability, consistency and uniformity
- efficiency during the system development and performance
- storage of graphics
- encapsulating the concept of time
- user interface

The above topics have been chosen to reflect the software needs of the complexities and the characteristics of the problem (ie. the Audit function and the PSC) which were discussed in the previous chapter. Some of them are:

- a. unclear interrelation of the various functions of the PSC
- b. unclear procedures / subjective information in the Audit function
- c. duplication and re-formatting of information
- d. dynamic environment
- e. fear of the new technology
- f. use of many hardcopy files

Data abstraction and naturalness in prototyping (for a, b and c) are vital in order to implement the system in a hierarchy of abstractions which is needed to overcome its complexity. Data abstraction in particular, helps give a clearer evaluation of the various formats in which the vehicle information appears resulting in the choice of the most suitable one for the development of the system.

Modularity, extendability, consistency and uniformity (for d) are important for the computerised development of the PSC, in particular which shows a highly dynamic nature.

The user interface and other facilities available from the software environment under consideration (for e and f) are important as a means of easing and accelerating prototypes of the system which in return increases confidence and communication with the people resulting in higher co-operation.

Finally, both the storage of graphics and the encapsulation of the concept of time are under constant consideration in the research: the need to store graphics in this specification documents in order to help the PSC was the main reason that launched the IPL project. The concept of time has been previously mentioned in that it is important to the PSC during the various build phases of the vehicle. In addition, performance and efficiency issues must be taken in account throughout the investigation of a software environment.

There follows further discussion on each of these topics.

7.6.1 naturalness in prototyping

All programming applications start with a conceptual model of the domain. In the object oriented software, these real world concepts are directly implemented with a class. This contrasts with the conventional structured programming where functions with numerous subroutines are coded to simulate the behaviour of the concepts. The complex interrelations between the functions and sometimes between their subroutines, often result in the whole system to bearing little resemblance to the concepts it supported at the beginning. In object oriented programming, the programmer focus on the *properties* of the objects being modelled which encourages the use of meaningful names, for the objects and their behaviour (functionality). Also, as the code is implemented by reflecting the domain being programmed, the development of the code becomes more natural and thus more rapid than using conventional programming.

7.6.1.1 dummy variables

In the relational database model the relationships among the tables are expressed by using the *join* operator on the keys of the tables. In the case that multiple values exist in a table and this table needs to relate with another one, a new table is created called the *intersection entity* of these tables which has as values in its fields unique pairs of the keys of the original tables. When the system grows and intersection entities have to be designed among multiple tables, multiple keys of tables have to be considered. In order to avoid dealing with multiple keys, database designers resort to the creation of unique identifiers

for each table. For instance, the fields `project_UID`, `part_UID`, `filament_UID`, etc. These identifiers are usually incremental files which are created during the software development of the database and initiate the value of a dummy field in the tables. This at least decreases naturalness in the design of the data base. In reality, the new version of ORACLE supports a mechanism which handles the automatic creation of unique identifiers and shows the close relationship of the relational model with dummy variables.

In OOP, however, the attributes of the object can be used as direct conceptual pointers to express relationships among objects which closely simulates the designer's thinking and hence increases naturalness.

In addition, such a representation of knowledge is more concise. As stated in [61] "often consiseness is the preferred metric for describing naturalness".

7.6.1.2 semantic overload

All the relationships between objects in the relational model are based on data **values**, and the designer must simulate pointers by comparing the values of identifiers in order to traverse from one relation to another (typically using the `join` operator). This limitation creates another problem: the relational databases become *semantically overloaded*. The term has been coined from the work of R. Hull and R. King [66] (and discussed in chapter 11 in

more detail).

Semantic overload in the relational database means that there is a limited choice of ways to express 'meaning' within the database. Meaning is usually expressed by relationships. In the relational model there are only two ways of expressing relationships: the fields within a table and the join operator.

On the other hand, relationships in the Object Oriented paradigm can be defined by many different primitives. In reality, all the research in OOP is concerned with the implementation of different primitive constructs of relationships ie. first strict hierarchy then multiple hierarchy, deterministic or dynamic (delegation in CommonOrbit [61]), as well as instances constructors, methods, entity's attribute types etc.

The broader choice of such primitives helps the design engineer to express the relationships on the way he conceptualises them, resulting in greater naturalness.

7.6.2 Data abstraction.

In relational data bases data abstraction is implemented through aggregation. As stated in the work of J. Miles and D. Smith [3] "an aggregation is an abstraction which allows a relationship between named objects to be thought of as a (higher-level) named object". For example, the table **Design-Feature-Group/ Vehicle-Part-Group** which expresses the relationship between the **Design Feature Group** and **Vehicle Part Group** tables - in the form of

uniquely identified pairs of values of feature groups and VPG codes - can be considered to be the aggregation of "*design-engineering-philosophy*".

The problem with the relational model is that although the new table represents abstraction in the database design, this is not the case regarding the data itself. Any time the programmer needs to retrieve information from this table he has to remember its internal structure. A hard job considering that this table may represent the aggregation of three or more tables. (This subject discussed in more detail in chapter 11).

On the other hand, in the Object Oriented Paradigm, methods can be attached to objects and called externally from any other function in the software development. As T. Rajan states in [64], "methods in Object Oriented Programming can be considered to be a communications protocol through which other programs can communicate with that object. This property is known as **data abstraction** ... The essence of data abstraction is that client programs need not concern themselves with the details of implementation or internal representations of the data within the object".

7.6.3 Modularity, Extendability, Consistency and Uniformity.

When a system is being developed to simulate an application area that changes rapidly, or when incomplete information exists for

the application and better understanding of the problem may create totally new alternatives, it is important that such changes can be implemented easily. The Rover specification problem falls to both these categories:

- Both the format of BAFC and the design logic in their compilation changes through the time and the same happens with the APN catalogue.

- The various departments that are involved in the Product Specification Concept maintain only a portion of the understanding of the problem which makes the initial knowledge in the design of the system incomplete and subjective.

Object Oriented programming offers the required type of **modularity** better than action oriented programming. For example, in object oriented programming it is easy to add new types of objects and extend the program, as it is only necessary to add new code, not to modify existing code. In action-oriented programming, the insertion of new types adds new cases in the generic function which must be updated ie. modified (see appendix 1). A detail discussion of this subject can be found in the work of K. Smedt [61].

As it will be proposed in chapter 10 the further development of the system can grow naturally, with the addition of new entity types to the data base such as **physical parts**, **fitment**, etc. Their behaviour can be defined independently of the rest of the objects in the database with individual methods associated with them.

Besides, the most sophisticated OOP environments such as FLAVORS or Statice, CommonOrbit [61], etc. provide internal mechanisms of *inferential integrity*. This means that if during the development of the system the designer changes the behaviour of the object then all the instances of the object are updated and that these changes are immediately accessible to the other objects sharing its behaviour (for example through inheritance). In Statice for example, the deletion of an instance of an entity from the data base would result in the automatic deletion of all references of this instance from the instances in the data base to which it was related.

In the relational example such actions must be 'coded' by the programmer. Remembering, as well, the *semantic overloading* of the relationships in the relational model this becomes difficult to maintain during the development phases. As R. Hull and R. King in [66] state "this is the reason that integrity constraints such as *key* and *inclusion dependencies* are commonly used in conjunction with the relational model. Although these do provide a more accurate representation of the data, they are typically expressed in a text base language; it is therefore difficult to comprehend their combined significance". Text base language means the tools that are usually used in relational databases to describe the design eg. in Oracle is CASE* methods, another more standard one is EXPRESS.

Finally, the modularity and uniformity in software development which the Object oriented style supports, makes easier programs

to maintain and expand (it discussed in more details in chapter 11).

7.6.4 efficiency during the system development and performance

Researchers in CAD/CAM design admit that relational data bases are slow comparatively to OODB. This is because of the nature of the CAD/CAM business: As stated in the work of D. Maier [76] "design operations aren't really single fetches and stores, but typically involve following a path to another entity or filling in a new object in a class." Consequently, performance increases if the data base offers facilities with which the designer can access directly only the local area of his concern each time.

OODB has been designed for interactive processes ie. the user is able to work with data from a specific small area of the data base, the data on the screen for example, whereas Relational data bases and other traditional data bases are designed such that the entire database must be on-line even if a very small section of this is required. Thus OODB design relates more closely to the PSC and Audit example as most of the time new parts are designed on the basis of old ones, consequently on updates of specific assembly structures (discussed in the following).

A detailed reference to the greater performance of OODB, comparatively to a relational one for other aspects such as locking or logging, concurrency control etc is covered in [76] and [58]. Especially in [58], R. Martin supports that only 10% of

the existing data-intensive applications (such as this Rover application is) have moved from the older database technology to the relational databases. The reason, he supports, is because the performance of the relational database is even slower than the hierarchical or network database technology of the past.

The indexing routines of the relational model are considered to be the most sophisticated and highly tuned of any class of software systems. OODBs and especially Statice support them, as well. Actually, R. Martin in [58] supports that sometimes, other database techniques for improving performance, such as clustering, could be superior in OODBs than the relational.

7.6.4.1 representing assembly structures

In chapter 4 the assembly concept (BOM) which represents one of the design components of the PSC in Rover was discussed. In a highly manufacturing oriented company such as Rover it is expected that it will be vital for the data base which would support its various applications be capable of maintaining hierarchical data structures (assemblies) and the complicated relationships between them efficiently.

The performance of the software tool under investigation becomes of the highest importance as the actual physical parts in manufacturing must be taken into account. Currently there exist 180,000 such parts recorded in Rover's PIMS database. More

importantly, in a CIE project where an image has to be tagged to the information of each of the data in the database in order to represent them in the outside user, this requires even more processing every time each of the data is manipulated (update of its usage statement, sourcing, quantity).

The direct access of one object to another within the computer memory, as it occurs with the OODBs is much faster than the relational model where the relationships have to be "flatten out" and retrieved by using the *join* operator. For further detail on this subject see the work of D. Maier in [76] and R. Martin in [58].

Lufthansa who uses Statice for its planning process, investigated ORACLE as a possible candidate in its feasibility and quotes: "firstly, the relational model is too "flat" to represent our complex data in a reasonable way. Secondly, we run into massive performance problems" [63].

7.6.4.2 efficiency during the system development

Considering the complex nature of the PSC that had to be tackled it became apparent in the beginning that OOP matches with the requirements of the efficient development of such a problem. For instance, it was not clear all the *object types* that may be used in the development of the program ie. objects for statistical analysis, a fitment object type or physical parts type objects.

In an action-oriented style, the number of object types will

increase the average search time because more cases will have to be considered (see APPENDIX 1 and the work of K. Smedt in [61]). In the object-oriented approach, such an increase has no effect because the information is associated directly with each type. This results in more efficient design and better testing capabilities during the development period and the final implementation. In addition, the new object types can smoothly interface with the rest of the program, instead of the relational approach which would need re-organisation of the data base, re-normalisation etc.

7.6.5 storing images

Image database management naturally leads itself to the paradigm of managing objects. The *presentation types* of the GENERA environment are discussed in appendix 1. Image database management represents the same philosophy. An object can have a way of *presenting* itself by some operations (methods) linked to it. Then the user can *think* of the presentation of the object as the object itself which makes data representation more natural. ie. the representation of a node with a circle.

As W. Grosky states in [73] "virtually all proposals from the database community for the management of nontextual information use object-oriented techniques".

Statice offers the facility to store permanently images as real values of an entity type attribute in the data base (raster arrays). The association of the rest of the textual data that is concerned with the image is direct as all belong to the same object. On the other hand, in the relational model images have to be stored in a separate database and then linked with the textual data by both a software intermediate (like Pacer) and a program 'linker' which has to be developed by the programmer. A complex task considering that more than one software environments are involved.

7.6.6 time

The variable of *time* is quite an important requirement for the selection of the data base which would support the PSC, as during the specification of a product, Rover keeps *timed* versions of the specification, currently duplicated in two different systems each time; AS (Applications System) and PIMS (see chapter 4). OODBs can more easily embed *time* in their design than relational models.

As stated in [6] "... another problem in studying the product variants in automobile is timing. The new product is designed before the specifications are completed. As a result of the information problems the number of variants of existing products is not known and neither are the distinctions of the different product variants. An investigation showed that 60% of the product

variants already existed in predecessor product".

T. Atwood who investigates the way how *time* variability could be implemented in a database model in [75] states that "in the relational models what the database returns from a query, is the most recent version of the object. This makes it impossible to get an older version, because the data manipulation language simply gives the application programmer no way of asking for one. ...The only option the programmer has is to circumvent the system - to roll the whole database back ..., copy the versions he wants out ..., then roll the database forwards. The programmer is forced to step out at least once, in the operating system... At worst the time involved in rolling the database backwards and forwards ...".

Cache values in Statice (other facilities exist for other OODBs) can keep an older version of the data and operations can be written easier to support *time* requirements. For further details in the subject the reader can refer to [3].

7.6.7 User Interface facilities

Graphics in Lisp-based workstations are implemented as mouse-sensitive icons in windows. The *presentation type* facility of GENERA which came out of the research into Object Oriented Programming [1] is referred into appendix 1. With presentation types, objects can be represented as icons and the user can use

all their information naturally by simply manipulating and working with their icons. As K. Smedt states in [61] "connotative information can be conveyed in graphical representations, such as the information conveyed to human users in the shape of objects".

Experience with the development of expert systems in AI has shown that one of the most difficult parts in their implementation is the acquisition of the required knowledge from the experts. The use of user friendly interfaces can facilitate knowledge acquisition as the user can view both the internal structure of the data and the decision process of the system and incrementally refine the program.

Additionally, already built-in facilities such as menus windows, icons (see *storing an image* section) and tree like graph of inheritance in Object Oriented Programming environments can facilitate early prototyping which in turn can encourage experimentation with many different approaches.

On the other hand, in the relational example, this would require the additional need of an external language like C, or a graphics environment such as GKS in the VAX system to interface with the ORACLE database or both.

In summary, the choice of software environment was reduced to two options either the relational approach with ORACLE and 4th Dimension from APPLE, and OOP and Statice from SYMBOLICS. The search was based on the characteristics and complexities of the

problem which were discussed in chapter 6. Those involved the unclear specification of the problem (ie. Audit function and PSC), the traditional human oriented environment which makes it difficult for people to co-operate with the new technology, the dynamic nature of the problem, the complex interrelationships among the vehicle parts existing in the form of experience and design engineering knowledge, the need to store images in the data base, the need to implement the concept of time in the PSC etc.

The search led to the same result as that of the two requirements of the problem ie. the design of the system in a "hierarchy of abstractions" and the development of automatic Meta knowledge programs to transfer knowledge expertise: the most suitable software environment is believed to be the Object Oriented Paradigm.

The great disadvantage of OOP (such as FLAVORS) is that they represent no real data bases. This is discussed in the following in more detail. Additionally, the existing OODB systems are new and consequently have not been thoroughly tested for their performance concerning their ability to support large industrial projects.

The following sections discuss the pragmatic needs which led to the selection of Statice, instead any other OOP environment (such as FLAVORS or Common Orbit) in order to support a realistic system for Rover.

FLAVORS is the standard object oriented data base environment in Lisp machines and a feasibility study of the Object Oriented Paradigm and OODBs always refers to this.

Statice is a new version of the existent FLAVORS environment in Lisp machines updated in the way in which the data are stored in the computer. The Statice characteristics are similar to those of FLAVORS.

Having decided to adopt the Object Oriented Paradigm, Statice was only later investigated in order to meet the pragmatic needs of ROOVESP ie. the implementation of a system supported by a real data base.

The gradual investigation and need for Statice as arised during the search is discussed in the following.

7.7 *Need for real OODB*

In the GENERA environment there are two ways of storing the data in the computer; the STRUCTURES as in conventional programming and FLAVORS. (A detailed reference to FLAVORS exists in appendix 1 and that to STRUCTURES environments can be found in [4].) The latter was chosen for two reasons:

Firstly, FLAVORS supports more facilities such as debugging mechanisms (FLAVOR EXAMINER), inferential integrity, etc. and is closer to the actual concept of the Object Oriented Programming

with multiple inheritance, methods inheritance, etc.

Secondly, by using FLAVORS one can more easily transfer the storage of the the data from the FLAVORS environment, in virtual memory, to a real Object Oriented Database such as Statice, on the disk.

The problem with the Object Oriented Paradigm in GENERA or generally with the Lisp workstations, at least until recently, is that the Object Oriented data base environments which are supported, exist only in the virtual memory of the computer. Data in the FLAVORS environment for example, although behaving as a real data base with complicated relationships among its data (objects), insertion, deletion etc, vanish when the machine is booted, or if the power failed as they exist only in the computer's virtual memory. To provide persistent storage, the user must arrange to copy the information out to files. In files, however, the interconnected structure of diverse objects must be flattened out, and somehow encoded into only simple bytes and characters. This encoding adds complexity to the application and takes time to run. As the files become large, saving out data and reading it back in becomes slow. Because information is being copied, the user has to worry about the copies becoming inconsistent. The benefits of object oriented programming are lost.

For this reason the literature investigation directed towards Object Oriented Data Base systems which would support in reality, as well, the merits of the Object Oriented Paradigm, as they

revealed previously.

7.7.1 REAL OODB AVAILABLE COMMERCIALY

Real Object-Oriented database systems are a new phenomenon, only recently emerging from the research world. There are only a few commercially released semantic or object oriented DBMSs for a knowledge engineer to choose from. These include: GemStone (from Servio-Logic) [59], SIM (from Unisys) [59], V-Base (from Ontologic) [78] and Statice (from Symbolics) [2].

The last OODB, Statice, which is a direct product of SYMBOLICS, was naturally considered as the first choice for developing ROOVESP's data bases, as it is fully compatible with the GENERA environment which were offered by SYMBOLICS to Rover during the demonstrations in the original IPL project investigation. However, the search continued in the case of better alternatives, concerning the cost, the performance and the compatibility.

7.7.1.1 Gemstone

Conceptually, GemStone is persistent Smalltalk. It runs on Vaxes, SUNs in a server request. It uses a programming language called

Opal which is Servio-Logic's version of Smalltalk-80. There are doubts as to its compatibility with the Lisp's FLAVORS environment and even integration were possible it is anticipated that it would be a difficult exercise. No referencies have been found that compares its performance favourably with that of Statice.

7.7.1.2 SIM

SIM is a full featured commercially robust semantic DBMS based on the D. Shipman's functional data base Daplex [60] and the Semantic Data Model [80], [84], [81]. Since Statice is also based on Daplex, the conceptual differences between Statice and SIM are quite trivial. However, the implementations differ greatly. SIM is much faster. Unfortunately, SIM runs only on Unisys mainframes. There is a choice of COBOL, Algol or Fortran for host language programming interfaces - no Lisp. The graphics it supports are at least primitive, whereas the hardware is very expensive (at least \$250,000 compared to \$10,000 per seat or \$50,000 per site for Statice). The vendors claim that SIM will be available under Unix in 1991 and will probably have a C language interface.

7.7.1.3 V-base

V-base is supplied by Ontologic. It supports full inheritance and persistent and shared objects. It is written in two languages: TDL the type definition or schema language and COP an Object-Oriented extended C. Its developers see favourable advantages in the clustering capabilities of the system to reduce access time on the disk [59]. It is not compatible with GENERA. Further details can be found in [70] and [59].

There are also as many university prototype systems that it is hard to keep track of them. Postgress from UCB (University of California) [65], Orion, Iris, Cactis from the University of Coloranto [67], [68] are some of the better known ones. Postgress is really an extended relational database whereas the others are true object-oriented. The reference of this can be found in [69].

Research which is generally confidential to the SYMBOLICS company [59] has benchmarked several of these prototypes and the report notices that in one case a prototype would need 3 years to load the team's data base before they could begin to benchmark it. The report concludes that at least two years of a commercial release are required to evaluate an OODB product. As it is stated in the report [59] "the way it is now (February 1990) the OODB world is as it was the relational database technology in 1981".

7.7.1.4 Statice

When Statice was investigated, was found to be FLAVORS compatible. It has been on the market long enough and regular benchmarks have run on it. There is documentation about some of the Statice known deficiencies [59] which is very important in order to be able to evaluate it and decide its feasibility in real life systems. The SYMBOLICS report [59] states that "it can be compared to the performance of the relational database systems 3 to 5 years ago and with careful design it can be used in real life systems".

In summary, Statice even after the investigation of other real OODBs, still seemed to be the preferable software tool to support ROOVESP, especially as the specifications were formed after the adoption of the GENERA environment for ROOVESP. Statice when initially investigated showed no hard limit on the number of entity types, the number of attributes in a type, or the length of names of entity types and attributes. Similarly, it showed no hard limit on the number of users per single server. In addition, it supported an SQL-like language for queries at the top level - the user can view the actual objects in the data base by entering queries upon the attribute values of the objects.

Finally, as real OODBs are a new phenomenon in the software world it is difficult to find referencies to such systems supporting real life problems and which consequently could be evaluated. However, reference to Statice being used in real life applications has been found in literature which highly indicated its feasibility to similar real life problems such as the

automation of the Audit function and the PSC in Rover. These systems are discussed in the following.

7.7.2 Other applications of Statice

There are only three real life systems for which the author found references to use the Statice OODB. Actually, only one of them it found to be fully implemented whereas the other systems are still under development.

Houston Power & Light is using Symbolics' Statice [62] to store and manipulate data for power-plant maintenance schedules. The data come from a mainframe payroll system. Each schedule contains 2,000 to 10,000 discrete activities.

Scheduling requires manipulation of teams of individual workers each having a unique set of skills, pay rates and work schedules; physical considerations such as distances from one plant to another; specific repairs required at each plant. Each power plant has a unique set of equipment, but each instance of that equipment shares many characteristics with others in its class and some pieces of equipment work together. Some of the attributes of the object express relationships with other independent object such as tools, parts, equipment, a particular class of worker is qualified to repair, and groups of items (such

as work teams). When an object is selected, the system loads the object ID (a unique number) and the object IDs of the other objects it has relationships with, according to the use the user application specifies.

The question is how the company can combine all these *permutations* to generate effective schedules? Houston P&L did it so effectively with Statice (plus an expert system) that overtime dropped by \$442,000 (38%). It used to take three hours to load a file to start scheduling one plant (out of 25), but it now takes five minutes because of the *semantic design* of the data base which results in the system only loading the specific object necessary. In addition, the expert system only restructures the relevant data instead of recompiling it all each time a change is made.

The *Houston Power & Light* project relates to the research of this thesis with the reduction of the permutations in scheduling.

Another company called Alcoa is experimenting with Statice [62] in order to manage information about the equipment and the processes in a *cold-steel rolling mill*. A future implementation under consideration by the company is the interface of Statice with an expert system for diagnosis.

At Lufthansa German Airlines there are currently two knowledge-based planning systems under development. Both of the systems are concerned with the development of the Lufthansa flight schedules

and one of them AMS (Aircraft Assignment and Maintenance Scheduling) is based on the Statice OODB.

Development of a flight schedule at Lufthansa starts approximately 4 to 5 years prior to the date of flight operation. Complex interrelationships exist between objects and/or events which form the appropriate environment for the application object oriented data base: to make sure that the flight schedule meets all the operational constraints such as maintenance requirements, station limitations, etc., to evaluate the schedule robustness with respect to unpredictable events and delays as well as integrate long term maintenance operations with the flight schedule. Lufthansa Airlines's intention it is not (yet) to substitute the planners capabilities with Statice and an expert system but rather to use Statice as an up-to-date database and use its *semantic* structure to provide as much as possible direct access and support to the planners in schedule evaluation. That is achieved partially by providing an appropriate user interface (presentation types) and partially by using a ruled based system for *statistical analysis* on the collected data which answers "schedule inquiries"

The AMS project started in September 89 and a feasibility study [59] for the software support of the system rejected relational databases - in reality ORACLE - because of their model nature being too "flat" which could not support the Lufthansa's complex data in a reasonable way. Germany Symbolics has undertaken the project together with Lufthansa developers and it is estimated that the production system should be available at the end of 1991.

The under development project of the *Lufthance German Airlines* relates to the search in this thesis by the means of the need of complicated relationships among the data and electronic support to the users as well as support from statistical analysis of the data. This is the case with the Audit function and the PSC where complicated relationships exist among the parts of a vehicle in the form of assembly structures and the need for support of the functions both with electronic equipment and statistical analysis to retrieve past knowledge.

In summary, Object Oriented Database systems are a new phenomenon in the software world. As a result there only few commercially available. Statice, coming from the implementation of FLAVORS in Lisp Machines, is one of them and there is at least one reference in literature for an existing real life system that uses it (Houston P&L). However, many more are under development (more than 60 units have been sold worldwide [62]).

Rover intends to be one of the pioneers in use of real life OODBs by automating the auditing function and the PSC using Statice accompanied by an expert system.

7.8 *knowledge representation in ROOVESP*

Having determined the advantages of OOP and OODB and the specific hardware/software environment, this section discusses from the software point of view, the reasons why other types of knowledge representation must be involved.

ROOVESP which will be discussed on the following chapter, represents a hybrid AI environment of the combination of object oriented, rule based, and procedural knowledge representation environments.

The Object Oriented environment is embedded in the system with the storage of the data in the Static data base (or Flavors) and the various operations that imply their organisation and retrieval.

The rule-based part of the system has been developed in the *inference engine* of the program, the 'experience' and heuristics which choose and combine the vehicle features.

Finally, procedural knowledge is implemented in the control structure of the program that drives sequentially the whole ROOVESP's process.

Although neither of those knowledge representations are efficient enough to tackle the problem in isolation, together they can harmoniously co-exist and constitute a powerful software environment.

OODBs implement mainly hierarchical structures of knowledge representation with the use of object types facilities and

inheritance - assemblies of a vehicle for example. In a pure rule-based system such a hierarchical knowledge representation, (see figure 41, 72) as well as the attributes of each object data type should be expressed as a set of rules of the form

```
(if (?x = Physical-Part) then (?x isa Part Description))  
                                (Inheritance),    or  
(If (?y = "seat-front-complete manual")      then  
                                (?y has VPG 1109AA))  
(assignment of the values of the attributes of each data type).
```

The number of the relationship rules increases almost exponentially to the depth of the hierarchical taxonomy. This can be the case of the implementation of large systems with a solely rule based knowledge representation: the rules of the decision-making of the system become overwhelmed by the rules which specify the domain itself. A reference to this can be found in the work of K. Hawley [61].

Practically, the implementation of ROOVESP by a purely rule based representation is impossible. There exist thousands of part descriptions each of them with at least 15 attributes, even more physical parts (180,000) which would need a huge number of rules to define the relationships among themselves. In addition, the knowledge of the domain in a purely ruled based system becomes more vague as definition domain rules and decision making rules are mixed in the same syntax. The program's consistency is affected, as well, and the maintenance of the system becomes a

nightmare for the programmer.

The adoption of an Object Oriented Data Base, to store and define the relationships of the data, eases the development process, as it distinguishes heuristics knowledge from concept hierarchy. The hierarchical relationships among the data having been defined abstractly with only few rules, can then be generated automatically from the inheritance mechanism of the environment. For performance aspects, it is crucial that only a few rules be checked by the system - the now distinguished heuristics - rather than a huge number of IF-THEN clauses.

On the other hand, the implementation of ROOVESP in a purely procedural knowledge representation would give disadvantages as far as *modularity* and *consistency* are concerned. As stated in [84] "in a procedural representation, the interaction between various facts is unavoidable because of the heuristics information itself". Consequently, the insertion of a new fact in the knowledge base may affect other pieces of code which should be updated as well. Besides, the complicated interactions make the *meaning* of the internal representation of the knowledge vague and a change may result in *side effects* in the consistency of the program which may not be seen immediately. Worst cases may require a huge re-programming of the old code or cause *incompleteness* in the decision mechanism of the system.

The embedding of the procedural knowledge for update of the data base, within OOP method, eases modularity. The rest of the procedures can more easily be tested for consistency and completeness as they only represent the **control structure** of the

system. The use of OOP, also, helps to perform *default reasoning* more efficiently than both production rules and procedural representation can implement.

Finally, notice that the use of production rules and procedural representation is vital for the system because both of them provide the *directness* which is needed for the system to reach the solution. Notice, as well, the smooth interface of all different representations of the knowledge domain (heuristics, procedures, Object oriented Databases) under a unique environment: GENERA.

7.9 *Summary on the chapter*

This chapter discussed the reasons that Object Oriented Programming was chosen to represent the software environment for ROOVESP, which automates the Audit function and the PSC.

The lack of clarity in the definition of the procedures of both the PSC and the Audit function suggested design of such a system in a "hierarchy of abstractions", adopting *almost-hierarchical decompositions* in its subsystems and gradually entering details in the form of *constraints*. The linkage of the subsystems in the later stages of the design could be defined by using constraints, as well.

Search on existing systems in the AI field which implement hierarchical planning indicated the use of an OOP environment. The same environment was indicated after the research for the implementation of automatic Meta knowledge mechanisms, which investigation showed that is needed in the system.

However, when Rover launched a new project, IPL, and a decision had to be made between the Relational and OOP alternative insufficient research had been done to justify the risk of using OOP. A research in the existent software literature was carried out to direct compare the Object Oriented Paradigm with the Relational example. The comparison was made on the basis of the characteristics of the Audit function and the PSC in Rover and it was concluded that OOP was the most suitable software environment for their automation.

An investigation of real OODBs which could support ROOVESP in a real life application then led to SYMBOLICS' Statice being chosen.

Finally, performance and programming issues led to the implementation of ROOVESP as a hybrid system which involves OOP and OODB, procedural and heuristic knowledge representation.

The two following chapters discuss the ROOVESP's implementation.

8 ROOVESP (Rover's Object Oriented VEHICLE Specification
(THE FIRST PHASE OF THE AUDIT FUNCTION)

This chapter considers the first phase of the audit function (as defined in chapter 5) which represents the core of the whole audit process. The original objective of this project was the implementation of a computer algorithm which would validate parts's Usage Statements coming from Component engineering. However, research showed that such a function (ie. a validation procedure which embeds auditing knowledge) cannot be isolated from the rest of the vehicle specification processes as all of them are linked together with design engineering and manufacturing rules. Consequently, the objective of the project was extended to cover at least the first phase, including the acquisition of the most general design engineering knowledge.

The implementation of a computerised system which not only audits usage statements of vehicle parts but virtually generates them is discussed in the next two sections of this chapter. The first section introduces the components of the system and their software specification whereas the second section details their functionality. In particular, the second section discusses the further needs which led to the change of the original project objective and the need for expansion of the system outside the scope of the Auditing department, to the Specification Services of the company. Chapter 9 discusses the automation of the audit

phases 2 and 3 which is currently under development. For the rest of this chapter the term *audit process* will be taken to mean only the first phase of the overall audit function.

The last section of this chapter describes further the functionality of the system and especially examines its Meta-knowledge component and testing of the system.

contribution to knowledge:

- Tightening up the existing conceptualism of the PSC in Rover (ie. no duplication of the information, possible applications of data extraction, correct compilation of the BAFC document, algorithmisation of the various audit procedures etc.).
- The introduction of new concepts in the PSC in order to tackle the problems which have appeared overtime.
- Proposal of the use of the Object Oriented Paradigm in the PSC to enable domain knowledge to be represented in the most natural manner.
- Using historical data to acquire Design Engineering knowledge. Mainly because of the revived PSC conceptualism and the way in which the existing knowledge is represented.
- The creation of an almost self-contained automatic auditing function in Rover.

8.1 INTRODUCTION TO THE SYSTEM DESIGN

It was mentioned in chapter 5, during the description of the manual audit function, that the way in which the audit function is presented does not truly reflect the way in which it currently operates within the company. Instead, it has been enhanced in a structural way. In this way, the structure of the audit process mainly represents the design of the system itself. Thus, the automated system right from its conceptual implementation eliminated the deficiencies of the existing manual audit function ie. duplicate information, subjective audit inferences, etc. It makes use only of the minimum required input and follows a well defined searching route that guarantees solution. In a further step, the system re-organises some part of the original information to achieve optimality.

8.1.1 DUPLICATE INFORMATION

8.1.1.1 Features List

The system considers some of the application tools discussed in chapters 4 and 5 redundant and consequently does not use them.

One of these is the Features List of the product. Instead of this document, it uses the Base and Additional Features Charts of

the vehicle. This decision was made on the basis of software engineering. That is, the format of the Features List (see figures 26, 27) is highly human oriented as much of the document is written in English, and follows a reasonably well structured english grammar and syntax. Take for example figure 26 Features List, there is the description "manual slide, recline, lumbar adjust, height 1". This description is written in relatively clear English grammar and syntax. The equivalent from the AFC document (figure 31) is:

"front seats reclining (rcseat)"

"ft seats recl lum & dr ht adj (lumbar)".

It would be difficult, if not impossible, to conclude from a data matching exercise that these pieces of information from two different documents are the same. It is not clear that "recl" matches "reclining" or "ft" matches "front" and there is no consistency in these abbreviations. Another solution might be the double cross linkage of the Features List with the Base and Additional Features Charts, using a *string matching* program. However, that has been proved unsuccessful as *common language of discourse* for both documents does not exist.

For example, in figure 26 the Features List of the Rover R8 expresses three options on the front seats of the product, that differ in their operation:

"manual slide and recline",

"manual slide, recline, lumbar adjust",

"manual slide, recline, lumbar adjust, height 1".

The equivalent AFC's information, compiles only two features for the operation on the seat:

"front seats reclining (rcseat)"

"ft seats recl lum & dr ht adj (lumbar)".

The same inconsistency exists for the material of the seats. It is apparent that the two documents could not be reliably linked automatically. Though the feature descriptions in the AFC have discipline, the english expressions of the Features List change subject to the way the Vehicle Directorate compile them. Besides, even the index reference numbers of the two documents do not match. ie. "B51", "B44".

However, the greatest problems arise when a Features List description expresses vehicle characteristics which belong to two different feature groups of the BAFC of the product. For example the Feature List's descriptions:

Height adjustable/tilting head rests trimmed in:

- seat facing material
- doeskin
- leather/PVC

actually refers to features of two different feature groups: the "B44: seats -reclining" and "B51 or B40: seat face material" of the AFC document. In consequence, a one to one link of Features List's descriptions to features compiled in the Additional Features Charts, is not feasible.

An additional problem is that there is a lack of discipline in

the storage of a Features List document to a backup tape. It only follows a loose pattern of specified records in order to be printed later to an A4 paper of a special format. The parsing of such a tape would not be reliable.

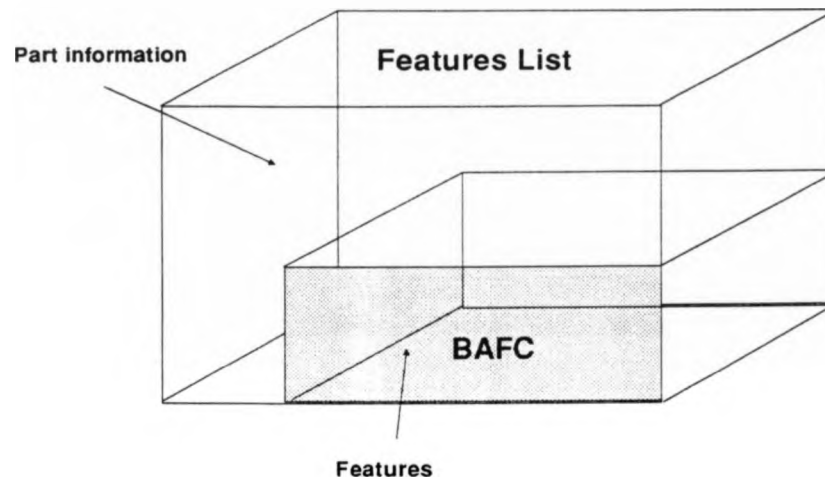


Figure 50: The intersection of BAFCs with the Features List of the model

The question, consequently, arises: If it is not feasible to use the Features List, how can the system access such information which it is so important for its application? Naturally, an answer at a first glance is the Base and Addition Features Charts which contains similar information. The problem, however, is that the Features List document keeps information for both the features and the parts of the car (see figure 50). In consequence, a high percentage of information may be missed by adopting this solution. Unfortunately, however, there are no

other alternatives.

The deadlock may be overcome eventually, when it is faced within the overall design of the system as a vehicle specification package rather than just audit process of specific input and output. The solution in this case, is the implementation of alternatives such as similar information relevant to the missing parts being retrieved electronically from other documents (such as EJA18V, discussed later) and when appropriately analysed, this part information could be linked to features which the system knows to use. This information gained, it would represent a part of the design engineering missing from the company. For example, it could be found (from analysis) that part information such as "key operated boot lock", "plastic sleeves over boot latch", "rear spoiler, black" etc. is linked to the vehicle feature "Rear Badge" (discussed in chapter 4, section 4.3.3). The details of such a process are discussed in the next section.

8.1.1.2 Territory Code Index Report and Model Summary Chart.

Both the Territory Code Index Report and Model Summary Chart documents are not used for the system. The Territory Code Index Report is redundant as the system uses double links to refer from supergroups to groups, to countries and in reverse. The information of the Model Summary Chart is expressed in the Base Features Charts more precisely, and makes therefore the use of the former redundant.

8.1.2 SYSTEM'S COMPONENTS.

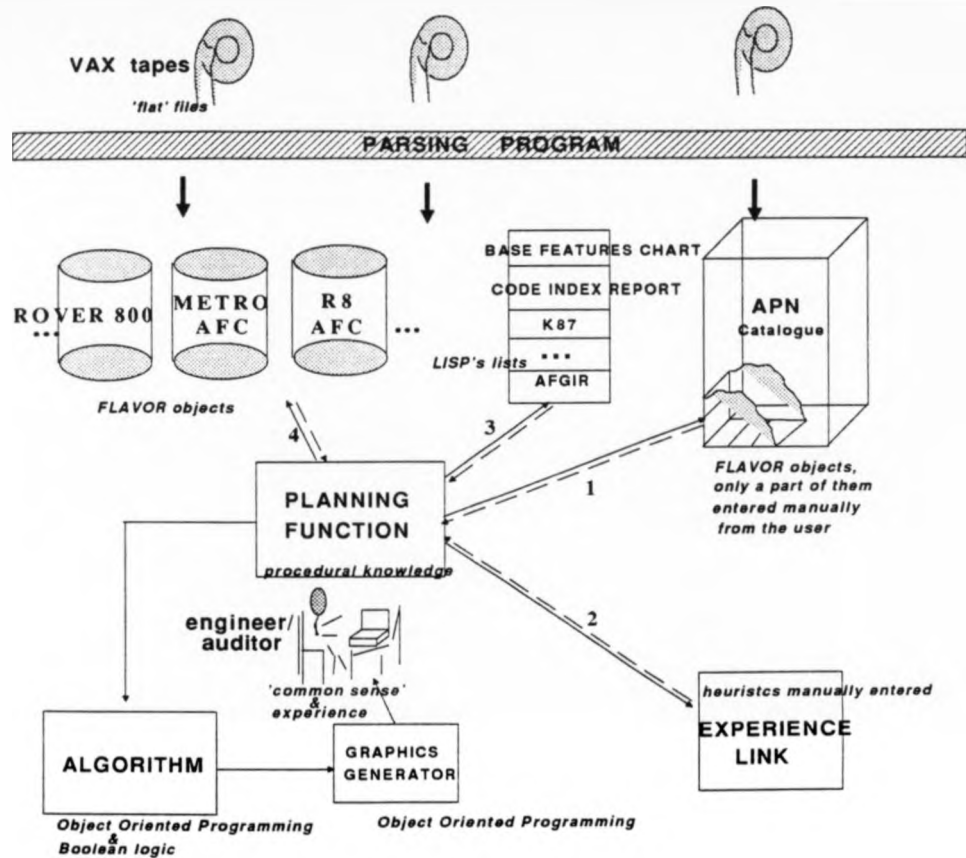
The basic components of the system represent the tools of the auditor as discussed in the previous chapter, together with some supplementary parts which either help the electronic transfer of the audit hardcopy documents into the computer or code the knowledge and experience within the auditing function.

All the components of the system are shown in figure 51.

All the information needed for the audit process is stored in the memory of the computer through the application of parsing programs to raw text data. Temporary object oriented databases are created and the audit process starts. The part description of a physical part is chosen from the APN catalogue. The auditor from experience links some feature groups with this part description in order to carry out his application. However, when questioned by the system to indicate the vehicle model for which this part has been designed, the resulting combination will eliminate the available feature groups as the process passes through the AFGIR. In sequence, the validation algorithm is invoked which works at two levels:

(i) planning function: It combines all the features of the available feature groups in order to create part specifications, whereas when it needs consultancy from the auditor/engineer queries are made in understandable English, by translating the source code with the help of the existent documents (K87, Base Feature Code Index Report, Territory Code Index Report etc).

(ii) validation procedure: it works out and graphically



The italics indicate the way in which the knowledge is represented in the various components of the system

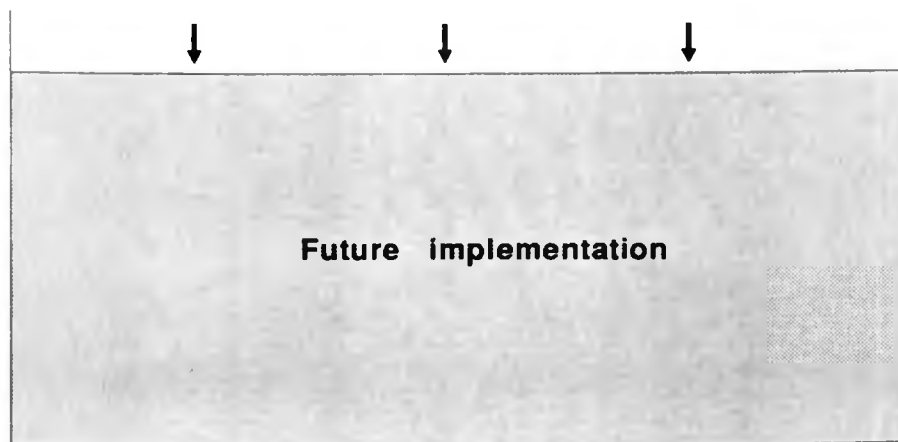


Figure 51: The first development phase of ROOVESP

outputs all the legal possible feature combinations for the specification of the part.

8.1.2.1 parsing program

The ISTEEL company implemented Rover's existing specification system and it is the main source of software support and maintenance. ISTEEL, consequently creates and keeps backup tapes of all the existing audit documents for all different systems.

The parsing program of the proposed computerised system uses mainly VAX tapes maintained by the Business System department of the company (BAFCs, APN, K87, etc.) to create its input. The parsing of all these tapes is done according to the specifications given by ISTEEL. The main function of the program, however, is independent of changes in the format specification of the tapes. Such a need arises from the dynamic nature of Specification Services. The format of the BAFCs and APN catalogue, for example, have been changed twice during the last year. The rest of this section covers the parsing of the BAFCs, which represent a typical example for the parsing of the rest of the data base. The result is a data base of records representing the features of the car and their restrictions. Fortunately, the BAFC shows a high degree of discipline in its electronic compilation from ISTEEL, hence can be used reliably by a computer application. The parsing of the restrictions of the features from the tape is based on tables of sequential exchanges of boolean

Q1	C1	Q2	C2	Q3	C3	Q4	C4	Q5
Q6	C5	Q7	C6	Q8	C7	Q9	C8	Q10
Q11	C9	Q12	C9	Q13	C10	Q14	C11	Q15
Q16	C12	Q17	C13	Q18	C14	Q19	C15	Q20
Q21	C16	Q22	C17	Q23	C18	Q24	C19	Q25

Figure 52a: Territory/Combination restrictions

TERGRP1	TERGRP2	TERGRP3	Territory code except 1	Territory code except 2	Territory code except 3	TQ1
TERGRP4	Territory code except 4	TQ2
...			...			TQ3
						⋮

Figure 52b,c: more specialised territory/combination restrictions

qualifiers and Rover codes, as shown in figure 52a.

The figures 52b, 52c show the way in which the territory restrictions are compiled in the tape in sequential fields of territory groups and countries exceptions.

The parsing of the BAFC represents a typical AI exercise in string-matching in combination with heuristic rules. The most difficult part is to assign to each restriction its qualifier and

distinguish between several lines of restrictions or a single one. For example, if both the Q5 and Q6 qualifiers in figure 52a contain a value it would mean that the restriction expression of the feature is continuing in the next line of the table following the original *parsing pattern*, whereas no value on the Q5 qualifier would mean that an additional restriction expression is added to the availability of the feature and a different *parsing pattern* needs to be applied (for more details see appendix 3).

8.1.2.2 Automatic Part Numbering catalogue

When the engineer has completed the design of a physical part of the vehicle, he consults the APN catalogue, from where he picks up a part description that matches the english representation of the part. eg. "seat-front complete manual-passenger". Consequently, it seems logical for the computer system to start its application from the APN catalogue, as well.

As mentioned at the beginning of this chapter the system re-organises pieces of information according to its software development needs. This is the case with the APN catalogue. The whole design of the system is built around the *behavioural* aspects of the parts of a vehicle - actually the part descriptions. That is, the way a usage condition (referenced to a specific part) is compiled, updated and made to co-exist with the rest of the usages in the data base. In this perspective, a part description from the APN catalogue is considered to be an

individual entity in the system that contains all information relating to its identity. Following this logic, the part descriptions of the APN catalogue have been implemented as objects rather than conventional datatypes.

More specifically, the part descriptions of the system have been represented as FLAVOR objects within the Lisp Machine and only a representative sample of them have been input manually during the development.

8.1.2.3 Additional Features Chart

In addition to the part descriptions from the APN catalogue, the Additional Features Chart of each model is implemented as a series of FLAVOR objects within the system. In this case, each feature of the model represents an object with knowledge of its availability and restrictions to the vehicle. The specification of the part then becomes conceptually clearer being represented in terms of feature objects which control the fitment of the part into the vehicle.

8.1.2.4 The algorithm

The validation algorithm uses object oriented programming to manipulate the overall knowledge represented in the system. It is the control structure of the system and works on two levels:

firstly as a planner and secondly as a validation procedure. As a planner, it collects the information relevant to the application from the engineer and/or auditor and returns relevant knowledge it possesses or feature mismatches that it found. It organises feature objects in hierarchical structures to meet the needs of the specification exercise and simultaneously maintains the information of the part description object under consideration (retrieve, delete, update).

As a validation procedure, it combines the availability of the feature objects and generates specification usages in Boolean form.

The nature of the knowledge it possesses is represented in the computer memory both in procedural form and production rules, and interfaced with the FLAVORS object oriented programming.

8.1.2.5 experience and knowledge link

Originally, this information is fed to the computer manually as a sequence of production rules. It represents the design engineering knowledge that determines the design engineering features which control the fitment of a part in the vehicle. This information does not currently exist in a documented form anywhere within Rover. Instead, it is knowledge and experience held by engineers, auditors and Specification Services people.

8.1.2.6 the remaining of the components of the system

The Base Features Chart, Territory Code Index Report, Territory Group Index Report, Base Features Group Index Report and K87 exist in the computer as conventional data structures: lists of strings, or atoms or nested lists. The system can therefore use them as they are stored with no further processing.

The following section of the chapter will discuss in more detail the main components of the system, such as the algorithm, the experience link and the APN catalogue.

8.2 DETAILED DESCRIPTION OF THE SYSTEM'S COMPONENTS.

Before the details of the main components of the system are described, it is necessary to outline certain peculiarities of the audit function which were discovered after extensive investigation of the problem. Those peculiarities not only affect the conceptual design of the overall system and require the introduction of new terminology but also affect the software solutions adopted. As mentioned earlier, part descriptions as chosen from the APN catalogue, drive the whole specification process within the computer system.

The difficulty arises from the fact that each part description

has to be viewed within the system from two different perspectives. Firstly, as a logical entity which abstractly represents a number of similar parts of the vehicle and secondly as each individual physical part of the vehicle. In the first case, each part description is linked with the design engineering features which control the fitment of physical parts of this type, whereas in the second case the actual assembly structure into which this part description (the specific physical part) fits, is assigned its dependency features. For the rest of the chapters, except where otherwise specified, the term "part description" will refer to the logical entity part description rather than the physical parts and the design of the computer audit system is built around this entity. That is, the relationship between the part description at the highest abstract level, the Rover design engineering features, and the restrictions of those features.

In that perspective, each part description can be viewed as an independent object of identical behaviour. It is at this point, however, that the peculiarities of the formalism of the existing audit function arise: part descriptions (or parts) are fitted to different vehicle models with different specification conditions. That means, there is no clear mapping between part descriptions and vehicle features (actually feature groups) and the implementation of such relationships would not be feasible under the current vehicle specification logic. For this reason two new vehicle feature concepts called the "musts" and "maybes" (or optional) feature groups have been invented for the new system implementation.

	(C) H05.1 HAVE	(C) IF	(C) AND	PAGE	HWP
(C) H05.3 NOT HAVE	(C) IF	(C) AND			
(C) H05.3 NOT HAVE TO	(C) IF	(C) AND			B5.3
(C) PACKAGE WITH	(C) OR	(C) OR		END	
(C) THCI ONE.5	(C)	* COMMENTS *			

REV	NO	REV	DATE	REVISION DESCRIPTION
5	31	AUG	09	REMARKS ADDED TO QUALITY THAT THIS FEATURE MUST BE USED WITH OSI GROUP

For example, the feature group "B40 or B51 : SEAT FACE MATERIAL" is a must feature group for the specification of a part description such as : "seat-front complete manual", as the seats of the cars must always have some kind of material at their finished stage. On the other hand, the feature group "B44: SEAT RECLINING" it is a maybe (or optional) feature group as its application to the specification of the seat depends of the design intentions of the engineer. Besides, investigation into the AFC has proved that the must and optional feature group concepts do exist vaguely within the document, sometimes compiled by specification services in the form of comments under combinations restrictions. For example, the definition of the "B53: SEAT SIDE ROLL" feature group in the AFC of the Rover R8 model (figure 53) comments (c) on its combination-restrictions field that its specification is controlled by that of the "B51: SEAT FACE MATERIAL" feature group. Thus it might be viewed as an indirect indication to a must feature group.

In addition, inconsistencies of the current manual system have led into the introduction of one more vehicle feature concept, the negative feature. This is, because of the inefficiency of the existing system in specifying the usage statements of the parts correctly and in flexibly updating them, as well as controlling the parts effectivity (see chapter 10). For these reasons, the specification people have introduced feature description into part descriptions. The aim is to help engineers and/or auditors to fully understand the parts they specify and/or design. In order to explain further with the negative feature group

terminology two examples are given.

There are many different types of front seats in a vehicle, such as:

"seat-front complete electric-passenger",
"seat-front complete manual",
"seat-front complete manual-driver" etc.,

though there should be only one part description, where its usage statement would discriminate its fitment each time (figure 45). It is the same case with the mirrors of the vehicle. There are at least 40 different types of mirrors (figure 54) such as

"mirror assy-exterior r/c body colour",
"mirror assy-exterior r/c self-colour",
"mirror assy-ext-driver r/c self-colour",
"mirror assy-ext-passr e/c-less-cover" etc.

In both instances, seats and mirrors, design engineering features are entered in the part description: the operation of the seat (electric, manual), the colour of the mirror (body colour), respectively. This creates confusion as the vehicle features compiled from Specification services show similar **specialisation**. For example, in the Additional Feature Chart of the Metro model (figures 55, 56) there are design features with description:

"B12C: BODY COL MIRROR -STANDARD HEAD (PANFIN)"
"B15K: DRIVER'S DOOR MIRROR (DRVMIR)"

The design feature "B12C: BODY COL MIRROR -STANDARD HEAD

PART	BRAND	PRIMARY	SECONDARY	DIST
QTY	U/L	IN-DESCRIPTION	IN-DESCRIPTION	IN-SP
WPC 1100CA BACKLIGHT GLASS FRAME AND FITTINGS				
CPU		WATER DRINKING	BACKLIGHT SHIELD	ANTHRA
CPU		INSULATING LENS	BACKLIGHT GLASS	ANTHRA
CPU		GLASS ASSEY BACKLIGHT	CLAMP	ANTHRA
CPU	737000	GLASS ASSEY BACKLIGHT	CLAMP	ANTHRA
CPU	737100	GLASS ASSEY BACKLIGHT	CLAMP	ANTHRA
CPU	737200	GLASS ASSEY BACKLIGHT	CLAMP	ANTHRA
CPU	737300	GLASS ASSEY BACKLIGHT	CLAMP	ANTHRA
CPU		GLASS BACKLIGHT	CLAMP	ANTHRA
CPU		GLASS BACKLIGHT	CLAMP	ANTHRA
CPU	73711	GLASS BACKLIGHT	CLAMP	ANTHRA
CPU		GLASS BACKLIGHT	CLAMP	ANTHRA
CPU		WATER DRINKING	BACKLIGHT	ANTHRA
CPU	73727	SMALLER GLAZING	BACKLIGHT	ANTHRA
CPU		EMERGENCY ALARM B/L LIGHT	CLAMP	ANTHRA
CPU		EMERGENCY ALARM B/L LIGHT	CLAMP	ANTHRA
CPU	000000	SILENT TONE THERMIST	BACKLIGHT	ANTHRA
WPC 1100CA BLU AND YELM FITTINGS (FIBERGLASS & THERMIST)				
YELM		INSTA. BLOC	WATERDRINK	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM BODY COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM		WATERDRINK ASSEY ELI	YELM-11555 COIN	ANTHRA
YELM	76700	WATERDRINK ASSE		

192

(PANFIN)" can be called negative to the part description "mirror assy-exterior r/c self-colour" because the same mirror assembly cannot logically be both body colour and self colour. Similarly, the "B15K: DRIVER'S DOOR MIRROR (DRVMIR)" design feature is negative to the "mirror assy-ext-passr e/c-less-cover" part description because the mirror cannot be for both driver and passenger's door.

In summary, the system is designed around the behaviour of abstract part descriptions entities, more specifically, their relationships with design engineering features that may control their fitment into the vehicle.

The need for a formal implementation of this idea has resulted in the introduction of new concepts and terminology within the audit and/or specification environment: must and maybe (or optional) feature groups and negative features with reference to the part description of the vehicle.

8.2.1 THE VALIDATION ALGORITHM

(first development phase of the system)

The validation algorithm represents the core of the computer audit system. It was originally named so because of the objective to validate engineer's usage statements. However, during the

```

NEW NO  REV DATE  A  R  E  V  I  S  I  O  N  D  E  S  C  R  I  P  T  I  O  N
V  00  DEC  08  PPL M21P/0B-----RECORD 012C,430 RAISED
                                TO "P" LEVEL
  
```

194

AUSTIN ROVER									
PRODUCT ENGINEERING									
ADDITIONAL FEATURE CHART									
MODEL RANGE METHOD 2									
FEATURE NO. CODE (FEATURE WORD)	AVAIL C S E M A N	USAGE C B D E F P H M S V N F V H L	B15 - EXTERIOR MIRRORS		GROUPS	TERRITORY RESTRICTIONS	DATE	DESIGN REF. AUTHORITY EFFECT SIB REFERENCE	
			COMBINATIONS- RESTRICTIONS	RESTRICTIONS				POINT	IN IN OUT OUT
4001810J DOOR MIRRORS - HAND SET (2HSBTR)	S	XP						80270	P M/30P/05
410	U	XP			1200	1200 +00		80270	P M/30P/05
420	U	XP			1200	1200 +00		80270	P M/30P/05
430	IXP	P H			1200	1200 +00		80270	P M/30P/05
440	IXP	D W			1200	1200 +00		80270	P M/30P/05
450	IXP	H M			1200	1200 +00		80270	P M/30P/05
460	S	XP			1200	1200 +00		80270	P M/30P/05
4001810K DRIVER'S DOOR MIRROR (DRMIR)	S	XP						80270	P M/30P/05
410	U	XP						80270	P M/30P/05
4001815B DRW MIRR MENDIE PASS PLACD (DRCPFA)	S	XP						80270	P M/30P/05
4001815D DOOR MIRRORS - REMOTE CONTROL (2HCHMR)	S	XP						80270	P M/30P/05
410	S	XP						80270	P M/30P/05
420	U	XP						80270	P M/30P/05
430	S	XP						80270	P M/30P/05
440	U	XP						80270	P M/30P/05

Figure 56: The "B15": "exterior mirrors" feature group for Metro.

development of the system it has expanded to administrate the scope of its appliance, as well. For this reason it actually operates on two phases instead of the one its name implies.

In the first phase, it plans the route to the solution by successively building up a hierarchical research space. In the second stage it introduces finer levels of detail for each node in the hierarchy and creates a combinatorial exercise which it solves later.

8.2.1.1 first phase (Planning process)

The system collects all the information for the features that concern the part description in question by accessing the feature attributes of that instance object. This information will determine the must and maybe feature groups associated with its intended specification. In sequence, the part description is linked with the vehicle model that it has been designed for. This combination will consult the AFGIR and will authorise only those features applicable to the vehicle model to pass to the next processing step. The must feature groups which pass the validation process continue by default towards the final specification of the part, whereas the system questions the user regarding the valid maybes (or optional) feature groups to determine whether they meet his design intentions.

The route of specification continues with the system checking the feature group specification inconsistencies within the data base

itself. For example, feature groups which refer to the same logical characteristic of the vehicle but appear more than once. eg. the feature groups "B51: seat face material" and "B40: seat face material". The specification people created the supplementary B51 feature group because the index references of features from A to Z ,for the material of the seat were filled up in the B40 feature group. It is not correct for identical design feature groups to be used in the specification of the part. For example, a usage statement such as "+ B40F + B51K" is logically incorrect as a seat of a car can only have one material when is finished, leather (B40F) or zenith (B51K) but not both.

At this point, where the feature groups (musts and maybes) of the specification have been established, the system searches through the AFC database to retrieve all the available features offered by Rover and referenced to those feature groups. Notice it picks up only those with live effectivity ie. the features which have not been deleted from specification services. In parallel, the system accesses the part description object under consideration and recovers the negative feature information known for this object. Both inputs are then combined to eliminate further illogical situations.

The engineer is then prompted to provide the remaining feature availabilities as negative features to the part description. Every positive answer from the engineer will result to the update of the part description object's attribute that keeps the information on the negative features. In this way, the engineer trains the system with regard to the negative features and the

part description, such that the system *remembers* those and does not ask the same questions in future regardless of the model specification. After a reasonable amount of training, the system should be capable of storing all the information and need not question the engineer.

After the part description negative features have been excluded from the scope of the application, the computer system continues its processing by translating all feature availabilities from conventional data (Lisp lists) already taken from the VAX files to FLAVOR objects. These objects are instances of the *feature-chart-entry* class. A *feature-chart-entry* class represents a single feature availability in combination with its restrictions as compiled by Specification services. Schematically, it can be seen as a single line of information across any page of the AFC (figures 31, 32, 33). Notice this is the way the data arrives from the VAX tape.

Once the information relevant to the application has been translated to "Flavor" objects, each of these objects implements attributes equivalent to the column headings of a page in the AFC hardcopy file. ie. feature code, feature description, class, ... territory restrictions, design effect point etc. Such a knowledge representation is flexible as methods can be applied to the *feature-chart-entry* objects and their behaviour can be made to simulate that of the human auditors.

For example, a specific method in the program taxonomises the *feature-chart-entry* objects according to the feature group to which they belong. This results in a single page layout of the

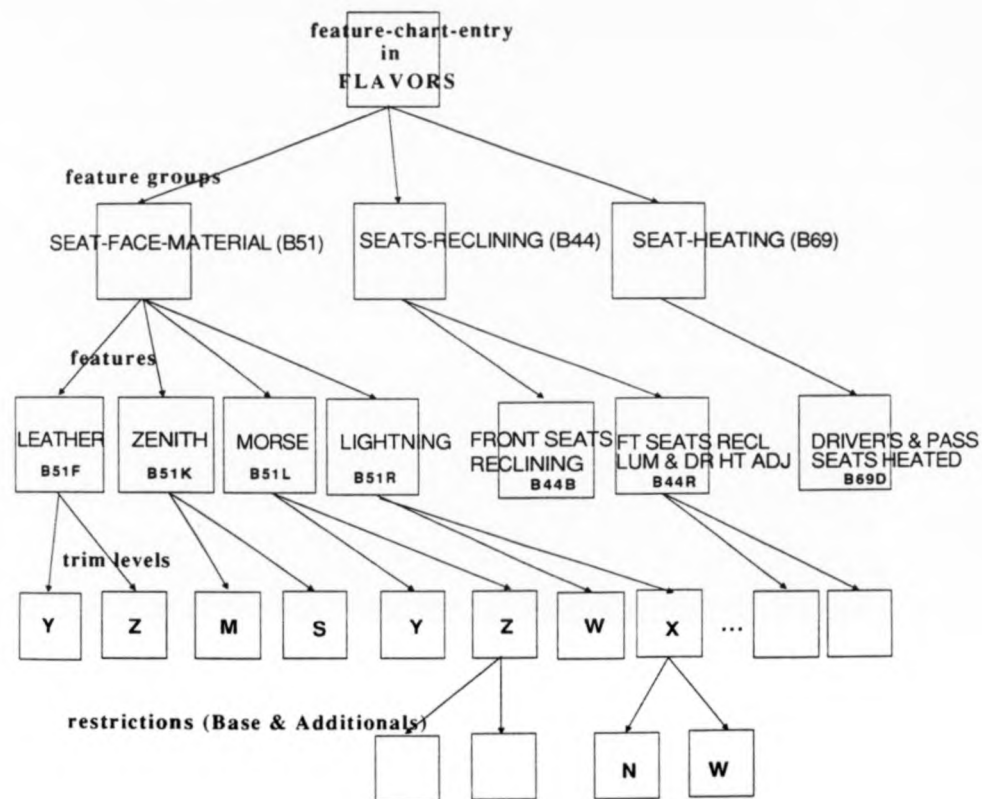


Figure 57: hierarchical representation of the feature-chart-entries in FLAVORS
(a nested hash table)

Additional Features Chart hardcopy file and naturally helps the knowledge engineer to work in the same way as an auditor. Another method groups the feature-chart-entry objects by the trim levels (classes) of their availability. Such data organisation, serves another system design need; the application of the audit procedure.

In reality, the feature-chart-entry objects relating to the

application are structured by the system in a hierarchy of abstraction levels: feature groups, feature descriptions and trim levels (figure 57).

Inside the memory of the computer the feature-chart-entry objects are represented as a nested hash table [79] of instances of the feature-chart-entry "Flavor". The keys of the table are the attributes of these objects: the feature code, the feature description and the trim level. The organisation of the data in a hash table is chosen because it represents the fastest available searching method. The disadvantage of space requirements is insignificant as the planning process of the system has already reduced drastically the number of objects.

The system at this stage has organised all the information (features-chart-entry objects) in such a way that it can be fed into the validation algorithm and create the usage statements of the part. However, additional processing may be required to smoothly complete the application of the validation algorithm.

8.2.1.1.1 ordering of feature groups

Although the validation algorithm can start its function by choosing any feature group as the reference basis for its combinatorial process, the correct selection of the Master (or pivot) feature group can ease the process.

It is more possible for the feature group with the most feature

availabilities to cover all the trim levels of the model and consequently guide the search fully, rather than choosing a feature group with only a few of the trim levels of the model and result to additional speculation and processing of the missing information.

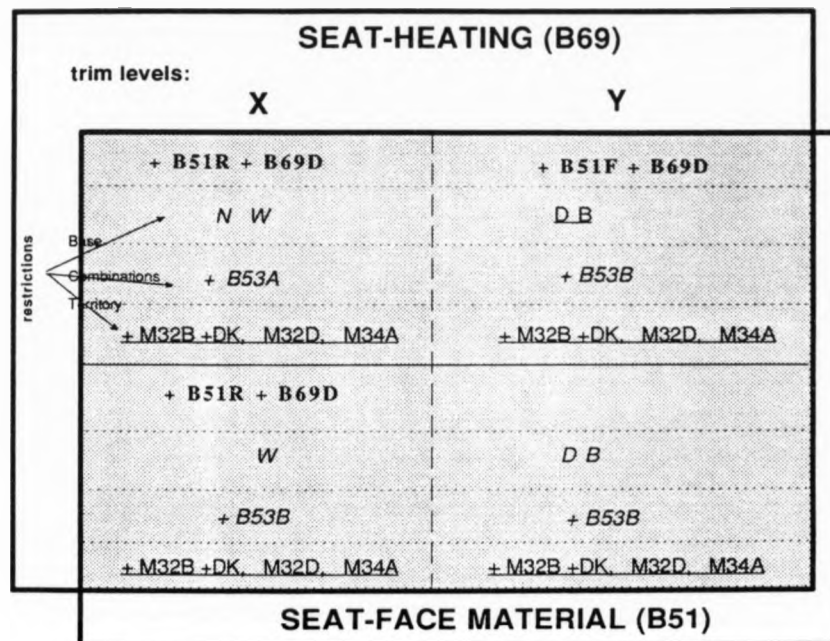
For this reason the system orders the possible candidate feature groups in priority sequence before it audits. The measure of ordering is implemented on two principles:

- a. must feature groups precede maybe feature groups
- b. feature groups with the larger number of feature availabilities precede those with fewer.

For instance, in the manual audited exercise of chapter 5 the sequence of the feature groups starting from the higher in the order would be "B51: seat face material" (must feature group with 4 feature availabilities), "B44: seat reclining" (must feature group with 2 feature availabilities), "B69: seat heating" (maybe feature group with 1 feature availability). In sequence, the **Master** feature group is detected - in this case "B51: seat face

material" which is combined in pairs with the rest of the feature groups consulting the priority sequence established from the system. eg. (B51, B44), (B51, B69).

8.2.1.2 second phase (application of the validation algorithm)



The restrictions from the heating group (B69) represented in italics

The restrictions from the seat-face material group (B51) represented in underline

Figure 59: The second step of the validation algorithm

8.2.1.2.1 combination of features

In order to demonstrate the combinatorial procedure, the manual example for the specification of the front seats of the Rover R8 model as discussed in chapter 5. is used as illustrated here, as well.

The feature-chart-entry "Flavor" objects of the master feature

Trim levels of the Master (or pivot) feature group

	Y	Z	M	S	W	X
B51F	<i>D,B / + B53B</i> <i>+ M32B + DK</i> <i>+ M32D</i> <i>+ M34A</i> <i>+ B53B</i>	<i>/ / + B53B - M32S</i>				
B51K			<i>/ / + B53B</i>	<i>/ / + B53B</i>		
B51R					<i>N,B / + B53B</i> <i>+ M32B + DK</i> <i>+ M32D</i> <i>+ M34A</i>	<i>W / + B53B</i> <i>+ M32B + DK</i> <i>+ M32D</i> <i>+ M34A</i>
B51L	<i>D,B / + B53B</i> <i>+ M32B + DK</i> <i>+ M32D</i> <i>+ M34A</i>	<i>+ B53B + M32S</i> <i>+ B53B - M32S</i>			<i>W / / + B53B</i> <i>N / / + B53B</i>	

features of the Master (or pivot) feature group

The italics represent the restrictions from the combination of the Master "seat-face material" feature group (B51) with the "seat-heating" (B69) feature group

The underlines represent the restrictions from the combination of the Master "seat-face material" feature group (B51) with the "seats-reclining" (B44) feature group

Figure 60: The third step of the validation algorithm
(a two dimensional array with possibly multiple values as its elements)

group are combined with the equivalent objects of the first feature group in the sequence pattern of the planning function. The hierarchical structure of both feature groups eases the combinatorial search of the algorithm.

The steps of the combination procedure in the Rover R8's front

seats example are shown in figures 58, 59, 60.

Firstly, the B44 group is combined with the master group. Their result is sorted against the possible feature combinations created and each of them against its trim level availabilities. Similarly, the B69 group is combined with the master group.

The last two results, however, are combined in a different way: The research space is a two dimensional array that maps the master feature availabilities against all the legal common trim levels derived from both combinations of the feature groups with the master group. (figure 60)

The common options combined with the most specific restrictions represent the result of the three feature group specifications. The final result is then sorted in a similar way to that discussed before. If no additional feature group is participating in the specification of the part, the result is combined according to the two dimensional array philosophy with the result of the combination of the new feature group with the Master group. If no additional result is participating, the system graphically presents the result of the features combinations to the user.

At each step the algorithm matches only candidate features with common trim level availability(ies). The intersection process applies to three perspectives for every feature combination: Base features, territory and combination restrictions. One mismatch of either of them is enough to disqualify the validation of the combination of features (for more details see appendix 9).

Notice, it is possible in the example for a one or two feature combination availability not to match with any of the trim level options of the third feature group. For example it may appear that feature combinations such as "B51K" or "B51K + B44R" do not combine with any of the features of the "B69: seat heating" group. Such results are not discarded from the system but they are passed as **carry-over** usage statements. These statements are, as well, active and may combine with a fourth feature group, if it exists at a later stage.

Finally it should be noted that the algorithm could assign the "B69: seat heating" feature group to be the Master feature group and still reach the same results. The process of the combination, however, would be different and harder to code. This case is discussed in chapter 10.

8.2.1.2.2 implicit trim levels.

Trim levels represent the main substrate for the matching of the additional features (feature-chart-entries) of the vehicle. As mentioned earlier, feature-chart-entry objects represent one line of information in the AFC. An attribute of their frame structure is the trim level. It happens sometimes that the reference of the trim level of a feature is not explicitly specified in the AFC of the vehicle.

Figure 61 shows that no values have been compiled by specification services for the trim level of all four features

AUSTIN ROVER		PRODUCT ENGINEERING		ADDITIONAL FEATURE CHART		MODEL RANGE		AUS ROV RD UK		-D	
OFFICIAL FEATURE NO CODE (FEATURE WORD)		AVAIL - USAGE CONDITION		C09 - BRAKE SYSTEMS		DATE		04 DEC 89		DESIGN REL AUTHORITY EFFECT SIG REFERENCE	
		15 - MODEL C N - ENG. D 110		COMBINATIONS - RESTRICTIONS							
		10 F10G/EN L O F C P R RH		GROUPS							
		10 RH A D A E A I A		DE							
		10 RH S Y M P F V RH		P							
035	C09G	ANTI LOCK BRAKE SYSTEM (HDSKID)	10 XW	- - M - - -	1 C09B					XW0A5	P RD/1P/88
040			10 XW	- - K - - -	1 C09B					XW0C1	A RD/80/87
050			10 XW	- - E V - -						XW0A1	P RD/30/87
060			10 XW	- - D R - -						XW001	A RD/20/89

Figure 61: The trim levels of the "C09G": "Anti lock brake system" feature are defined implicitly in the AFC of the Rover R8.

REV NO	REV DATE	REVISION	DESCRIPTION	PAGE	GRP
6	11 SEP 88	REC NO 060	ADDED	1	C09
				END	

(C) MUST HAVE	(C) IF	(C) AND
(C) MUST NOT HAVE	(C) IF NOT	(C) AND NOT
(H) REJECTED TO	(C) OR	(C) OR
(P) PACKAGED WITH	(C) "COMMENTS"	
(I) INCLUDES		

that control the anti-lock brake system of the car. Consequently, no values will be assigned to the attributes of all the feature-chart-entry objects that represent those features in the system. Nevertheless, the trim level availabilities of each of these features are implicitly defined through the rest of the base feature restrictions. The trim levels expressed in such a way in the AFC document have been termed *implicit trim levels* in this research.

In order to decode the implicit trim levels the system consults the Base Features Chart data. It searches through the whole document and picks up all those trim levels that can be legally combined with the base feature specifications of the additional feature under consideration.

8.2.1.2.3 boolean algebra.

The validation algorithm uses the conventional data structures (Lisp lists) of the Territory Group Index Report, Base Group Index Report, etc. to translate the restrictions that appear in the AFC document. The boolean expression restrictions sometimes tend to be written by the specification people with no a specific discipline.

For example, a territory restriction such as "- M32S" and "- T51X - T53Z" would mean that the feature under consideration is not offered to the Spain Group (M32S) except the Canary Isles (T51X) and Greece (T53Z). This, in other words means that it is offered to Greece and the Canary Isles. The proper boolean restriction it

should be "+ T51X + T53Z" to avoid the double negative above. Fortunately, the validation algorithm can tackle odd situations like these and make sense of the contents.

8.2.1.2.4 the PLUS (+) and MINUS (-) signs of the usage statements.

The boolean qualifiers that appear in the final usage statements of a part (+ and/or -) are created by the validation algorithm. This process is governed by both design engineering and specification logic rules.

(i) Design engineering knowledge: feature groups which are offered with more than one option (feature) in the AFC of the vehicle cover - *or should cover* - all the current trim levels of the model for that point in time.

(ii) Specification Services: it is preferred by Specification Services that feature groups which cover all the trim levels of the model should be specified *positively* in the usage statement of the part. The usage condition qualifier of such a feature group is expressed in Boolean logic with the **plus (+)** sign. On the other hand, feature groups which are offered only with a single option (feature) in the AFC of the vehicle - eg. "B69: seat heating" - they *may not* cover all the trim levels of the model and should be specified **negatively (-)**.

In both cases the system works out all the trim levels of the feature group by referring to the Base Features of the model to compute implicit trim levels when necessary.

In the situation that only one feature is available from the feature group in question, three cases of its boolean qualifier may exist, plus (+), minus (-) and plus and minus (+/-).

In order for the system to work out the proper qualifier of such a feature group it consults both the trim levels of the vehicle and the availabilities of each feature in particular. The algorithmic expression of this case is shown below. (Remember the combination of the feature groups "B51: seat material" (master group) and "B69: seat heating").

IF the trim levels of the master feature group do not cover
all the trim levels of the feature in question

THEN + and -.

ELSE

IF the feature covers all the trim levels

THEN IF all the availabilities of the feature are
standard

THEN +

ELSE

IF (all the availabilities of the feature
are optional or base or legal etc.)

THERE IS A REFERENCE IN THE record OF ANSWERS?

assign it to the qualifier (RETURN)

ELSE query the engineer.

```
(if answer is no then +  
else + and -)  
update the record of answers.  
OTHERWISE -
```

The query to the engineer looks like: "If the model does not get this feature, does it get something else instead?"

There is a design engineering logic behind this situation in which the system interrogates the engineer. In the example the system has analysed all the trim levels of the "B69: seat-heating" feature group and concluded that it does not cover all the trim levels (ie. all the different top assemblies in manufacturing) of the Rover R8 model. In addition, not all the availabilities of the feature of the group are offered as standard fitments to the car. Instead, all of them are optional. Consequently, it suspects that something else may replace this fitment to the car. This can be illustrated with a real life example of a car.

The front seat of a car, for instance, is usually offered with a head restraint fitted on the top. If the case occurs that the design feature group that controls the head restraint condition of the seat shows the above characteristics:

- (i) it does not cover all the trim levels of the model and
 - (ii) all the available features are optionals (see availability in figures 31, 32, 33),
- then the system questions the engineer/auditor if there is

anything else that is fitted in the seat when the head restraint is not fitted.

If the answer is no, ie. no other part is fitted, the boolean qualifier is assigned to PLUS (+). On the other hand if the answer is yes ie. something else is fitted instead, then this means two seats different in appearance are added at the manufacturing stage: one with head restraint and another one with something else on the top. The boolean qualifiers in this case are + and -.

The first **IF-THEN** statement of the algorithmic expression is based on the logic that if the trim level availabilities of the feature under concern are more than those of the master feature, then the boolean qualifies must be assigned PLUS (+) and MINUS (-) in order to anticipate the missing combinations of the validation algorithm (the master feature is the reference point of the combination process).

Notice, the system distinguishes among feature groups which have only a single feature option; ie. these which have been actually compiled in such a way by Specification Services and those which resulted in a single feature option because some of their availabilities were excluded by the system with the application of the negative features discrimination.

Finally, notice that the system recursively creates all the different combinations of the PLUS and MINUS usage statements, for all feature groups which present similar behaviour. For example, some of the usage statements of the R8 front seat

(manual-passr) after the combination of the three feature groups are:

"B44R +B51L +B69D" , "B44R +B51L -B69D"
"B44R +B51F +B69D" , "B44R +B51F -B69D".

The involvement of the "B45: sport style seat" feature group in the above specification result which shows similar characteristics with the "B69: seat heating" group, would generate the following usage statements:

"B44R +B51L +B69D +B45", "B44R +B51L -B69D +B45"
"B44R +B51F +B69D +B45", "B44R +B51F -B69D +B45"
"B44R +B51L +B69D -B45", "B44R +B51L -B69D -B45"
"B44R +B51F +B69D -B45", "B44R +B51F -B69D -B45" .

8.2.1.2.5 Graphics Generator and Truth Maintenance Mechanism.

When the algorithm runs it creates all the possible combinations of the features of the feature groups specification under consideration. Thus, it usually generates a lot of information which makes it difficult for the engineers to absorb the whole bulk of data in textual forms (first prototype version of the system).

In the latest version of the system a piece of code has been written to present the result graphically. It concerns the possible usage statements of the part found by the algorithm

presented to tree-like legal combinatories that are restricted from their Base Features, Territory and Combination specifications.

A **truth maintenance** mechanism is supported at the user interface level, as well. This mechanism run invisibly to the user in the background of the validation algorithm's process. When it is invoked it graphically presents the route of legal combinations which the system followed to reach to the solution. The engineer and/or auditor can view internally any node of the **truth maintenance** tree and have fed back the reasons *why* a feature combination did not occur or *how* the restrictions in the top usage statement were generated. Additional menu choices enable the engineer and/or auditor to view the Base features of the vehicle or the internal structure of the Static data bases, as well (for more details see photos A10 to A14 in the appendix 10).

The whole function of graphics generator is driven by the *presentation type* facility built-in to Object Oriented Programming and FLAVORS, in particular. For more details see appendix 7.

8.2.2 **THE EXPERIENCE LINK**

(second development phase of the system)

The first phase of the system's development was based on the idea that the relation of the part descriptions to the feature groups that specify them is known. This is the design engineering knowledge and experience. In the first instance this knowledge was entered manually into the computer and only for a representative sample of the parts of the vehicle ie. exterior mirrors and front seats. The intention was that design engineers and specification people would provide all the knowledge required by the system, for all the parts in the company. However, when the second development phase of the system was launched - to cover fully all the part descriptions in the company - delays by Rover to feed the system with the information showed the problem faced: A team of people would be required to be constituted and work on a full time basis over a substantial period of time. There were two reasons for this:

Firstly quantity; the amount of information which had to be handled was extremely large. Almost 15,000 different part descriptions, each of them mapped against more than 300 feature groups currently exist in Rover. The definition of all *negative* features to each part description it would reach the range of $15,000 * 5,000$ manual checkings.

Secondly quality; the nature of this knowledge is very subtle. The *must* and *maybe* feature groups have to be distinguished unambiguously for every part description. However, only a few people within Rover really *understand* the Product Specification Concept. ie. experienced engineers, auditors or Specification Services. These people may disagree, as well, in the assignment of the *must* and *maybe* feature groups to the part

description. This is because everybody views the newly defined ~~model~~ and ~~maybe~~ concepts from his/her own profession perspective. It soon became apparent that a new solution to the problem would need to be found. Further investigation into the PIMS database brought to light the existence of two documents which are created through overnight batch processes: the VPG and EJA18V documents.

Both of these documents compile the final stage of a specific model. That is, all the parts that make up the vehicle (about 5,000), their usage statements, assembly links and quantity relationships. The difference between the two documents is that the VPG document pulls out from the PIMS data base all the parts of the vehicle which have ever been used all the years of the life of the model under consideration whereas the EJA18V states only the current parts. The first document costs Rover 10,000 pounds whereas the latter 5,000 pounds and as it will be shown later, such important results from their analysis could justify their cost rather than only been used as backup information. Both of the documents can be electronically fed to VAX tapes which makes their usage from a computerised system practically possible.

Figures 62, 63, 64 illustrate typical pages from the EJA18V document [15].

The first column in the left hand side (fig 62) records the part number of the part (HAH10022xxx), its fitment (0001) and the usage condition code (XW1134 99) which represents a usage statement for this part for this fitment. The third column describes both the part and its fitment. The fourth column keeps

Figure 62: A page from the EJA18V document.

Figure 63: A page from the EJA18V document.

the usage condition of the part. The fifth and sixth column is concerned with the effectivity of the part. ie. the time it was designed and used within the company and the time it replaced by another part. The last column indicates the other parts that this part replaces or the parts which is replaced from.

Information on the relationships between part descriptions and feature groups can be derived through the document. An expert system has been implemented which uses both the design format of the document and statistical analysis to generate information:

Firstly, the data are parsed by a Lisp program. Because of the huge amount of information that is carried in the EJA18V document, ISTEEL has split it to four separate files. The system links the files by means of logical pointers and constructs a new single data base of conventional type in the computer's virtual memory. This data base is constructed of nested lists that represent the format of the EJA18V document ie. physical part, fitments, usage conditions, date of appliance (figure 82). This parsing together with the grouping of the data to VPG areas takes an hour of processing time.

Secondly, every VPG group of the data base is individually processed and statistically analysed. The rules (information) generated for each specific VPG are stored into the equivalent area of the APN's catalogue VPG.

The statistical analysis of a VPG group translates the hierarchy of the conventional data to a hierarchy of equivalent objects and

makes use of its structure to derive facts for the parts. The automobile logic that the hierarchy represents is that a part description covers a wide area of actual physical parts (see section 4.1.1) each of them fitted to the car in different ways under different usage conditions, at one point in time.

Consequently, it is the need to acquire information from the layout format of the EJA18V document that forces the hierarchical grouping of the data even more. The data base of such FLAVORS physical-part objects is created *on the fly* and exists only for a short period of time. It disappears once the statistical analysis of the VPG group has been completed.

The same software engineering tool has been used for the structure of the hierarchy as that of the validation algorithm ie. nested hash tables. Notice, that the system only passes to the data base current parts (especially when the VPG data tape is used). These are parts that have not been replaced by other parts.

The next section explains the statistical analysis of the system in more detail.

8.2.2.1 statistical analysis

8.2.2.1.1 temporary data base of physical part objects

When a temporary database of FLAVORS *physical part* objects has been structured hierarchically representing a VPG group of data coming from the tape, the system applies a statistical procedure. At the beginning it assigns feature groups to every object which is implicitly expressed through the usage statement of the object. Furthermore, it corrects any inconsistencies which appear in the EJA18V data tape before it applies evaluation to the data.

For example, in the VPG document or even in the EJA18V document of the current model, there are referencies for feature groups which are no longer in existence such as "B40: seat face material". The system assigns the feature groups in respect to the current feature group availability in Rover. In sequence, it evaluates the result in respect to a level higher in the data base hierarchy, that of **part description** instead of **physical part**.

The statistical analysis is based on a combination of data *percentages* and *patterns* that appear in the EJA18V document. The reason for this combination is that the EJA18V document which represents the search environment of the statistical algorithm is not standardised eg. there may exist many referencies for a part description or none. The patterns of usage statements may vary from one to many, as well. The term *pattern* used in these contexts means a combination of similar feature groups. For example, using as basis the "B51, B44, B69" the "+B51K -B44R +B69D" matches the pattern but "B51K +B44R" does not.

The following pseudo code represents the statistical algorithm applied on the data.

IF the feature group appears in the usage condition of all the physical parts with the same part description, with 100% rate, then this feature group is a must feature group for that part description

ELSE

IF the feature group appears at least 70% and it is involved to 4 different feature combinations patterns of physical parts with identical part description O R

IF the feature group appears at least 60% and it appears to 3 different patterns

O R

IF the feature group appears at least 50% and it appears to 2 different patterns

O R

IF the feature group appears at least 80% and it appears to 5 or more different patterns

T H E N

the feature group it may be a must and it should be checked from the validation algorithm

OTHERWISE

the feature group is a maybe (or optional) for the part description concerned.

8.2.2.1.2 Need for further investigation

The statistics algorithm is a *prototype* procedure. In every statistical marker appearing in the IF statement of the procedure, the test area has been increased by at least a 5%. For example, though practising with the EJA18V document has shown that a 75% is enough to conclude correct results in the first condition of the algorithm, the author has reduced the conditional marker to 70%. This gives an additional 5% possibility of the data satisfying the condition. The 5% expansion of the IF statements, however, gives more features to the validation algorithm for auditing. The requirement for additional processing does not impact significantly on real time terms, as the computer performs all the calculations, but increases the credibility of the results of the algorithm drastically.

Initially, the statistical algorithm was designed with the simple idea that if feature groups appear all the time in the usage conditions of different *physical parts* with the same part description, then these feature groups would seem to control the fitment of this part description. In the terminology which the author has previously defined is a *must* feature group for the part description. Similarly, feature groups that do not appear always are *maybe* (or *optional*) feature groups for the part description.

Experience with the VPG and EJA18V documents has shown, however, that though such assumptions are generally the case, they may be proved invalid in some situations. For instance, the usage condition of the third physical part of figure 64 and the fourth

usage condition in the same figure are:

" +B40C +B44B " and " +B40B +B44R -B69D" ,
respectively.

The "B69: SEAT HEATING" feature group does not appear in all cases where physical parts of the "SEAT-FRONT COMPLETE MANUAL" part description are referenced. This, does not necessarily mean that the feature group "B69: SEAT HEATING" is a maybe feature group for the part description. It is possible that the engineer wanted to specify the part he designed with the "B69: seat heating" feature group but the combination of all three features was not valid. Apparently, this is the case as the AFC of the Rover R8 model reveals.

Consider that the old B40C feature represents the current B51K (seat face material zenith). Then the combination "+B40C +B44B " and "+B51K +B44B " are identical. The combination "+B51K +B44B " is offered in the trim levels M and S ie. trim levels 3 and 4 (see figure). Similarly the "B69: heat-seating" feature group is offered in the trim levels X and Y ie. 5 and 6. It is apparent that the "+B51K +B44B " combination does not intersect with the "B69: SEAT HEATING" feature group.

In this perspective, it can be assumed by the system that the "B69: SEAT HEATING" feature group may be a must feature group, until it proved otherwise because it was found by the statistical analysis to be on the limit of the must feature group confidence.

The question which arises now is how can such a situation be checked electronically?

8.2.2.1.3 The need for statistical markers

The general idea is that as all the data (BAFC, Territory code Index report etc) needed for the specification of the parts of a vehicle are already compiled in the memory of the computer, the statistical procedure could consult the validation algorithm to check upon feature combinations. For instance, the validity of the example mentioned earlier:

" +B40C (or B51K) +B44B against the feature group "B69D".

The system could investigate all the possible combinations of features which may infer an optional feature group for all referencies of a part description but this would require a large amount of processing time. For this reason the system needs to discriminate among all the feature groups of each part description. This now makes it clearer the reason why the statistical markers were invented: to let pass only the feature groups whose need in the specification of the part is *highly* doubtful. The word *highly* expresses the statistical conditions under which a feature group must be examined from the validation algorithm. Notice, the statistical percentages have been assigned by the author after studying and practising with both EJA18V and VPG documents and may be subjective.

8.2.2.1.4 The need for feature combinations patterns

It was mentioned earlier that the data of the EJA18V document do not follow any rules. In particular, a quantitative approach which is so important for statistical analysis discipline does not seem to exist. References to different physical parts (hence quantitative figures) can vary from zero to many. For example there may exist 17 references for the part description "seat-front complete electric-driver" or 3. Similarly reference for the the *squab frame* of a seat may exist only once.

The problem arises because of such high differences in the distribution of the data for different part descriptions. It seems almost impossible to implement a general purpose procedure which would infer statistically new rules only from the number of part description entries, with the same credibility for every part description.

The following example is chosen to illustrate the logic in the design of the statistical rules. Assuming we have to analyse statistically the following data for the "seat front complete electric-driver" part description:

"seat-front-complete electric-driver"

Physical-part1 : + B51N + B44R + B69D

Physical-part2 : + B51L + B44R

It can be seen that the feature group "B69: seat heating" is used in the 50% of the references, which shows high indication that this part is *maybe (or optional)*. It has not be used for a very high percentage (50%). However, this does not reflect the *reality* as it can be seen that it is only one case in which the "B69:

seat heating" group has not used. It is quite possible that it was only in this one combination that it was not valid.

Now, given a different distribution of data:

"seat front complete electric-driver"

Physical-part1 : + B51N + B44R + B69

Physical-part2 : + B51L + B44R

Physical-part3 : + B51N + B44R + B53F

Physical-part4 : + B51K - B69K

In this case, as well, the feature group B69 appears in 50% of the references to the part description but it is more possible now for the feature group "B69: seat heating" to be optional to the part description. That is because it is combined with many more different combinations which decreases the possibility of all of them being invalid. It is because of that the author has introduced the concept of *feature patterns*. To give significance into the actual structure of the data themselves within the EJA18V document in the decision of the statistical algorithm.

The rest of the section deals with the interface of the statistical procedure with the validation algorithm and inferences which can be generated.

8.2.2.1.5 Inference from the history of the EJA18V data

It will be shown shortly that some information can be derived

from the past history by studying the format of the EJA18V and/or VPG document provided the data are structured and interpreted accordingly. This was one of the reasons that the data was structured hierarchically in the first phase: to reveal information of the domain. For example, it can be seen from figure 63 that the feature group "B69: SEAT HEATING" is a ~~maybe~~ (or optional) feature group:

The usage statements of the last two physical parts do match except in the last feature; the "B69D". This means that it is valid to combine the B69D feature (driver's & pass seats heated) with the rest of the features combination as there is at least one reference for this (the second combination). However, in the first feature combination, the "B69D" feature has not been mentioned in the past by the engineers. This can be interpreted that the feature group "B69D: Driver's & pass seats heated" is not always needed to specify the design of the seat, consequently it is optional for the part description "seat-front complete manual-driver".

The system, searches each path in the temporary hierarchical data base (VPG group, family, primary description, part-description) down to the **part description** level and combines all the referencies for the part description as described above. In this way it tries to acquire most of the knowledge for *optional* feature groups from the history of the EJA18V data.

The need for individual routes is important as the secondary descriptions of parts from different primary-description parent

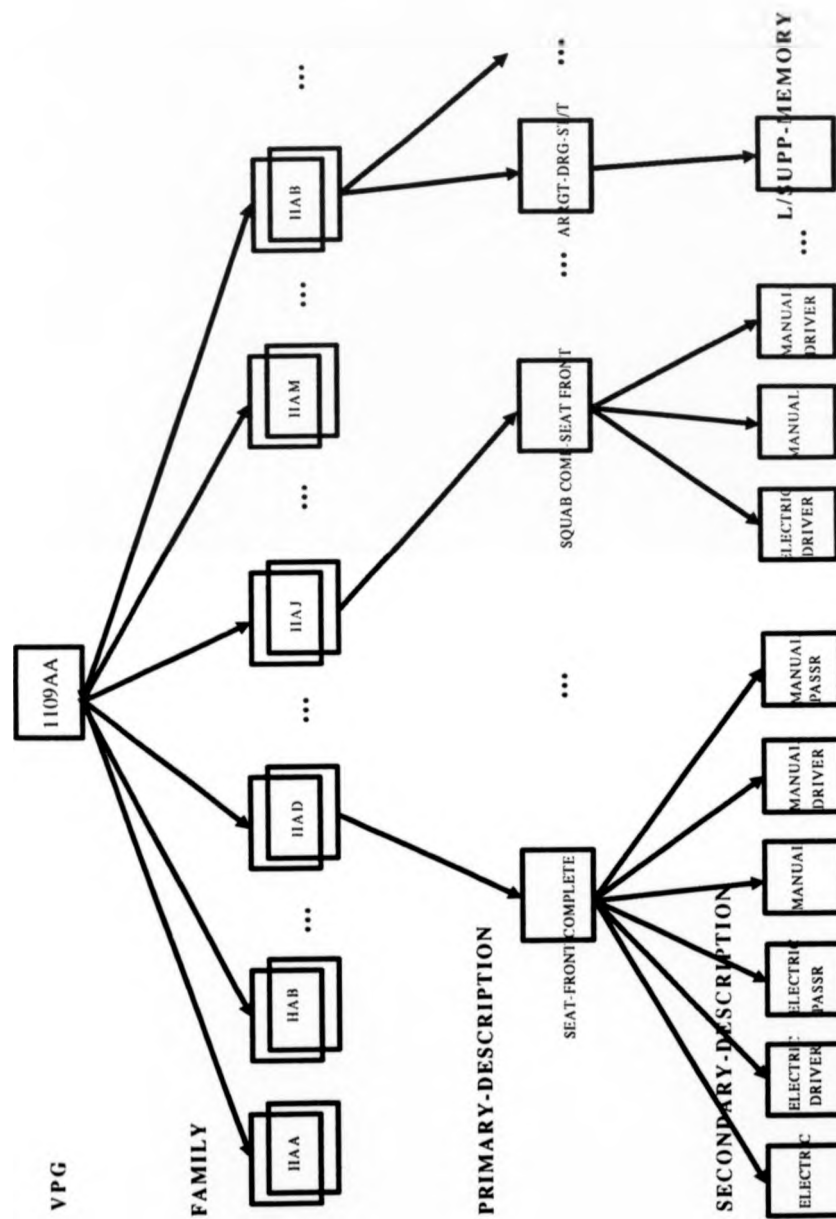


Figure 65: The 1109AA VPG : SEAT-COMplete (FRONT)
as constructed temporarily in a hierarchical OODB.

may match and the results be mixed up. ie. "seat-front-complete manual-driver" and "cushion-comp-seat front manual-driver" (figure 65, 82).

The system cannot derive all the information for optional features from the history of the EJA18V tape because not all the relevant information has been entered in the past. Although the validation algorithm has been implemented on the principle of generating all the possible combinations of the feature groups at specification, the existing manual specification system has operated for years to input only some of them at a point in time.

Consequently, a large amount of past information is missing. The only solution in this case is the use of the validation algorithm itself to check for maybe (or optional) feature groups.

8.2.2.1.6 Inferring from the validation algorithm

The following steps underline the major phases through which the feature groups which have been found to need further investigation pass until finally audited by the validation algorithm.

Firstly, each feature group for re-evaluation is related to referencies of feature combinations existing in the same part description ie. the specific path in the hierarchy under consideration. These are feature combinations that indicate that

a decision can be derived for the feature group under investigation. For example, the feature combination "+B40C +B44B" (or "+B51K +B44B" figure 64), mentioned earlier, is related to the feature group "B69: seat heating" as a possible indicator to reveal the nature of its application in the usage statements of the part (optional or standard). This is because another feature combination for the same part description reference has been found in the data base that refers to the same feature groups as the original feature combination and the feature group in question. This is "+B40C +B44R -B69D" (figure 64). It is shown diagrammatically below that the "+B40C +B44B" feature combination is going to be audited against the "B69: seat heating" feature group from the validation algorithm.

" +B40C + B44R" ----> "B69: seat heating"

The system may need to calculate more than one features combination references for the same feature group and more than one feature group for the same part description.

The AFC data referring to the feature groups to be examined are accessed from the data base. The validation algorithm checks if the feature group under examination can be specified somehow with any of its feature combination references. If the answer to the validation algorithm is positive at some stage, the process terminates and passes to the next feature group in the queue to be examined by the validation algorithm. A positive result from the validation algorithm would mean that the feature group is maybe (or optional) whereas a negative one after the validation of all referencies of the feature group it would mean that the

feature group may be a must. If there is a combination of features which are all negative to the part description under consideration (for the specific path of the hierarchy), then the outcome if it is negative, it is ignored.

The following example considers that the feature group "B69: seat heating" is going to be validated against its two features combinations referencies: "+ B51L + B44R" and "+ B51N + B44R + B53F". The validation process is shown diagrammatically in figure 66.

The *global feature group* represents a pseudo feature group which is created dynamically by the system and offered to all trim levels of the model. The pseudo feature group afterwards is combined with the feature group in question. The need for the dynamic creation of a pseudo feature group is to enable the statistical procedure to use the routines of the validation algorithm and therefore must maintain the original format of the arguments (feature combinations) passed for evaluation.

8.2.2.1.7 the history of the EJA18V document helps the validation algorithm

All the way through the validation algorithm's process of combining the features of the feature groups under investigation, the system increases efficiency by recovering information -if it exists - from the data base concerning the validation of specific

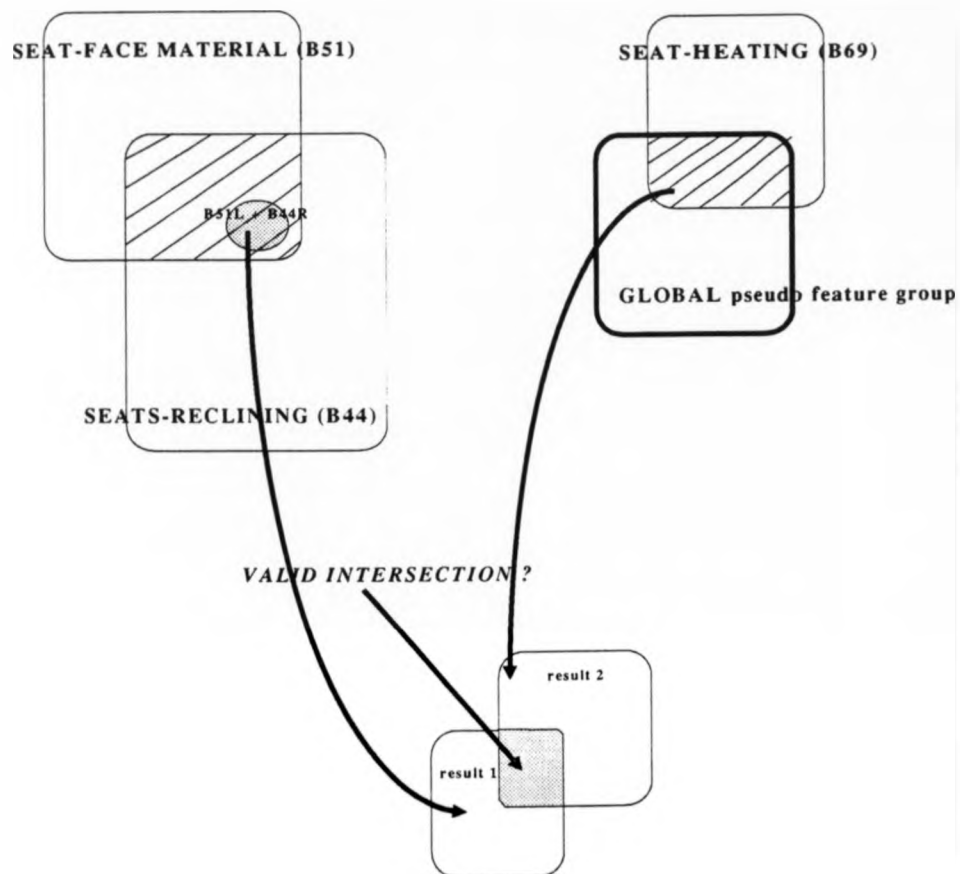


Figure 66: How the validation algorithm helps in the statistical analysis of past data.

combinations of features, instead of always running the algorithm. The algorithm runs when no such information can be recovered from the data base. The following example explains the situation.

Consider the statistical procedure has determined that the part

description "cushion complete seat front" and the " +B25G + B51L" feature combination must be validated against the feature group "B53: seat side rolls". It can be seen from the data base, from another part description's referencies (ie. a different node in the VPG hierarchy) that such a combination it is not valid.

In figure 62 the part description "headrest-compact-front bolster-nodding" is conditioned with two usage statements (the last ones) such as

" + B25G + B51L" and
" + B25G + B51R + B53B".

These two usage statements represent the valid combinations of the feature groups of a single physical part that has been specified with (B25, B51, B53). This is because the feature combinations entered into the system at the same time as shown by

XW9H1

VAAO2 /511.

which represents a *manufacturing* event in Rover. This could be for instance the Methods build phase of the Rover R17 coupe model, which eventually can be translated to a real date. , 1)

Speculating with the above example, it can be thought that as the engineer entered usage statements of the same physical part at the same time, he would obviously refer to the same design intentions of the part (feature groups). In that respect, the absence of the "B53: seat side rolls" feature group in one of the usage conditions of the part must mean that such a combination was not valid. Hence, the feature combination " +B25G + B51L"

cannot be valid with any feature combination of the "B53: seat side rolls" feature group and consequently, the validation algorithm does not need to be consulted.

Finally, it must be clear that though the system uses the validation algorithm when it needs it, it also updates it for its benefit. That is, it does not exclude feature groups which no longer exist from the AFC data base when it accesses the scope of its search, for instance the "B40: seat face material" feature group. Instead, it treats them as being currently available in order to infer more information from the past history of the EJA18V document for *must* and *maybe* feature groups, as such references still exist in the document. (see section inferring from the validation algorithm).

The above process is repeated for all part descriptions in the VPG group. When all information for *must* and *maybe* feature groups is generated for all the parts of the VPG group it is fed temporarily to the computer's virtual memory, in the form of the *experience link* part of the system and afterwards is stored permanently in the knowledge base of the APN catalogue. The same process is repeated for the next collection of the data coming from the EJA18V tape representing another VPG group. A schematic of the system at this point in its development is shown in figure 67.

8.2.3 THE APN CATALOGUE AND THE ADDITIONAL FEATURE CHARTS (third development phase of the system)

The EJA18V data represent a finished current product (model) within Rover. Any information generated by the statistical analysis of the EJA18V data is then important for the company and it consequently should be stored permanently for use of any department that may needs it.

Up until this stage only a representative of the part descriptions of the vehicles has been implemented manually in the form of FLAVORS *part-description* objects within the Lisp's virtual memory ie. exterior mirrors and front seats (FLAVORS can only store data in the virtual memory).

The Additional Feature Charts although they are created automatically as FLAVORS *feature-chart-entry* objects from conventional data during the function of the validation algorithm, are also stored in the computer's virtual memory. The problem is that whenever the computer is switched off for re-booting or from power failure, all those objects resident in its virtual memory are lost. That means that all the information created from the statistical analysis and fed to the *part-description* objects is lost. Additionally, the system has to start the installation procedure of the AFC and structure the knowledge base of the *feature-chart-entry* objects from the beginning each time. The requirement for a permanent storage of the information becomes crucial. It is achieved in the third phase of the system's development which translates it from a

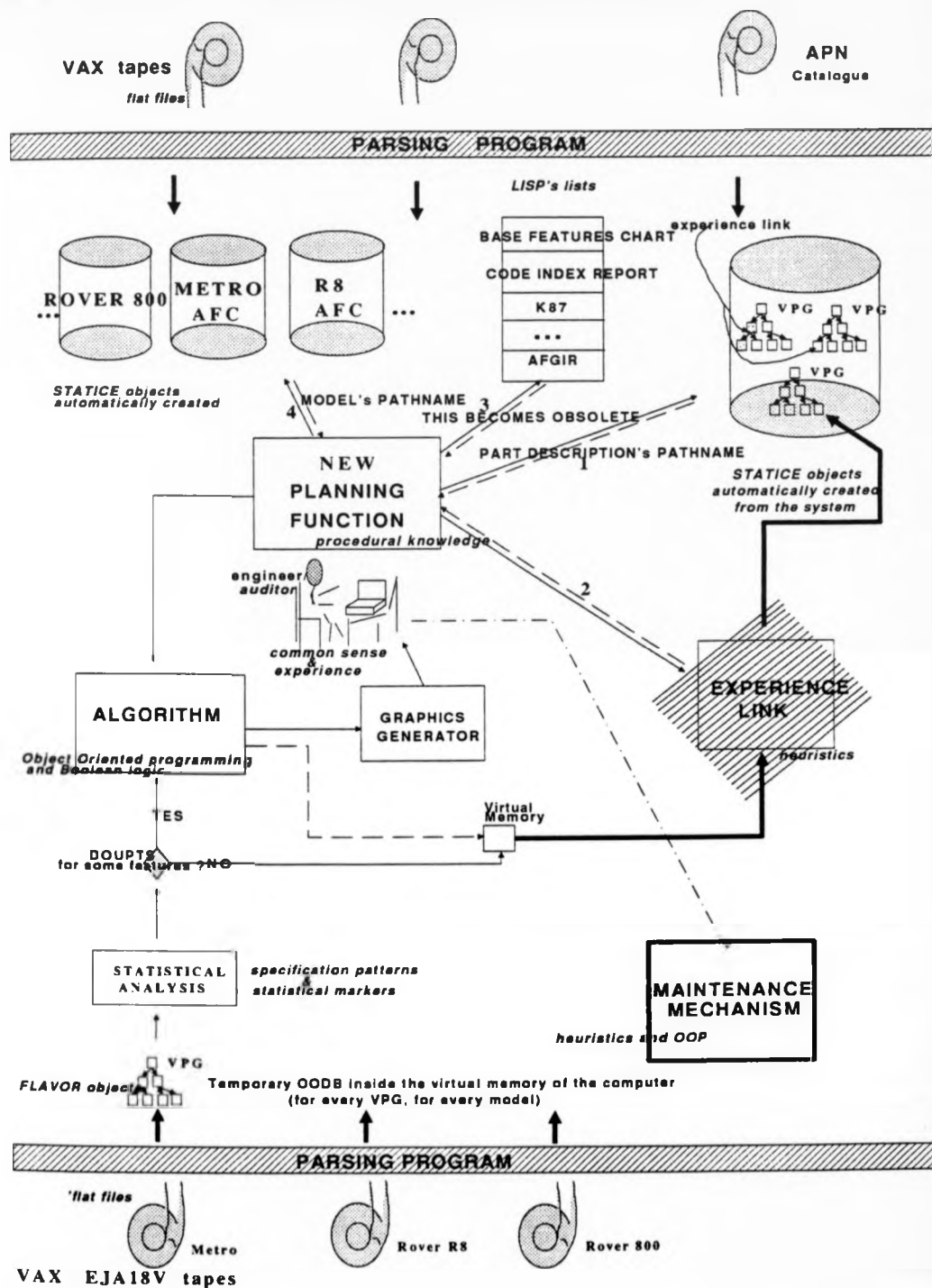


Figure 69: The third development phase of ROOVESP

prototype to a real life system.

A real Object Oriented Data Base, Statice, has been used to store the information of the system that extends the power of objects beyond the bounds of the Lisp's virtual memory, out onto the disk and over the network.

The design logic and functions of the system remain unchanged. The system still thinks and operates in terms of FLAVOR objects and instances of them. In reality, however, these objects are instances of Statice entities that keep all the characteristics of the information belonging to a real data base: retrieval under certain password, sharing among users, update and deletion with concurrent transactions and, most importantly permanent storage in the hard disk of the computer. Only the planning function of the system it has been changed slightly.

Figure 68 shows how both the *feature-chart-entry* and *part-description* objects were stored in the memory of the computer in the first and second development phases of the system. The *part-description* objects were entered into the system manually by the user.

Through the *experience-link* of the system, the initial planning function translated to FLAVOR *feature-chart-entry* objects the parsed data belonging to the feature groups which would specify the usage condition, from the Additional Features Chart tape. These *feature-chart-entry* objects defined the search space of the validation algorithm.

In the third phase of the system's development, all the AFC data

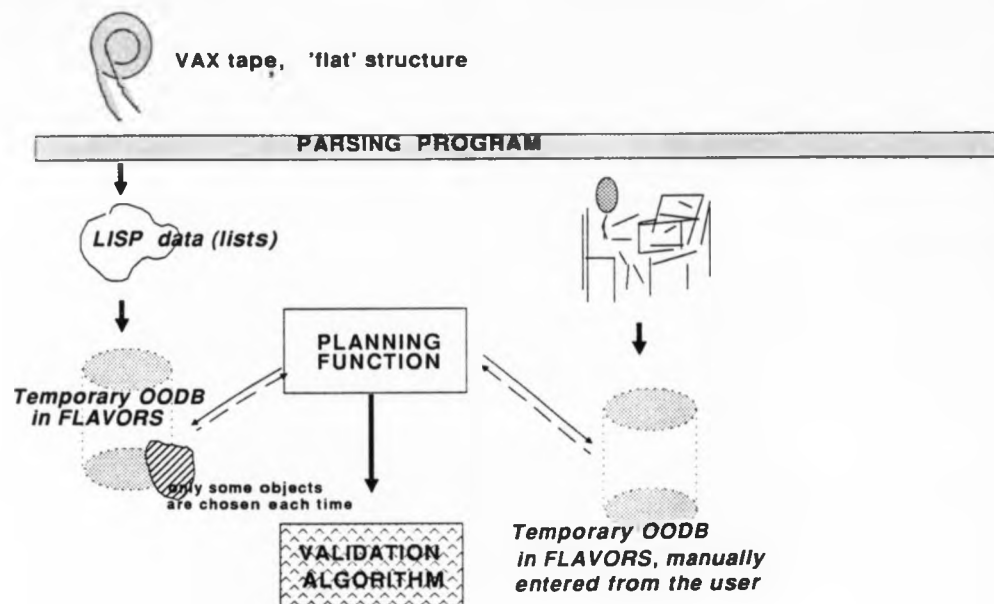


Figure 68a: Input of data in ROOVESP during the first and second development phases.

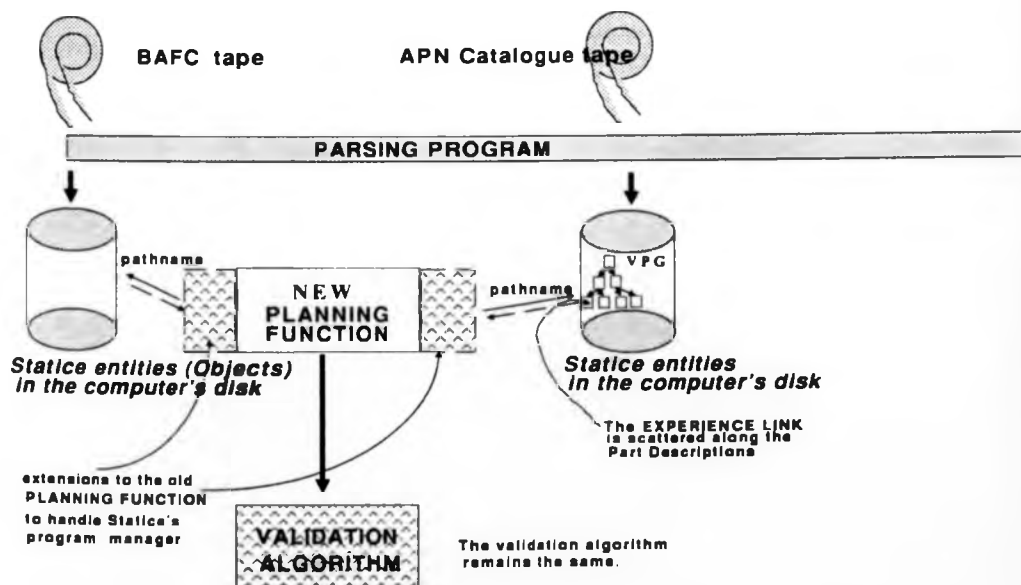


Figure 68b: Input of data in ROOVESP during the third development phase

are translated to objects (instances of Statice entities) well before the actual application of the program and stored in the Statice Object Oriented Data Base. The APN catalogue is created automatically, as well, directly from the VAX tape to a new Statice Object Oriented Database (new pathname). Figure 69 illustrates the new version of the system with real data bases of information.

It must be mentioned, that although the transmission of all the AFC of all the Rover models (12) into Statice takes less than half an hour of processing time, the creation of the APN catalogue has been estimated to require about 4 or 5 days of continuous running of the system. This is not a major problem as it only needs to be done once.

The reason for the long processing time of the creation of the APN catalogue Object Oriented Data Base is that there is firstly a large number of part descriptions existing in Rover (almost 15,000) and the system has to **hard-wire** all the relationships of these objects with **bidirectional** links for all VPG hierarchies (figure 65, 72). Below is detailed the process of the creation of the APN Statice OODB.

8.2.3.1 Storage of the APN catalogue to the Statice OODB

The system stores the APN catalogue data in Statice by VPG group,

in three steps:

Firstly it designs the appropriate Statice *schema*. This is, the definition of all the Statice *entities* required to represent the data hierarchy in APN catalogue ie. VPG group, family, primary description, secondary description.

Secondly, from the structure of the data coming from the tape it assigns the equivalent Statice entity in consideration and automatically creates new instances of the entity. The creation of new instances (objects) involves the initialisation of some of the attributes's values of the entity coming from the parsing of the data. (VPG code, design-responsibility, part-description etc).

Finally, the system **hard-wires** the relations of all created instances of all different entities inside the VPG. Hard-wiring means the linkage of the instances (objects) with the internal memory references of the computer for the objects. In simple terms, the objects themselves. The hard-wiring of relationships of the objects is implemented in both directions in the VPG hierarchy.

For example, consider the "seat-front complete" instance (object) of the Statice entity "**primary-description-id**". It has two additional attributes in its frame design as well as those that represent the parsed data from the tape:

has-secondary-descriptions and belongs-to-family.

Both attributes are initialised with actual objects. The first attribute associates the instance with the rest of the objects downwards to the hierarchy whereas the latter makes the association upwards.

8.2.3.2 Update of the planning function of the validation algorithm

In the third version of the design of the system the planning function has been updated to meet the needs of *interfacing* the data retrieval and update of the Statice OODBs objects with the already existing parts of the system eg. validation algorithm, statistical procedure etc.

The planning function in the higher level of the application of the validation algorithm isolates it from the Statice data base and treats the data as Statice entity instances rather than FLAVOR objects. That is, it is concerned with the new software protocols of the new environment ie. data security for access and retrieval and data transactions that guarantee the consistency of the data base at any stage of the process even after a cold-boot of the machine or a system crash. That makes the Statice data manager program interface without complaints with the rest of the program.

The new domain of the planning function is shown in detail in figure 68b. It accesses both the part description of the APN catalogue and the AFC data of the model through specific pathnames in the Statice OODB. In sequence, it accesses the values of the attributes of the *part description* object that keep the *experience link* as it has been generated from the application

of the statistical procedure (see photos A16 and A17 in Appendix 10). These values in combination with the value of the attribute in the part description that keeps the *negative* features specify the search space of the validation algorithm. These are instances of *Statice* entities retrieved from the AFC OODB and passed as arguments to the validation algorithm.

The close relation between the *Statice* entities with the *FLAVOR* objects that represented the data of the system in its first two development phases, means that as far as the algorithm is concerned there are no changes in the type of the arguments it applies.

As mentioned earlier, the *experience link* is retrieved from some of the attributes of the part description object. This is the case with the last version of the system where the *experience-link* becomes obsolete and no longer exists as a separate entity within the system but in the form of *scattered* attribute values within the frame design of each of the part description object. In this case the *experience link* would be deleted from diagram 69.

8.2.4 THE *STATICE* OODBs OF ROOVESP

8.2.4.1 The Additional Features Chart OODB

The Additional Features Chart schema is implemented in *Statice* with only one entity type: the **feature-chart-entry** entity (figure

70). For more details see appendix 5.

This entity represents one line of information from the AFC document after it has passed through the parsing program of the system. The need of the AFC database to fit with the rest of the program created two new requirements in its design.

Firstly, the **feat-group** (feature group) attribute of the entity has an inverse reader function, the **entity-of-the-feature-group**. This function represents a *one-to-many* relationship. Originally, the planning function of the validation algorithm accessed the data that corresponded to each feature group that would specify the part by *matching* the feature group's code. Afterwards, it would translate them into objects of type **feature-chart-entry**. This represents a *one-to-many* relationship. The definition of the inverse reader function **entity-of-the-feature-group** implements that relationship automatically. An inverse index has also been defined to speed up the performance of the relationship.

Secondly, the **:cached** tag appears in all attributes of the **feature-chart-entry** entity. This results from the need of the validation algorithm to use data base information (instances of entities) in the same way as was implemented originally to combine flavor instances.

The objects for combination are now entities of a real data base, that everybody can use instead objects in the computer's virtual memory. Their access is governed by data base management

```

(define-schema the-data-base-of-the-AFC-of-R8
  (feature-chart-entry))

(define-entity-type Feature-Chart-Entry () ;;; no inheritance
  ((feature description string :cached t :read-only t)
   (all-the-combination-restrictions list :cached t :read-only t)
   (all-the-territory-restrictions list :cached t :read-only t)
   ***
   (class string :cached t :inverse-index t :read-only t)
   (body string :cached t :read-only t)
   ***
   (feat-group string :cached t :inverse entities-of-the-feature-group
:inverse-index t)
  .....))

```

Figure 70: the Feature-Chart-Entry entity type of the AFC databases in Statice.

principles, such as definition of the data base's pathname and most importantly, consistency of the data during all the transactions by the users. This, in software terms means that all functions of the original code would need to be interfaced with the Statice's primitives that control such behaviour. The modification of the original algorithm in order to incorporate those primitives is a lengthy task which may affect the internal structure of the program and destroy the system integrity. Fortunately, the Statice facility of *snapshots* can be used to

overcome the problem without touching the original code.

8.2.4.1.1 snapshots

Attributes of Statice entities can keep a copy of their original values with the **:cached** option tagged in their definition. The cached values reflect a *snapshot* of the database state - a frozen copy of what was in the database at a certain time in the past. This is because as transactions access and change the values of the attributes in the database, the cached values never change. The good thing however, with cached values is that they can be accessed outside a transaction without causing a problem with the Statice database manager program.

Summarising, within a transaction, a reader function gets the value of an attribute in the database. If **:cached t** is specified, the reader can then work outside a transaction, skip the formalities of the database principles and access the cache slot for the database value. This characteristic of the Statice database has been used to smoothly interface the original algorithm with the newly created AFC Statice data base. The arguments passed to the algorithm are Statice entities. The virtually identical functional similarity between them and the FLAVOR objects and the access of only their cached values, makes the algorithm 'think' that they are FLAVOR objects upon which it carries out its process without complaint.

In addition, no data inconsistency is created as the validation algorithm only uses the values for combination and does not change them. Besides for double security the **:read-only** option has been tagged to all attributes of the **feature-chart-entry** entity. All the AFC of all different Rover models are compiled and loaded to Statice with the same *schema* definition under different pathnames.

8.2.4.2 the APN catalogue OODB

8.2.4.2.1 The APN catalogue entities

The schema definition of the APN catalogue database, as mentioned earlier, is comprised of the entity types : Sub-Sub-Sub-VPG (VPG-code), family-within-SSS-VPG, primary-description-ID, secondary-description-ID and a logical entity type called **reference-to-the-Rover-cars---actually-Austin-Rover** (see figure 71). For more details see appendix 5.

The latter entity type is designed because every **secondary-part-description-ID** should include such information structure. For economy of development, therefore, it is preferable to define such a structure rather than input the same kind of data to all entities individually.

```

(define-schema ROVERS-APN-CATALOGUE (SSSVPG, Family, Primary-Description-
ID, Secondary-Description-ID, CARS-reference)

(define-entity-type SSSVPG ()
  (VPG-code string :unique t)
  (Has-families (SET-OF family) :inverse index t)
  ...))

(define-entity-type FAMILY ()
  (Family-code string :inverse the-actual-entity-of-the-family)
  (belongs-to-SSSVPG SSSVPG)
  (Has-primary-descriptions (SET-OF primary-description-ID) :inverse-index
t) ....))

(define-entity-type PRIMARY-DESCRIPTION-ID
  ());; no inheritance
  (primary-description string :inverse ... :cluster t) ...
  (has-secondary-description (SET-OF secondary-description-ID) ...))

(define-entity-type SECONDARY-DESCRIPTION-ID ()
  (part-description string :unique t :no-null t :inverse ... :cluster t)
  (belongs-to-primary PRIMARY-DESCRIPTION-ID) ... (Has-Must-feature-groups
CARS-REFERENCE) (Has-Maybe-feature-groups CARS-REFERENCE) ...))

(define-entity-type CARS-REFERENCE
  (Mini-musts list) (Mini-maybes list) ... (Rover-R8-musts list) (Rover-
R8-maybes list)
  ...))

```

Figure 71: the Feature-Chart-Entry entity type of the AFC databases in Statice.

The original design intention of this logical entity type was to represent a dynamically adjustable array. However, the inability of Statice to support arrays as attribute types forced the implementation of static vectors. The Lisp environment supports good documentation for arrays and adjustable ones, as well. The solution to this drawback in the future, consequently, would be the implementation of a piece of code which would define a *physical* type of adjustable array on the top of the already built-in attribute types in Statice.

8.2.4.2.2 Uniqueness of the instances of the entities

It was mentioned in the discussion of the AFC data base that the reader function of the attributes express relationships between the objects in the database. Therefore, when a knowledge engineer designs a schema in an OODB such as Statice he must be careful to think about each attribute, and decide which kind of reader function he wants it to have. In this way the database design becomes *object oriented* by the means that the knowledge engineer thinks in terms of the attributes (properties) of the object which in sequence naturally imply the relationships in the database, instead of thinking of the relationships first as happens with conventional databases.

Several important choices for the APN catalogue schema have been made by the designer. Take, for example, the implementation of the entities in the APN catalogue schema that represent the

hierarchical structure of the APN tape: VPG code, family, primary description, secondary description. The function **secondary-description-ID-part-description** is *one-to-one*. This is convenient, because one would think that a part description should be unique. The same for the VPG code, the family code and the primary description. In reality, however, the secondary description of parts of different primary description is not unique. The secondary description of the "seat-front complete manual-driver" is "manual driver". The same with the secondary description of "squab-comp-seat front manual-driver" (see fig 28). Consequently, there is an *one-to-many* relationship of *secondary-description* names to Statice entities. For this reason in the definition of the **secondary-description-ID** entity the *secondary-description* of the parts is not used as the name but rather the whole part description (*primary* and *secondary*) which is unique.

8.2.4.2.3 Performance issues

The main problem with the APN catalogue OODB is that it takes a considerable amount of time to be constituted (4 or 5 days of continuous running). Therefore, the knowledge engineer must decide on the implementation of indexes which may be inverse or direct dependent on the relationships created by the reader functions of the entities's attributes.

Although the difference in data quantity between the AFCs of all models and the APN part-descriptions is not large, during the trials of the creation of the APN catalogue OODB the time needed was proportionally longer. (It has been estimated that there exists about 15,000 part descriptions whereas an average number of feature-chart-entry entities for each model is about 500, hence $500 * 10$ (Rover models) = 5,000 feature-chart-entry entities in whole).

The difference is that the feature-chart-entries for each model are not related to each other and the structure of each AFC data base is quite simple (only one entity). This is not the case with the VPG catalogue, however. There are five different entity types which are linked together bidirectionally in hierarchical structures. Consequently, the system needs time to work out such relationships and as the database grows the process slows. For this reason **inverse-indexes** have been defined for all the inverse reader functions of all entity types in order to speed up access time for the Statice entities. These functions are constantly called by the program in order to create the links in the hierarchy. In addition, direct indexes are defined for **set-valued** entity attributes for even greater performance. This subject is discussed in more detail in appendix 2.

In a further step the author defines **clustering** techniques in order to achieve greater performance in the internal structure of the way Statice itself processes. Statice is a sequence of *pages*. Everything in the database resides on some page (records, indexes) etc. When Statice accesses information in the database

it creates a buffer in the virtual memory of the computer. The buffer contains a page worth of data. The speed of accessing any data on that page is greater than accessing a different page which must be fetched from the database. Reference in this subject can be found in the Statice's manual [3]. Increase in performance can occur if the knowledge engineer can predict group of entities that would need to be accessed together during the running of the program. Entity types as they are created automatically by the program could predict where they would reside because of the hierarchical connections (see photo A15 in appendix 10) at the VPG-group level and be manipulated accordingly in such a way that relevant entity types would reside in the same group of pages. Indexes and clustering techniques have reduced the creation time of the APN catalogue from almost 4 to five days to 2 days of processing.

8.2.4.2.4 images

Graphical data have been stored in Statice in the form of arrays (two dimensional bit arrays) in order to meet the more general objectives of the CIE project mentioned in section 7.5. The images have been scanned by both Apple Macintosh and Symbolics platforms and are represented as **real** values in the image attributes of the Statice's entities (arrays). Consequently, these images can be scaled accordingly and because of the hierarchical links which exist between the Statice entities in which they belong, and the facility of *presentation type* that

the Statice entities possess, graphical representations of all the different vehicle's assembly structures can be derived. In this way Statice could support graphically the manufacturing orientation of the Rover's usage condition data base (see photos A8 and A9 in Appendix 10).

8.3 FURTHER DESCRIPTION OF ROOVESP

8.3.1 *The statistics procedure*

In the previous two section of the chapter the way in which the system statistically analysed the data of the EJA18V tape was discussed and how it electronically fed them to the equivalent VPG groups of the APN catalogue in Statice. During that process FLAVOR objects were created temporarily in the virtual memory of the computer, taxonomised and used by both the statistical procedure and the validation algorithm. The overall process is reasonably fast. It has been estimated that up to two or three hours of processing time is required for the system to infer fully all the rules of the data of a single model. Most of this time is consumed in accessing the APN database in Statice rather than in the actual statistics and combinations of the features.

Figure 72 illustrates the "experience link" of the system as it now exists scattered in the form of attribute values of instances of all part description objects (actually *secondary-description-*



256

ID entities) in Statice. The attributes that keep the information of the must and maybe (or optional) feature groups in the part description object are the has-must-features and has-maybe-features. These attributes are by themselves instances of the reference-to-the-Rover-cars---actually-Austin-Rover Statice entity (symbolised, for simplicity, "cars's reference" in figure 72. See, also, appendix 6).

The reference-to-the-Rover-cars---actually-Austin-Rover entity (in figure 71 it is represented as the CARS-REFERENCE entity) represents a logical rather than a physical object in the APN database, defined to map in a matrix form the feature groups availability against the Rover models, either maybes (or optional) or must, for the part description which this entity is linked to (for further details see appendix 5).

The statistical procedure of the system as described in the previous chapter refers to one EJA18V tape, consequently a single current model. This is therefore repeated for the rest of the Rover vehicles. However, still further statistical analysis is performed by the system, invisibly to the user, on the "knowledge" (design engineering rules) already acquired and stored at the beginning. That it happens from two different perspectives:

Firstly, all the secondary-description-IDs of the same primary-description-ID entity are checked against the stored values of the has-must-features attributes. If a must feature group is a must feature group for every secondary-description-ID then it is inferred that this feature group it is a must feature

group for the **primary-description-ID** as well, in reference to the specific model. Similarly, **must** feature groups of all **primary-description-IDs** which meet the conditions of this rule are assigned as **must** feature groups to the **family-within-SSSVPG** they belong to. The same process is repeated for all Rover vehicles, individually.

Secondly, similar logic is applied to the **maybe** and/or **must** feature groups information stored in the APN database but with reference to all Rover vehicles. That is, if a **must** feature group is a **must** feature group for all **secondary-description-IDs** belonging to one **primary-description-ID** for all Rover models, then it is inferred that this feature group it is a **must** feature group for the **primary-description-ID** as well, for all vehicles.

For example, the "**B51: seat-face material**" is a **must** design feature group for all Rover vehicles whenever the engineer specifies the usage statement of their front seats. This is represented in the database with the insertion in the empty value of the "**has-must-features**" attribute of the "**seat-front complete**" instance of the **primary-description-ID** entity, "**B51: seat-face material**".

The same process is repeated with the rest of the entity types in the VPG group ie. **family-within-SSSVPG** and **SSSVPG**. eg. feature groups which meet the condition of *wholeness* "travel" upwards in the VPG hierarchy filling the empty slots of the **has-must-features** attributes of the nodes. The values of the **has-maybe-features** are initialised with the collection of all the optional feature groups lower in the hierarchy.

seat-front complete

	B51	B44	B68	B45	B69	B53	B38	B76	B25	B51	B45
Rover R8	X	X manual -passr -driver	- X electric -passr -driver manual -passr -driver	O	O	O	-	-	X	X	O
Rover 800	X	-	-	-	O	-	O	O	X	X	-
Metro	X	X manual	-	O	-	-	-	-	-	-	-
R6	X	X manual	X manual	O	-	-	-	-	X	X	-
Mini	X	O	O	-	-	-	-	-	-	-	-
...											...

HEAD RESRAINT-COMP-FRONT

KEY: X Must feature group for the specific Part description in the box (if any), otherwise Must feature group for the whole Primary description
 O Optional feature group
 - Not known information for this combination of model and feature group for this part description

B51: Seat face material
 B44: Seats-reclining
 B45: Seat-type-front
 B68: Seat Operation
 B69: Seat-heating
 B53: Seat side rolls ...

Figure 73: The result of the statistical analysis of both "seat-front complete" and "headrestnt-comp-front" by ROOVESP

R6	X Body colour	O	-	-
Rover 800	X	X assy-ext driver assy-ext driver	X	X
Metro	X	X	-	-
R8	X assy-ext driver	-	-	-
	B15	B12	B14	B68

KEY:

X Must feature group

O Optional feature group

- Not known information for this combination of model and feature group
for this part description

B15: EXTERIOR MIRRORS
 B12: EXTERIOR MIRROR FINISH
 B14: EXT MIRROR GLASS CONDITION
 B68: SEAT OPERATION

Figure 74: Design engineering knowledge for EXTERIOR MIRRORS derived by the statistical analysis of ROOVESP

The result of the statistical analysis of the system for 5 Rover vehicles, the Rover R8, Rover 800, Rover R6, Mini and Metro for the VPG groups of the exterior mirrors and front seats is shown in the figures 73, 74.

The development of the statistical procedure uncovered peculiarities in the data base that had not come to light previously. Not all the part descriptions of the APN catalogue appear in the data of an EJA18v tape ie. a single model. Consequently, there is insufficient information for a thorough statistical analysis of all part descriptions of a model. This happens because of the highly human oriented way the current specification system in Rover works. The Specification Services people in order to make conceptually clear to the engineers and/or auditors the nature of the part they design or specify, have entered feature descriptions in the part descriptions.

For example, consider "seat-front complete manual" being the part description chosen by the engineer from the APN catalogue to describe the part he designed for the Mini model. It may happen that if in the future another engineer designs an electric seat for the Rover R8, the Specification Services people will view the two seats as significantly different and create a new part description for the new part: "seat-front complete electric" or "seat-front complete electric-driver". Apparently, the new part description will not be referenced anywhere in the lowest in specification Mini model, whereas it will definitely appear in the data for the Rover R8. In addition, the "seat-front complete manual" part description may exist in the data of the Rover R8,

as well, concerning a derivative of the model. This explains the highly unstandardised data of the EJA18V tape mentioned in the section 8.2.2.1.4.

The proposed system overcomes such inconsistency in two ways.

Firstly, the way the validation algorithm is implemented, the statistical procedure specifically links every single part description with every model in two dimensions:

- (i) known information from the past, for this part description for this model, or

- (ii) unknown information from the past for this part description for this model, but default consultancy from the rest of the part descriptions belonging to the same **primary-description-ID**.

Secondly, for the purpose of documenting information for the company, the statistical procedure analyses the data of the EJA18V tape into a higher abstract level, instead of the part description (ie. secondary-description-ID). This is the **primary description** of the part. In the seats example this is the "seat-front complete" rather than each part description "seat-front complete electric" or "seat-front complete electric-driver" or seat-front complete manual etc. individually.

The results of the system, in particular for the front seats of the car, have been checked and approved by Rover's seat engineers with enthusiasm (discussed in the following) [100]. This was as anticipated because:

The general idea behind the adoption of the statistical analysis of the tapes of the Rover's finished products was that it was

bound to guarantee reliable results for two reasons:

Firstly, the analysis occurs in the lowest level of the Company's information, actually the functional level ie. the physical parts of the vehicle, and

Secondly, the EJA18V data represents finished products that have been available in market long enough to not have specification errors.

Notice, the whole system works in a **close loop** methodology. The statistical analysis procedure (or experience link) needs the validation algorithm to infer its knowledge and the validation algorithm needs the statistical results (or experience link) to associate the part description with the feature groups that specify it in order to generate its usage statements (figure 69)

Finally, the information created by the statistical procedure is vital for the company, as it has not formally existed anywhere up till now. It is therefore intended by the Product Management department to be used in the future by Rover as reference source for a project (within PROMS or IBOMP which are discussed later) that would appropriately update and unambiguously document it and which it could support the PSC.

8.3.2 **maintenance of the experience link**

New feature groups are always compiled by Specification Services

when a new derivative is launched in the market and sometimes old ones are deleted. Part descriptions are updated in a similar manner by Specification Services.

ROOVESP, having foreseen such needs to maintain dynamic updates in Specification Services, supports a maintenance mechanism. The maintenance mechanism represents a menu driven interface which at the lowest level interacts with the APN catalogue and updates the values of the attributes of the part description objects which maintain the maybe (optional) or must feature groups or negative features. It also updates the AFC of the models when the EJA18V tapes are changed by Specification Services (on average once every nine months). Additionally, because the maintenance mechanism deals with a real data base password protection lockings can be implemented to prevent corruption of the data or the implemented design engineering knowledge.

From the software design point of view the presentation type facility of object oriented programming has been used. The user can enter only feature groups or features and part descriptions whose format is compatible with the K87 data base and APN catalogue. A default option for the generation of must and maybe feature groups of a newly entered part description exists based on the old inferences of the specific sub-tree of the VPG hierarchy to which it belongs, as well as help facilities.

8.4 TESTING

The validation algorithm has been tested throughout its development period by an auditor from Rover. Furthermore, the algorithm has been tested for exterior mirrors and front seats for both Rover R8 and Metro vehicles by two of the most experienced auditors in Rover and showed that the results were correct. The validation algorithm has been developed in a modular manner with a series of subroutines performing tasks such as the combination of the features, the search for the intersection in their restrictions (ie. base features, territory, combinations restrictions), the assignment of the PLUS and MINUS qualifiers in the usage conditions, etc. These subroutines have all been tested for accuracy individually so it could be expected that the overall algorithm would work correctly and it did. For more details see appendix 9.

The testing process showed that the algorithm can take less than 15 seconds to reach (actually to create) the specification of a part with three feature groups, whereas an auditor would need about an hour of manual validation. Time is reduced, as well, considering the time required for the auditor or engineer to write down all the combinations found for all restrictions and territories, whereas this is now done automatically by an external device ie. printer.

Additionally, in its planning function, the system takes less than two minutes to interact with the auditor or engineer for the confirmation of the model, the feature specification requirements, negative features etc. whereas the manual approach

takes more than one and half hour. This is because the engineer or auditor has to login in PIMS, choose the Part Description (the time for Automating Part Numbering does not account), consult both Features List and Base and Additional Features Chart and finally by indexing collect the scope of its search within the AFCs.

Most importantly, in real life terms, time is reduced considering that the end user is the engineer and not the auditor. This avoids the need for communication between the Component Engineering and Auditing departments with the redundant time consuming consequences that may occur such as absence of either party, misunderstanding etc.

Rover has estimated that the manual validation of a specification package takes an average of 5 working days. Using a complete system to support the PSC and the Audit function as proposed by this thesis, it is estimated to require less than a single working day. This results in at least an 80% reduction of validation time and considering the other time consuming factors which mentioned above a drop of the 90% of the overall time in the specification of the product is a realistic objective.

This, naturally would increase drastically the design time spent in Component Engineering, as now the time spent for validation from the design engineers would be decrease to 90%.

8.5 SUMMARY ON THE CHAPTER

In this chapter the design and implementation of **ROOVESP** was discussed. **ROOVESP** was implemented in three different phases:

- (i) a prototype of the system (a validation algorithm) with temporary Object Oriented database (**FLAVORS**)

- (ii) then the Meta-knowledge component of **ROOVESP** was developed to transfer design engineering knowledge in the prototype system

- (iii) the prototype system was implemented to a working system for Rover with a real Object Oriented database (**Statice**) to support its processes, maintenance of the knowledge it possesses.

In particular, the Meta knowledge element of **ROOVESP** is of great importance because it acquires design engineering knowledge for the company, from past experience, and which has been proved to be of high reliability. Its importance increases when it is considered that such knowledge is not formally documented anywhere in automotive industry, in general.

ROOVESP, is one of the few systems in software literature that supports real Object Oriented database in its operation (ie. **Statice**).

However, the most important characteristic of **ROOVESP** is that it is self-contained. That is, each major component of the system needs the other for the system to work. For example, the validation algorithm needs the experience-link (design engineering knowledge) to work, but also the experience-link needs the validation algorithm to acquire its knowledge.

In this chapter, **ROOVESP** dealt only with the first phase of the audit process; the validation (actually, the creation) of the usage conditions of each part in the specification package. The next chapter continues the description of **ROOVESP** and covers the last two phases of the audit process.

9. FURTHER IMPLEMENTATION OF THE SYSTEM AND THE PSC
(second and third phases of the Audit process)

The system, described earlier allows for cooperative (man-machine) problem solving in the generation of usage statements for the parts of the vehicle. It uses the auditor and/or engineer's relevant knowledge obtained by statistical analysis of the old data and it applies it systematically in order to specify usage conditions of newly designed parts. Its major objective is to guarantee that parts are specified with the proper feature groups ie. feature groups that represent their design, and also those be combined and validated in their features level. Top key points of the design to date are:

Firstly, ROOVESP tackled only the first phase of the Audit function.

Secondly, it concerned only with the specification of the parts in the abstract level ie. for a given part which are the features (feature groups) which specify it for a specific model and under what restrictions.

In real life terms for the company it could be viewed that the system up till now deals only the new parts ie. parts whose design need change. For example, given that a front seat of a car needs to become heated, which are the features with which the engineer must specify it based on his experience and the new situation of the company. In addition what are the restrictions of this new design. However, in some cases the usage condition

of the new part may **affect** the usage conditions of the carry over parts ie. old parts which transfer automatically with the new design such as overlay, squab, cushion, seat frame etc. ROOVESP does not update the PIMS database in these cases. This is done manually by the specification people in a similar way to that of the IPL project.

The reasons that the design of the system focused on these principle are:

(i) the first phase of the Audit function represents more than the 90% of the overall audit process.

(ii) the generation of the usage conditions of only the new parts simplifies design and enables the Auditing process to be separated from the whole Product Specification Concept. The details of the PSC can be added afterwards and complete the system (design in a *hierarchy of abstractions*).

In this chapter, ROOVESP deals with the PSC in the company rather than just the Audit function. Notice the implementation of the second and third phases of the Audit function coincides with the implementation of the system to tackle the whole PSC.

The first part of this chapter is concerned with the expansion of the system beyond the above boundaries. This is discussed in two stages:

In the first stage introduces the original intentions for the further implementation of the system to cover the second and third phases of the Audit function.

The second stage discusses a new approach which further investigation of Component Design and Specification Services has

revealed, the *intelligent networks*.

The latter, though not rejecting the former, has been adopted by the author and developed to a prototype phase. That is because of two reasons:

Firstly, The new approach expands the current system out of all its boundaries and only a single design of a system and software support is required. In addition, the same design can be used to tackle the PSC. Secondly, the new approach automates fully the Product Specification Concept.

contribution to knowledge:

- Tightenig up of the theoretical background (ie. boolean logic, feature combination of alternatives, etc) upon which new parts are specified on the basis of the existing ones, within the PSC.

- Introduction of computer intelligence in the BOM area of the PSC, based on the revised theoretical background.

- The creation of a working system which fulfils the whole area of the business (PSC and BOM) and, also, meets the pragmatic needs of the company.

9.1 ORIGINAL THOUGHTS ON THE FURTHER IMPLEMENTATION OF THE SYSTEM

9.1.1 *Parts quantities (second Audit phase)*

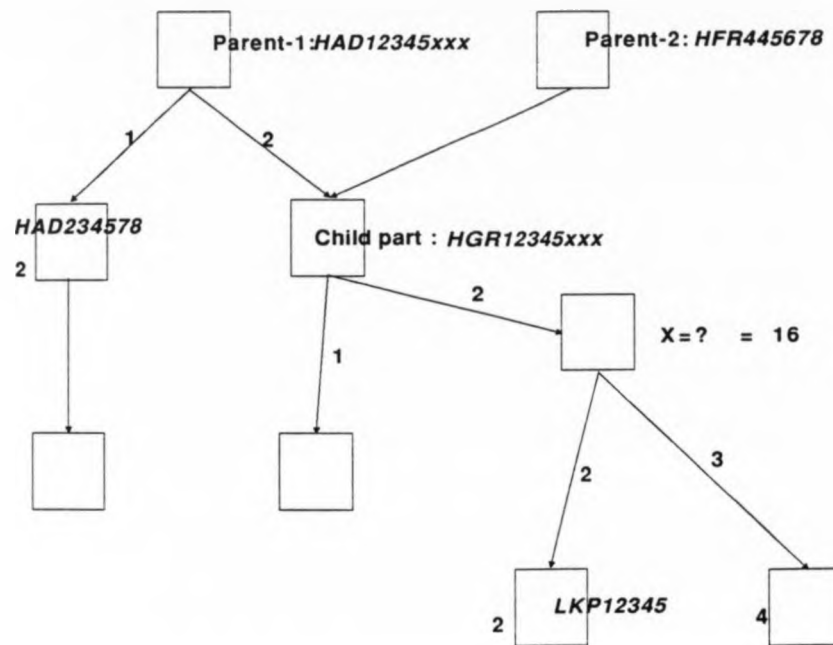


Figure 75: The calculation of QUANTITIES.
(per vehicle and per assembly) as originally thought.

The second phase of the Audit function was originally thought to simulate the auditor's job. The auditing of the **quantities** of the parts per assembly or per vehicle (actually, per usage statement using the correct terminology) is easy - at least in the way it currently operates.

Assume all the usage conditions have been validated for every single part in the first Audit phase of the *specification package* (The system internally has created *objects* each of which represent a specification document in the package). Then the engineer or auditor can link the parts (*objects*) at the user

interface level by using the information in the "NEXT ASSY" field (see figures 23, 24). The system updates the *next-assy* attribute of the objects and creates the internal links in the specification package.

In parallel, the engineer or auditor can enter the quantities required for each part in the assembly. Then the system can graph the assembly tree of the package and derive the missing quantity information at each level, following alternative routes if required (figure 75).

More specifically, the system tries to solve the equation at each level of the specification package tree:

$$\begin{array}{l} \text{Quantity-in-lower-level} = \text{Quantity-per-assembly} * \\ \text{Quantity-in-higher-level} \end{array}$$

If more than one variables happens to be unknown in the equation, the system can search neighbouring nodes and recover the value of either of them before it solves the equation (*a prototype of the process has been developed*).

Notice, the above process has been presented in a simplified form. The system in reality has to work out which link is which between the various usage statements existing in each of the specification documents. That helps the code in the "USAGE SEQ. No." field (figure 23, 24).

9.1.2 *relationships among the parts (third Audit phase)*

The system uses the structure of the network that has been created in the virtual memory of the computer during the second Audit phase. Procedures can validate the Usage Conditions of each part of the network on the principle that the usage condition of a part it must result in the Boolean combination of the usage conditions of its parents (figure 75).

9.1.3 Other validation procedures of the Audit function.

There exist other minor checking procedures that the auditors apply in the validation of the specification packages coming from Component Engineering. These procedures have not been mentioned during the auditing example in chapter 5 in order to concentrate on the major concern of the system design: the correct usage specification of each part in the package.

One of the procedures validates the "sourcing" of each of the parts. For instance, the CAC code in figure 23 under the heading "SOURCE" indicates that the "seat-front complete manual" assembly (A) came from the Canley (C) division and went to Canley (C). Implicitly that means that it was brought from outside as a complete assembly. CDC (fig. 24) represents a subassembly brought from outside. Consequently, not only the sourcing code must be correct but the logic of the various information fields within the specification document must match, as well. eg. If there was a value in the "NEXT ASSY. NO" field of figure 23 this would contradict the sourcing field (CAC) which represents top assembly. Another validation procedure is concerned with the

information, if it exists, in the "ALT/ACCOM." field, which tackles accomodation numbers in the specification document (see [17] for more details).

There exist some even more minor procedures. The problem which arises is that the functionality of all these minor procedures is not related to the part itself but also among themselves, which makes it hard to code in a program in structural manner. Consequently, automatic validation of the specification document on this level would be difficult.

9.1.4 Manual update of the old parts in PIMS

As already mentioned ROOVESP has up till now dealt with the specification of newly designed parts. However, the usage statements of old parts can be affected by the insertion in the data base of new ones. This is because old parts represent a condition of the company in the past and they must be updated to the new market characteristics in order to discriminate themselves from the newly designed parts.

For example, if an old part represents the standard seat of the Metro model with global specification (ie. "ALL Metros") and Rover introduces a new type of heated seat, then the global specification of the old part has to be changed to reflect the fact that it is not heated (ie. "Metros with NO-HEATED seat"). (The details of the above situation are discussed in the

following section.) At the moment such changes are supposed to be done manually by the Auditing or Specification Services department, in the same way the IPL system operates.

In summary, the system as it has been discussed till now, implements the major concern of the Audit function: "get the usage statements of the parts, right first time" [17]. This represents more than the 90% of the Audit problem. This section has discussed the original thoughts for the system's further implementation. It has also shown the immense complexity of the problem, considering all the minor audit procedures involved and the dynamic association of all parts new and old. Although the validation of the parts usage statements is obtained automatically, at least one stage of manual manipulation of the data is required for the complete process of the Product Specification Concept: the update of the usage statements of old parts.

The next section proposes a new design methodology accompanied with a prototype software support that completes the existing system and potentially could automate the whole Product Specification Concept in Rover.

9.2 INTELLIGENT NETWORKS

9.2.1 *Existent implementation*

(THE SECOND AND THIRD PHASES OF THE AUDIT FUNCTION)

Up till now the design of the ROOVESP has been implemented around the concept of the **part description**. Part description in the context of the system means the high level description of physical parts with similar physical appearance (ie. nimbus mirror, body colour mirror, remote control mirror). This section approaches the Specification problem from a different perspective, the **physical part**. It builds up usage condition statements based on the Bill Of Materials (BOM) of the designed parts.

In order to gradually introduce the new methodology and the concepts involved, an example has been chosen. For commonality, this example uses the same context used previously ie. the specification of the two front seats of the Rover R8 model. Furthermore, among the numerous subassemblies of the vehicle's seat assembly, only the *squab* subassembly is discussed in the example (figure 76).

Figure 76 illustrates a simplified version of the assembly of the front seat of a car. Usually a seat assembly is made of 50 to 60 components [5]. Of importance in the example is the *squab* subassembly which among its other subassemblies and parts gets an *overlay* subassembly.

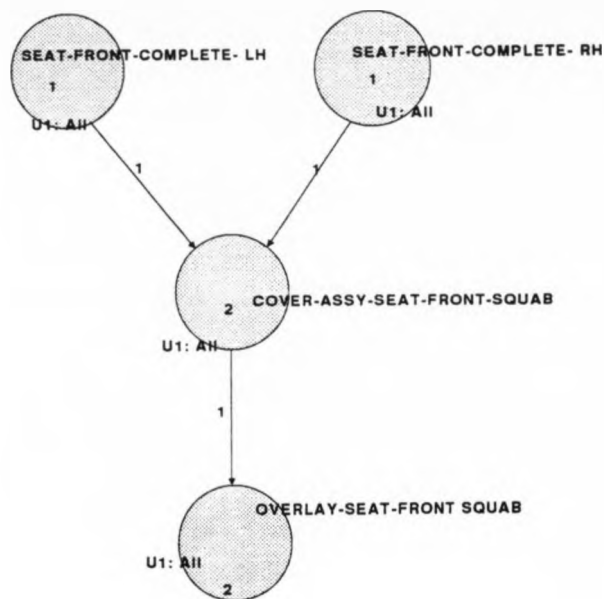


Figure 76: a simplistic view of the assembly of both front seats of a vehicle.

As far as the squab of a front seat is concerned, it is irrelevant which position the seat is in the vehicle and it is fitted to both seats (Design Engineering knowledge) as shown in figure 76.

Consider, in the initial vehicle design that the same seats are offered to all R8 derivatives, irrespectively of trim levels or other Base Feature specifications, body, drive etc. The original specification of the two seats is:

ul: "offered to all R3 derivatives".

The same therefore applies to the *squab* and *overlay* assemblies

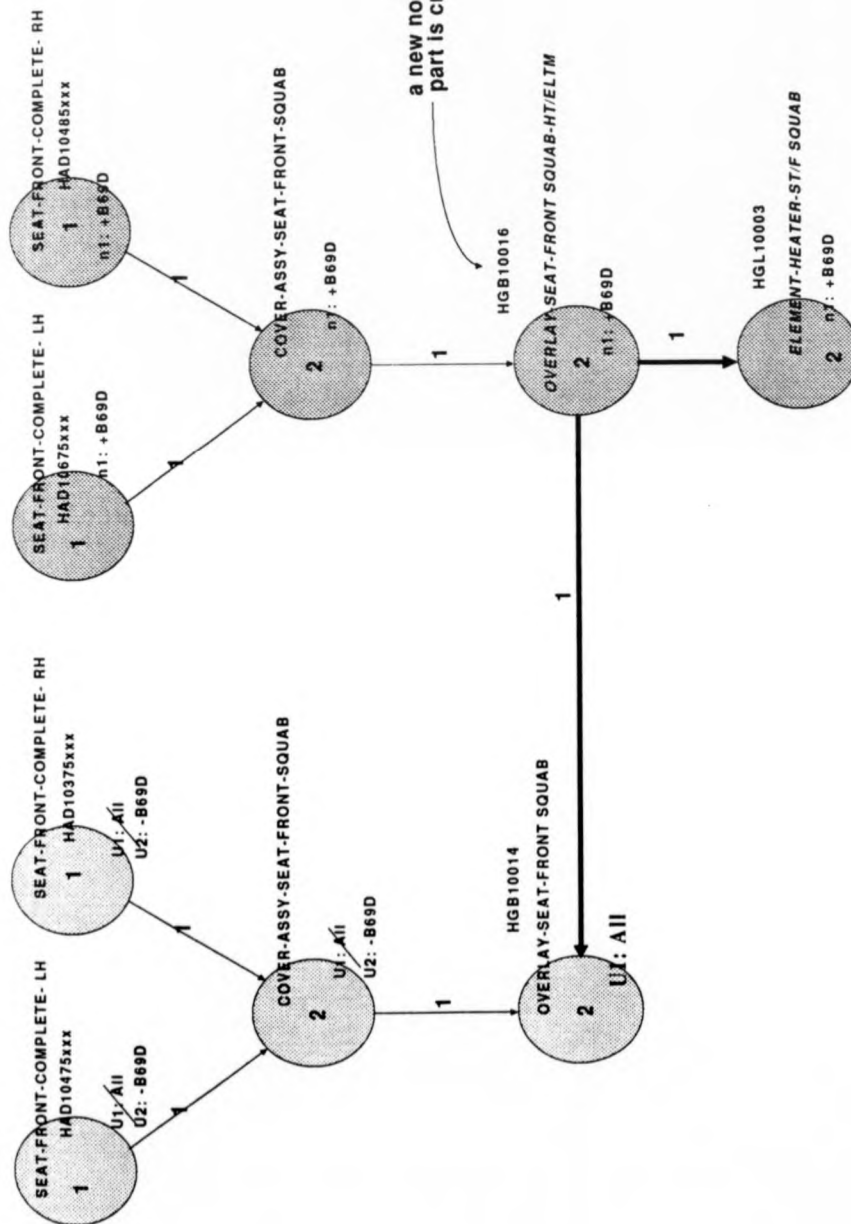


Figure 77: The 'heating' feature is entered into the front seats assembly

which share common usage condition (solid lines).

The quantity in the arrows indicates the quantity per assembly of the lower subassemblies or parts in the hierarchy. Notice, as well, that each assembly or part has its own part number that uniquely identifies it in Manufacturing.

Let's assume that after some time Rover decides to offer heated front seats as an option on the Rover R8 model. The new heated seats are designed in such a way that new overlays are fitted in their assembly structures. The rest of the parts or subassemblies do not change. The new overlay assemblies are consistent with the old overlay subassemblies plus a new heating element. The new situation is shown in figure 77. It is Manufacturing policy within Rover that whenever a physical part or assembly (ie. seats) changes in one of the products that drastically affects the new design of the product, then all the part numbers of the physical parts or assemblies higher in the hierarchy have to change, as well. In the example, because a new overlay is fitted to the new assembly, new part numbers for the cushion and for both right and left hand side seats are created. The creation of new part numbers by the engineer (in reality automatically from the Rover's current system) means the insertion of new parts or assemblies in the Manufacturing. In this case, heated seats. The usage conditions of all the physical parts of both assemblies have to change, now, to reflect the new Manufacturing situation.

Firstly, the usage conditions of the parts of the original assembly have to change. This is so that Manufacturing can

distinguish the unheated seat assemblies at the production level. The usage condition of both the right and left hand side seats in the original assembly changes to u2: -B69D. The "-B69D" code means "Offered to the Rover R8 derivatives which get no-heated (-B69D) seat assemblies. This is because it must now become explicit that the original seat assemblies are not heated, as new heated seat assemblies exist in Manufacturing. The new usage statement **overwrites** the old one. Overwrite is used in this context to mean that the link of the old usage condition to the physical part becomes obsolete. From the top of the assembly structure the *discriminating* characteristic traverses downwards to the lowest assembly level, the physical parts. The usage condition of the original squab subassembly now becomes the Boolean combination of the usage conditions of its parent nodes. ie. u2: -B69D.

Secondly, the heating characteristic of the usage condition in the new assembly is explicitly specified all the way downwards in the hierarchy, as well. That is n1: +B69D, which means that the part either seat or squab or overlay or the heating-element is "offered to the Rover R8 cars with the heating option". However, in the case of the original overlay, the usage condition of the part does not change. This is because the Boolean combination of its parent nodes result in the *empty set*. In other words, the original overlay is fitted to both front seat assemblies, irrespectively of the insertion of the heating discrimination in Manufacture. If the engineer or auditor, specifies this part with either "- B69D" or "+ B69D" usage conditions, then he falls to the *overspecification* error (for

more details in this subject see [17] and the following section 9.2.3.

Now let's assume that an order for a number of Rover R8 cars with heated front seats is received by the company. The **Usage-condition-to-parts** system depends on the correct specification of the parts in PIMS in order to retrieve the right components in the right quantities and build up the assemblies on the manufacturing line. In the example the **Usage-Condition-to-Parts** system will retrieve parts from the data base (PIMS) whose usage conditions *match* exactly the order specification (ie. heated) or broader specification areas. In parallel, the quantities of each part which are linked to the selected usage conditions are retrieved, as well. The result of the search is shown below:

ORDER: "Rover R8 cars with heated seats"

<u>PART DESCRIPTION</u>	<u>SELECTED CONDITION</u>	<u>QUANTITY</u>
seat-front complete manual-LHD	+ B69D	1
seat-front complete manual-RHD	+ B69D	1
cover-assy-seat-front SQUAB	+ B69D	2
overlay-seat-front SQUAB-HT/ELMT	+ B69D	2
overlay-seat-front SQUAB	ALL	2
element-heater-ST/F squab	+ B69D	2
...

In this way, Manufacturing can estimate the volume of parts

required to build the specified number of Rover R8 cars with heated seats by multiplying this number with each of the quantities of the chosen parts. The source code (see section 9.1.3) tagged to each part indicates whether the part is going to be brought from outside, hence manufacturing can authorise purchasing of the parts for the project or build in house which starts the operations for the establishment of the assemblies needed.

The example continues, with the introduction of sport-style seats as an additional option to the Rover R8 model (the sports style of the seat is achieved by adding a new lumbar support to the basic frame of a seat). For simplicity only one of the seats is considered in the example, either left or right hand drive.

The insertion of more than one *feature* in the seat adds complexity in the logic of the methodology discussed above. In addition, a new factor has to be taken in account for the further consideration of the specification of all the parts in the seat assemblies; Rover's marketing policy.

9.2.1.1 ROVER'S MARKETING POLICY

Let's consider that Rover for marketing reasons decides to offer the sports style option only to non heated seats. Consequently, a new assembly is entered in Manufacturing the "+ sport style - heated" seat. The old assemblies "- heated" and "+ heated" seats have to update their usage statements with the "- sport style"

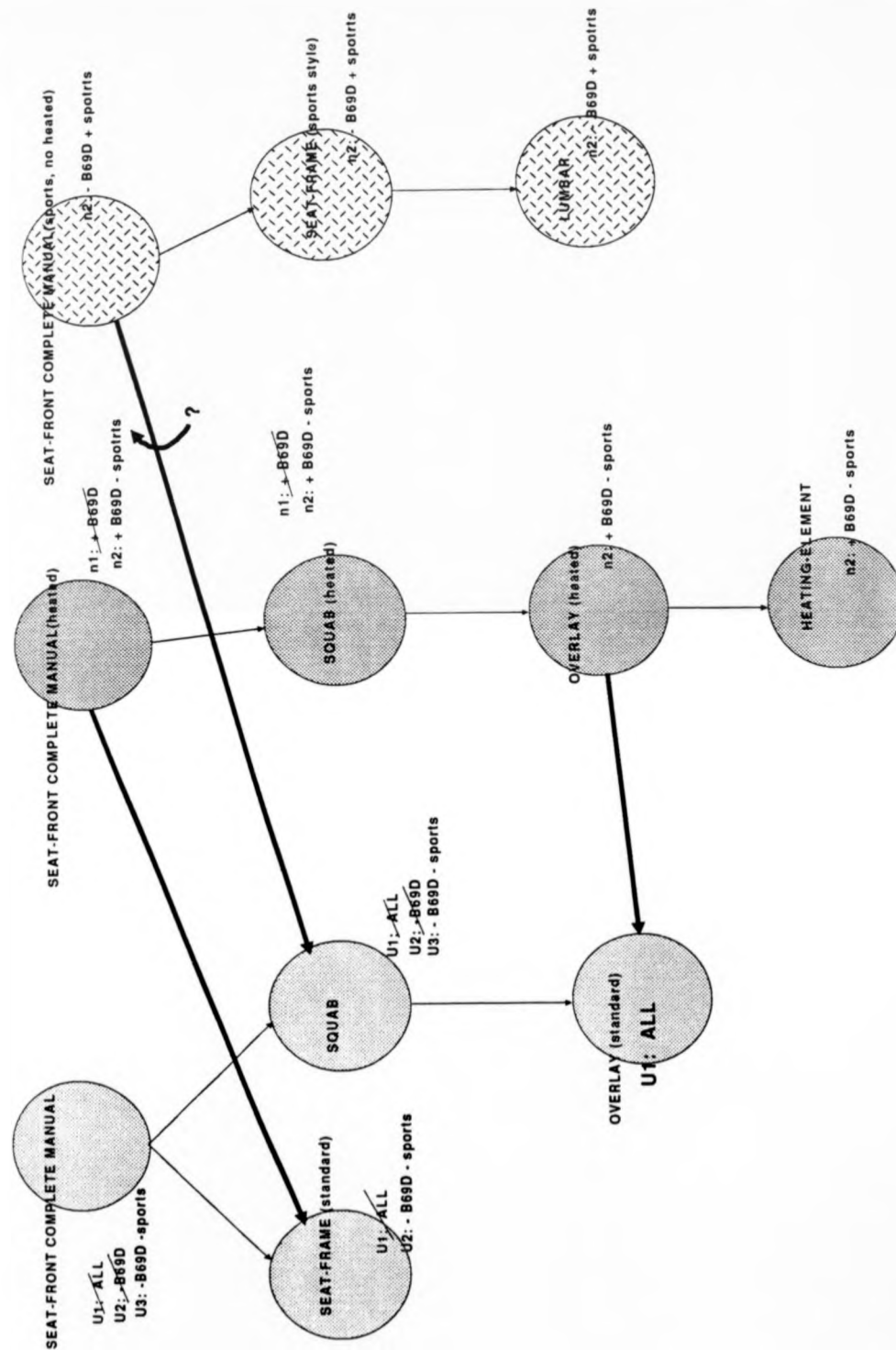


Figure 78: Rover's policy decides to offer sport style seats only with non heated ones

condition in order to reflect the new Manufacturing requirements. This will indicate that the original seats are not of sport style.

The update of the new Manufacturing condition is shown in figure 78.

On the other hand, in the case where the sport style feature is offered irrespectively to all seats in Manufacturing, this creates four new potential assemblies:

" + sport style - heated"

" + sport style + heated"

" - sport style - heated"

" - sport style + heated".

Notice, the existence of the original seat assemblies, as well:

" - heated" and

" + heated".

The latter, as mentioned earlier, have to update to

" - sport style - heated" and

" - sport style + heated".

Some, of these assemblies match, hence no new part numbers need to be created for the common ones.

The difference, from the previous situation is the insertion of an additional assembly " + sport style + heated" which was not valid before. As it can be seen in figure 79, the different usage condition of the *non heated squab* (and all of its subassemblies) now reflects the different policy of Rover, ie. to offer the sport style option to **all** seat assemblies in Manufacturing.

Up till now the analysis of both the new and old assemblies in Manufacturing, refers to changes in the design of parts which make them not *interchangeable* There is a company definition for interchangeable parts in Rover:

"A part is interchangeable when it can be interchanged in production and in service new for old, old for new both physically and functionally including performance, appearance, safety and legislative requirements such there is no requirement for discrimination between before and after conditions".

Interchangeability is a complex concept in an automobile. This is because interchangeability can be viewed from many perspectives, according to the philosophy of the various departments within Rover such as manufacturing, customer services, specification services, etc. The study of interchangeability is beyond of the scope of this section. Further details can be found in [17].

In the context of this chapter interchangeability is defined from the design engineering and Manufacturing views. In the example above, for instance, the selection of a heated-squab or a no-heated one naturally affected the design of the top seat assembly. Consequently, heated and not heated seats are not interchangeable. In a tyres example, however, it does not really makes much difference to the design of the wheel assembly the brand of the tyres used eg. Michelin or Pirelli. Consequently, wheel assemblies with different tyre brands are interchangeable. In the case of interchangeable parts, the part numbers of the parts higher in the assembly structure that are linked to them do not change (no new assemblies are created).

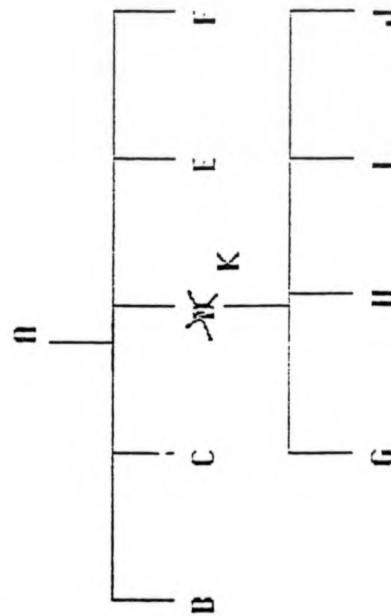
Figure 80 illustrates a case of interchangeability.

In summary, this section is concerned with the engineer's job within the Component Engineering department. Notice, the way in which usage conditions are affected by changes in the data base and shown in figures 76 to 79 amended by the author, represents his own understanding of how the engineers within Rover *should* work. However, the design engineers work only with the general concept of this way ie. to specify newly designed parts or assemblies using the already existent specification information. In reality, they don't apply any thorough procedure, such as the one described above. Instead, they obtain a hardcopy of the VPG data (parts) that are relevant to the part which they have designed and manually update their usages. This is done by crossing out with pen some old parts and usage statements, and inserting new parts (new part numbers created automatically from the Rover's Automatic Part Number system), new usage statements or combining old ones.

The correctness of the application depends solely on the engineer's knowledge and experience in the area of his design. That is, if the engineer *knows* of the existence of parts in the VPG area which may represent similar *functionalities* in some of the assemblies of his design. The nature of the problem becomes even more subjective when the variations that exist for a single part description are considered and may cause the selection of the wrong parts in the VPG area. Additionally, the engineers do not use a specific procedure in the specification of the parts.

Update Transactions based on Engineering Statements

Part D is replaced by part K in the following assembly structure :



One engineering statement (above) performs the following system transactions...

- D's two lines of usage (say) are obsolete,
- K has a fitment to this model range,
- K has two current lines of usage,
- Parts G,H,I and J no longer go into part D,
- Parts G,H,I and J now go into part K,
- Part D no longer goes into part A,
- Part K now goes into part A (interchangeability assumed).

Figure 80: The update of an existing assembly with an interchangeable new part/assembly.

They are dependent only on their knowledge to specify them or they pass the corrected hardcopy file to the Auditing department in order to compile it to usage statements.

The above examples have demonstrated that a new approach to the part description is possible. By no longer using an abstract entity but the actual **physical part** in the assembly structure, it is feasible to create usage conditions and quantity relationships of the parts. It appears, as well, that 'tidying up' such a process seems to maintain its **uniformity** throughout the various levels of its application to the update of new design features in the component parts (ie. for one, two, three or more features). The steps of its application would be determined by the *planning alternatives* that would be chosen by the engineer. ie. if the parts in the assemblies are interchangeable or not with the insertion of a new feature, the Rover's marketing policy etc. Consequently, an "algorithm" could be designed which in interaction with menu driven options could implement *intelligent networks of inference* of usage conditions and part quantities from the way physical parts are linked into the assemblies. This algorithm would complete the system discussed in the earlier chapters and extend its boundaries to fully automate phases 1 and 2 of the auditing function and dynamically update PIMS. Furthermore, it would contribute to even more precise usage conditions as the reason for the design of the new part traverses throughout the whole assembly.

9.2.1.2 EXISTENT WORK IN INTELLIGENT NETWORKS

A general purpose prototype has been developed to demonstrate the intelligent networks approach. It is actually an extension of the "nodes and arcs" example from the Lisp Lore book [2] enhanced with intelligence of the Rover's specification process. The term *network* is preferred to hierarchy because in the case of non interchangeable parts, assemblies are linked together, which looks more like a network than a tree-like structure. The prototype at its highest level supports a user interface substrate for linking physical parts (nodes) with assembly relationships (arcs). At its application level it infers quantity relationships and usage conditions of the physical parts (nodes) from the internal structure of the assembly.

9.2.1.2.1 *User interface*

The intelligent networks prototype is implemented in its own *framework* interface, with commands of its specific application. These are update of Part Descriptions, Part Numbers, quantities, Usage statements, the association of the parts (nodes) with assembly links (arcs) or the deletion of an existent link, the move of the parts around the screen and finally the image representation of the parts in the data base.

The commands can be applied in three different ways: by typing

the first letters, or from the menu window or with mouse gestures ie. (click together Control and mouse Right button, etc.). Notice, the user interface from the software point completes all the needs of the intelligent networks methodology ie. graph the nodes, move or delete them, or link them with arrows etc. Consequently, the user interface is a complete implementation rather than a prototype.

9.2.1.2.2 *application level*

The prototype is based on the idea that component engineers design new parts based on already existing ones. With this perspective, when the prototype is invoked by the engineer, it queries him regarding the characteristic of its application design. This is to determine if the new part is **interchangeable** with its ancestor or not. If yes, the assembly structure of the ancestor part is retrieved from the data base and displayed. The engineer then can delete and add new parts in the assembly and assembly links. If the answer is no, then the prototype displays in the screen both the ancestor assembly and a new copy of the assembly with automatically created new part numbers. The engineer then can indicate the innovation of the new design by linking the assemblies appropriately. Additionally, he can insert the usage conditions and quantities of the parts per vehicle (actually, per usage statement). This can be done automatically by the system based on similar specifications in the past. For

example, the prototype gives its *best guesses* to the engineer for the quantities of the parts in the new assembly based on the past experience. That is, it is quite possible for the new seat, for example, to always have one squab assembly irrespective of whether it is heated or not.

Notice, when a physical part is linked to another physical part at the part number level, the quantity referencies among themselves almost never change (actually in more than the 99% of the situations). The information regarding the quantities at the part number level exists in one of the files of the EJA18V tape. Consequently, such *best guesses* can be obtained automatically from the EJA18V tape; a program already exists that does it.

9.2.1.2.3 knowledge representation

The prototype has been implemented in Statice entities and represents only an example of the front seats of the Rover R8 model. The Statice schema consists of two entity types: nodes (**physical parts**) and arcs (**assembly links**). A brief representation of the definition of the schema in Statice is shown in the following.

```

(define-schema INTELLIGENT-NETWORKS (node arc))

(define-entity-type node
  ()
  (arcs (SET-OF arc))
  (xpos integer)
  (ypos integer)
  ***
  (quantity-per-usage-for-one-vehicle integer)
  (part-number string :unique t :no-nulls t :inverse-index t)
  (usage-condition string)
  (radius :needs-calculation)
  (image raster-array)...)

(define-entity-type arc
  ()
  (node1 node)
  (node2 node)
  (parts-per-assembly integer)
  ...)

```

Figure 81: The intelligent networks Object Oriented implementation in Statice.

A node includes information on the arcs that are attached to it, it has a part description which is not unique, neither is its usage statement, or quantity. However, its part number is unique

and an **inverse-index** has been defined in order to speed up the program during the deletion or insertion of nodes or arcs. Also, as the node is represented graphically as a circle it needs to keep local variables like *xpos*, *ypos* which would determine its position on the screen. Finally, the *picture* attribute is a two dimensional array (actually a raster array, see []) which keeps the scanned image of the physical part represented by the node.

The **arc** entity type, for similar reasons of printing itself in the screen has attributes such as *xpos*, *ypos* which are automatically updated according to the *xpos* and *ypos* values of the nodes in which they are attached to. The arc entity type records the nodes which it is attached to by the *node1* and *node2* attributes which represent the goal and destination nodes, respectively.

In summary, an internal representation of the figures 77, 78 and 79, for some of the objects it would look like:

```

      NODE: <# node:SEAT-MANUAL-LHD 1234567>
      arcs: (<#arc123... from seat-manual-LHD to -> not-heated-squab>)

      NODE: <#node:SEAT-MANUAL-RHD 4567892>
      arcs: (<#arc345... from seat-manual-RHD to -> not-heated-squab>)

      NODE: <#node:NOT-HEATED-SQUAB 3333334>
      arcs: (#arc123... from seat-manual-LHD to -> not-heated-squab
             #arc345... from seat-manual-RHD to -> not-heated-squab
             #arc678... from not-heated-squal to -> overlay-no-heat)
```

ARC: <#arc123... from seat-manual-LHD to -> not-heated-squab>

node1: <# node:SEAT-MANUAL-LHD 1234567>

node2: <#node:NOT-HEATED-SQUAB 3333334>

ARC: <#arc345... from seat-manual-RHD to -> not-heated-squab>

node1: <#node:SEAT-MANUAL-RHD 4567892>

node2: <#node:NOT-HEATED-SQUAB 3333334>

Notice, the numbers such as 1234567, ..., 4567892, etc. which appear in the computer attached to the Statice entities (nodes or arcs) represent the memory references of the instances of such entities (objects) inside the computer. The instances of entities in Statice are called **entity-handlers**. In reality, the computer presents the above instances with the "entity-handle" prefix to show that are Statice entities. ie.

<#entity-handle-node:NOT-HEATED-SQUAB 3333334> instead the
<#node:NOT-HEATED-SQUAB 3333334> which we used for
simplicity.

9.2.1.2.4 application program

The Intelligent Networks prototype works in the way that whenever a change occurs the whole network is updated, automatically. For

example, if the engineer deletes or inserts a new assembly link in the network, the prototype's application program firstly, reorganises the information of the *destination* node. This is, the usage statement and the quantity of such parts (node) needed per vehicle, as well as the proportions of such parts per higher assemblies. The last is graphed in the middle of the assembly arc which links the two nodes.

Secondly, from that node downwards to the network all the children nodes of the node are **recursively** updated in a similar manner. *Recursion* and *macro* functions, the two most powerful tools of the Lisp environment, are used in this operation. A similar, process occurs when the engineer updates the usage condition of a node or its quantity value.

Notice, the arc object type (assembly link) with the existence of the *node1* and *node2* attributes in its definition it incorporates *direction* in its appliance ie. *from node1 to node2*. Consequently, the validation of the insertion of an assembly link can be checked against the data base. The information of such a validation is derived from the EJA18V tape.

9.2.1.2.5 presentation of the objects

Linked to the node and arc object types is a **method**, of different functionality for each of them, **present-self**. This method determines the way the object presents itself in the user. It was

mentioned earlier that nodes are presented as circles and arcs as arrows. For the nodes (parts), in particular, their presentation method has been expanded to encapsulate the scanned image of the part in the data base. This makes the specification of the parts with the *intelligent networks* methodology more natural. Additionally, the engineer can view in detail the whole image at the COMMANDS level.

9.2.2 *Further Implementation in the Intelligent Networks prototype*

The *intelligent networks* methodology has only been applied to a single simplified example of front seats of the car and the information has been entered into the Statice data base manually. Therefore a future implementation of the prototype would be the electronic transfer of all the physical parts to Statice. This would require extension to the design of the current Statice APN catalogue database (discussed in the next section of this chapter). At the moment the prototype does not implement Rover's policy in its application routines ie. which top assemblies are valid. Additionally, the automatic generation of the updated usage statements of the old top assemblies (eg. -B69D) could be coded provided the new *design features* of the new parts are given. Similarly, the usage statements of the new assemblies could be generated.

The existing code for analysing the assembly data in the EJA18V

tape could be used to interface with the prototype and fully support its application program in two levels:

Firstly, by mapping the valid directions of association between the nodes (physical parts) based on the part description level ie. overlay is a subassembly of the squab assembly whereas the opposite is incorrect.

Secondly, the quantity relationships at the Part Number level.

Finally, the incorporation of images in the Intelligent Networks prototype slows down the performance of the whole system as the images have to be re-graphed every time that a change in a part (node) or assembly link is made by the engineer ie. changes in the usage condition, usage quantity, part description etc. A further implementation of the Intelligent Networks prototype should include the :redisplay facility of the GENERA environment within the existing code in order to speed performance.

9.2.2.1 *Interface of the Intelligent Networks prototype with the rest of ROOVESP.*

The part of ROOVESP which was discussed in the first two sections of this chapter implemented design engineering knowledge and a validation algorithm for specification of parts. It dealt mainly with the first phase of the audit process. The Intelligent Networks prototype completes ROOVESP by dealing with the second

and third phases of the audit process and with the same functionality tackles the Rover's PSC.

The first part of ROOVESP studies the general knowledge in automobiles. The main concept around which the design of this part focused was the part description. In this abstract level of a vehicle part the design engineering knowledge (feature groups) that specifies a part in the vehicle and the valid combination of such knowledge according to the company's policy (Base features, territory and combination restrictions) was important.

The Intelligent Networks prototype of ROOVESP focused on the pragmatic conceptualisation existing in automobiles. This is the need of the parts to discriminate themselves in the data base in such a way that the volume of production required for an order to be always correct. The main concept around which the Intelligent Networks prototype was designed was the physical part itself ie. how a specific physical part fits in an assembly and in what quantities.

Consequently, the two approaches are different. However, they can co-exist together and each benefit from the other. For example, the first part of ROOVESP needs the Intelligent Networks prototype to complete the quantity requirements in the specification of a part and update the usage conditions of existing parts for which reference exists in the specification package.

On the other hand, the Intelligent Networks prototype needs the design engineering knowledge of ROOVESP in order to specify the

top assemblies in the network which would drive the whole process of the prototype. Design engineering knowledge would be required as well in order to implement *directness* in the prototype ie. whether it is valid or no a specific part to be fitted in a given assembly on the screen. Additionally, design engineering knowledge would help the Intelligent Networks prototype to establish the quantities of the old existing assemblies and give the best guesses for the quantities in the new ones.

The engineer, therefore, when working on a specification package, could use the first part of **ROOVESP** to establish the usage conditions of the top assemblies or the newly designed parts (ie. "seat-front complete HEATED", "heated- SQUAB"). He then could swap to the Intelligent Networks prototype if reference for already existing parts appeared in the specification package, in order to update their new usage condition.

Working with the Intelligent Networks prototype, the engineer would have a first glance at the **feature groups** which govern the new specification conditions in the assemblies for all the parts. Then these feature groups could be passed to the validation algorithm which would generate all the possible feature combinations of the new design situation.

Notice, in the examples which were used above to describe the Intelligent Networks prototype, we considered the simplest case where a feature group has only one feature. For example, it mentioned that a seat can be of sports style (or heated). However, they may exist more than one availabilities of sports style (or heated) seats as there are various types of seat

material ie. zenith, leather etc. For this reason the validation algorithm is needed to audit the combinations of the features of the feature groups which are participating in the specification of each part in each assembly.

In summary, the validation algorithm and the design engineering knowledge, which was automatically transferred from past data, interface with the Intelligent Networks prototype to implement ROOVESP. In this way both the Audit function and the PSC can be tackled with a single system and a single database, Statice. The enhancement of the design engineering knowledge, in particular, (discussed later) will show that ROOVESP as well as being a working system can potentially be a complete real life system for Rover which would automate its PSC.

9.2.3 *Testing of the Intelligent Networks prototype*

It was mentioned that the Intelligent Networks prototype represents - after investigation - the author's understanding of how people in Rover should maintain the usage conditions data base. However, although indications of such a process seem to appear in the company, no standard way of relating the specifications of the parts seems to exist. The investigation therefore focused on whether such a process has a logical basis and can be proved to be correct applying mathematical rules ie.

boolean algebra.

The process when conceptualised was presented to a team of two experienced auditors from Rover. A series of sessions followed where the process was studied manually and it was found to work. For example, the steps of its application (ie. one, two, three or more features) showed uniformity and therefore it could be **algorithmised**:

(i) the creation of all the combination of the possible top assemblies

(ii) the logical reference of the new created assemblies after combination with some of the existing ones. In addition, the application of Boolean algebra to the generation of the usage statements of the child part from its parent assemblies was shown to work correctly all across the network and represent realistically the situation of the company. Only one problem appeared but it was considered of minor importance indeed it could well be considered not a problem at all but to represent more realistically the needs of the company. This is discussed in the following example:

Consider figure 78. The part "overlay-seat-front SQUAB-HT/ELMT" (heated squab) is shown with a usage condition of "+ sports - B69D ", as derived by combining the usage conditions in the network. In automotive industry this means that the design of a SQUAB part (irrespective of whether it is heated or not) depends on whether the seat is heated or not and on the style of the seat (ie. if it is sport or standard etc). However, this is not fully true as from the component design point of view it is not

correct. The squab part is not dependent on the style of the seat (the frame of the seat depends on this feature group). This is called overspecification within Rover. It occurs because of the two different philosophies which have been applied in the first phase and third and second phases of the Audit process. In the first case it was of importance that automotive design engineering knowledge be used in order to support the application of the validation algorithm. In the latter the Intelligent Networks prototype describes the pragmatic situation of the company. However, this situation is not viewed as a mistake by the auditors of Rover for the following reasons:

(i) When the Usage-Conditions-to-Parts system applies to the data base for a specific order (ie. + sport style seats - heated), the correct parts are pulled out and with the correct quantities. Consequently, the forecasting for the volume of the parts in production required will be always correct.

(ii) The usage conditions are updated in line with Rover's policy and in the most common situation, where Rover offers a part with all the possible combination of its features, overspecification disappears.

(iii) The existing design engineering knowledge acquired from statistical analysis can always be used to clear the usage conditions data base from overspecification, if this is required.

In summary, the Intelligent Networks prototype when tested manually for its uniformity and correctness was proved to work. Thus it can be algorithmised in a structural way and its process

can be repeated with the insertion of new features. Its results, as well, although they sometimes may not be strictly compatible with the design engineering knowledge in automotive industry, are correct in a pragmatic manner for a real life system for the company.

The code of the Boolean algebra in the Intelligent Networks prototype has been implemented in some extent and it was proved that not only manually but also electronically correct results can be derived.

In this chapter the Intelligent Networks prototype was discussed. The Intelligent Networks prototype introduces a slight different conceptualism in **ROOVESP** in order to tackle the whole PSC in the company ie. the physical part and assembly structures (photos A4 to A7 in appendix 10 show a work sample of the Intelligent Networks prototype).

The next chapter investigates the further implementation of **ROOVESP**.

This chapter discusses the further implementation of ROOVESP in two aspects: from the software point of view and the further design of the system in order for it to become a real life application for Rover.

contribution to knowledge:

- Knowledge acquisition of the various aspects of the PSC ie. Purchasing, Manufacturing, Services, etc.
- Representation of the knowledge in the database in such a way to represent the business domain itself.

10.1 SOFTWARE ISSUES

10.1.1 APN catalogue

10.1.1.1 assembly structures

The Statice APN catalogue needs to be updated in order to organise the assembly links information of the part descriptions in a well structured way which can be later used efficiently by the system.

A new logical entity **model** could be defined which would discriminate physical parts in assembly for different models. That reflects Rover's policy to generate different part numbers for physical parts which fit to different vehicle models. In sequence, the implementation of the Statice data base could

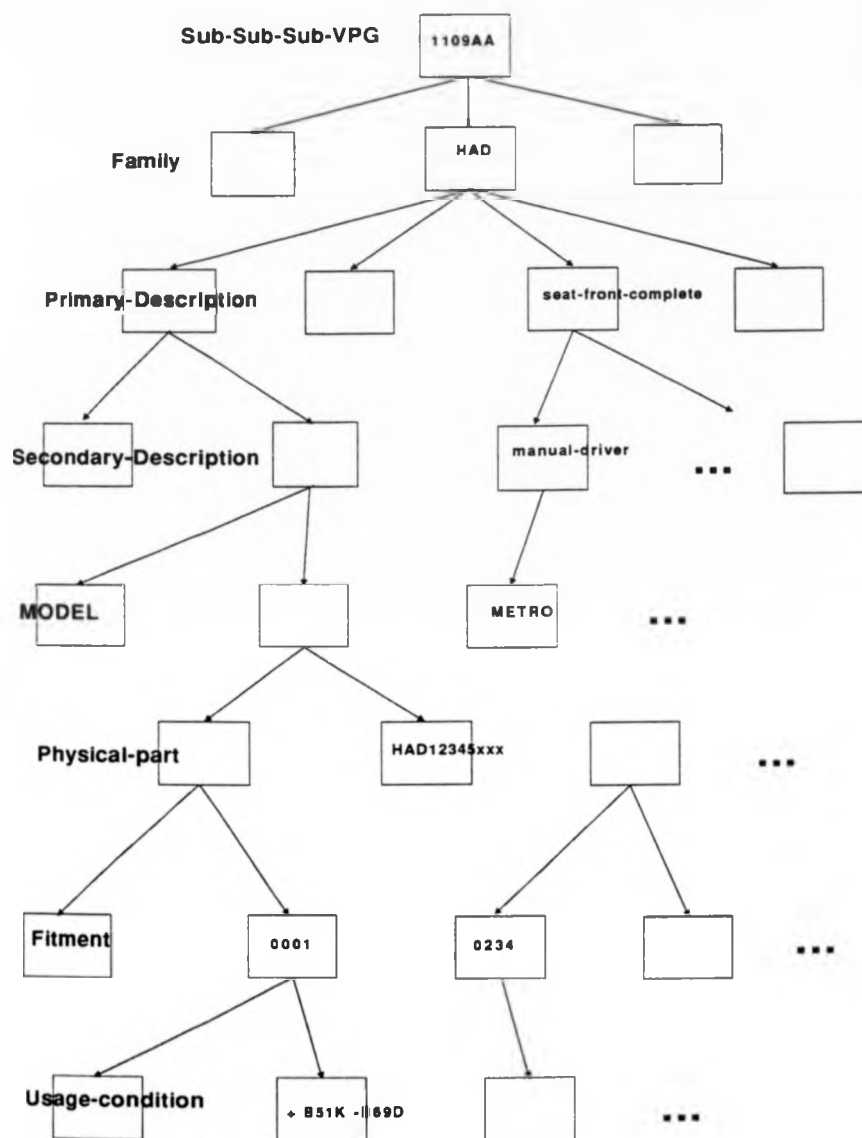


Figure 82: Further implementation of the data representation in STATICE

follow the hierarchical design of the EJA18V tape. That is,

Physical part, Fitment, Usage-conditions, as shown in figure 82. Notice, such an implementation of the Statice APN catalogue would increase the performance of the Intelligent networks prototype discussed earlier, with the intention for it to be supported by Statice, and statistical analysis in the future. That is, because of the extended semantics of the data base. The decision, however, whether to detail the features of the hierarchy structures or not is a trade-off. Extended semantics makes querying faster, but impose a penalty on space storage and calculation.

10.1.1.2 physical entity types

At the moment the must and maybe feature groups are implemented in Statice in the form of vectors of integers. The integers are decoded by the system to the actual feature descriptions. That is because Statice does not support arrays as built-in attribute types. The knowledge engineer, however, has the facility in Statice to define his own attribute types. It would be advisable in the further implementation of the system to define a physical attribute type of adjustable array. Physical types, when defined, do their own transactions of a data value from "raw bits" into a Lisp object within Statice.

The adjustable array attribute type could replace the sequence of vector attributes used in the ~~the-reference-of-Rover-cars----~~ actually-Austin-Rover entity and dynamically adjust itself when

new models are entered to the company. (Note adjustable arrays are supported by GENERA).

10.1.1.3 images

The author has implemented on a demonstration basis the storage of bit files or scanned images from Apple Macintosh and Symbolics in a Statice data base. Those images are not only a publishing document within Statice but an array value of an attribute of the object (actually a raster array). In that respect they can be manipulated as any other attribute value in Statice and Object Oriented Programming applied to them. For example, the graphical output of this value may represent the *presentation type* of the object itself which naturally follows. In addition, as the object under consideration is related within Statice to other parts by assembly relationships (data from EJA18V tape), a procedure (method) could drive the graphing of the overall assembly structure directly from the data base (provided the parts were scaled accordingly). This would facilitate the electronic storage of all the "prototype" top assemblies of the *intelligent networks* methodology.

10.1.2 *EJA18V data parsing (minor)*

The data of the EJA18V tape are in three or four separate files

Combination of B51 with B44 feature groups:

COMMON TRIM LEVELS

B51F + B44R	Y, Z
B51K + B44B	M, S
B51L + B44R	Y, Z
B51R + B44R	W, X

Combination of B51 with B69 feature groups:

COMMON TRIM LEVELS

B51F + B69D	Y, X
B51L + B69D	Y
B51R + B69D	X

Result:

Combination of all feature groups:

COMMON TRIM LEVELS

B51F + B44R ⁺	B69D	Y, Z
B51K + B44B		M, S
B51L + B44R ⁺	B69D	Y, Z
B51R + B44R ⁺	B69D	W, X

Figure 83: The result of the validation algorithm with the B51 the Master feature group.

ie. one file which keeps the usage conditions of the parts, another file maintains the assembly links of the parts etc. The system needs to link them together before it can proceed to the statistical analysis of the tape. The processing time for this function is one hour. It could be reduced by using a kind of search technique ie. bubble sort etc.

10.1.3 Validation Algorithm

At the moment the validation algorithm run the combinatorial exercise passed to it by the planning process of the system in the default order of feature groups. That is, it chooses the master (or pivot) feature group for a 'scoring' function based on the two principles outlined in Chapter 8.2.1.1.1.

Combination of B44 with B51 feature groups:

COMMON TRIM LEVELS

+ B44B + B51K	M, S
+ B44R + B51F	Y, Z
+ B44R + B51L	Y, Z
+ B44R + B51R	W, X

Combination of B44 with B69 feature groups:

COMMON TRIM LEVELS

+ B44R + B69D	Y, Z
---------------	------

Result:

Combination of all feature groups:

COMMON TRIM LEVELS

+B44R + B51R+	B69D	X
+ B44R + B51F+	B69D	Y
+ B44R + B51L+	B69D	Y
+ B44B + B51K		M, S

Figure 84: The result of the validation algorithm with B44 the Master feature group

As discussed in section 8.2.1.1.1 the system always starts its combinatorial exercise with some of the most feature groups or a

feature group which has at least two feature options in the AFCs of the model. The sequence of the combination of the feature groups does not affect the end result of the validation algorithm provided the algorithm does not start the process with a feature group which has only one feature availability (and other feature groups exist in the specification exercise). In that case, it may miss information. This is discussed in the following where in the "seat-front complete manual" example, the validation algorithm has being applied from three different starting points. Notice, for simplicity only the trim levels have been taken in account and no any other Base feature restrictions or territory and combinations restrictions.

The results are shown below firstly with the "B51: seat face material" feature group as the **master** feature group, secondly the "B44: seat-reclining" feature group and finally with the "B69: seat heating" feature group.

As it can be seen the first two approaches (figures 83 84) produce the same results , no matter the **master** feature group in question. In the third approach, however, information is missing. That is of the feature groups which do not *combine* with the master feature group in concern, in this case the "B69: seat heating" feature group. The algorithm shows this peculiarity because the "B69: seat heating" feature group does not cover all the trim levels of the Rover R8 model and there is the case where the rest of the feature groups do match in those missing trim levels. The problem can be solved with the update of the algorithm process as it is shown on the following diagramatically

Combination of B69 with B51 feature groups:

COMMON TRIM LEVELS

+	B69D + B51F	Y
+	B69D + B51L	Y
+	B69D + B51R	X

Combination of B69 with B51 feature groups:

COMMON TRIM LEVELS

+	B69D + B44R	Y, X
---	-------------	------

Result:

Combination of B69 with B51 feature groups:

COMMON TRIM LEVELS

+	B69D + B51F + B44R	Y
+	B69D + B51L + B44R	Y
+	B69D + B51R + B44R	X

+	B44B + B51K	M, S
---	-------------	------

This was not picked up from the algorithm

However, it can be picked up by creating a new pseudo feature 'B69T' and run the validation algorithm again as shown below.

Combination of B69 with B51 feature groups:

COMMON TRIM LEVELS

+	B69D + B51F	Y
+	B69D + B51L	Y
+	B69D + B51R	X

+	B69T + B51K	M, S
+	B69T + B51R	W
+	B69T + B51L	Z

Combination of B69 with B51 feature groups:

COMMON TRIM LEVELS

+	B69D + B44R	Y, X
---	-------------	------

+	B69T + B44B	M, S
+	B69T + B44R	Z

Figure 85: The process of the validation algorithm (and its Implementation) with the B69 the Master feature group.

(figure 85).

A new IF-THEN statement can be added to the system's planning process which interrogates the background of the master feature group. If the feature group happens to only have one feature option and its availability does not cover all the trim levels of the model a new pseudo-feature (ie. -B69D) is automatically created by the system and specified with the missing trim levels. Then the system continues its combinatorial exercise in the normal way and results in a *first guess* of valid combinations. In sequence, a *string match checking* against the record of the pseudo feature group is applied. This function collects all references for the possible legal combinations of the rest of the feature groups that do not appear in the *first guess*. Then the algorithm runs once more with these references as arguments. The result is the *missing information*, which updates the *first guess*. Notice, the new result is compatible with the rest of the system for further manipulation (graphing the output, statistical analysis etc.) as it has been generated by the validation algorithm itself. In addition, the practical implementation of this approach is reasonably easy by the means that functions which already exist in various applications within the system such as the automatic creation of a pseudo feature group (statistical analysis), or the background investigation of the feature group (definition of the PLUS and/or MINUS signs) etc. can be used.

In summary, the default mechanism of the validation algorithm is

the one acceptable by Rover. The investigation of cases in which a trivial feature group can drive the combinatorial exercise is practically obsolete. It has been discussed only for completeness of the software aspects of the system.

In the following section the further implementations which are required to be incorporated in the functionality of ROOVESP in order for the system to realistically tackle the PSC and interface with the existing work in the company are discussed.

10.2 System Design issues

10.2.1 Illustrated Parts List (IPL).

The validation algorithm of ROOVESP can interface with the IPL project in the sense that it could update its user interface by querying the engineer with IPL windows or matrices such as are proposed from IPL (see the third column of figure 47). The arguments which will represent the feature groups under consideration will be used as variables for the algorithm which will carry out its normal combinatorial search. Such an interface is possible because of the compatibility of the software and hardware platforms of Symbolics (algorithm) and Apple Macintosh (IPL) currently existing in parallel on the MacIvory machines. Those machines are standard Apple Macintosh machines with an additional Ivory board for the Genera software (Symbolics Lisp).

The Macintosh software is used as the front end of the applications (windows, scanned images etc) whereas Lisp runs the actual program. Apparently, the design of these machines matches with the requirements of the algorithm and IPL interface.

10.2.2 *Implementing the concept of time*

The variable of *time*, which has not been mentioned till now for simplicity, can be entered in the system, as well. Remember the main documents which drive the Product Specification Concept within Rover, the Product Development Letter, the Features List and the Base and Additional Features Chart reflect marketing intention at a point in time. For this reason the feature groups which are compiled by the Specification Services exist only for a specific period of time. That is the Design Effect Point (DEP) or *effectivity* (mentioned in section 5.1.1). It is vital for the auditing department to check that the "DEP. NO" of each specification document (figures 31, 32, 33) falls within the timescale of the life of a feature group or a feature. Consider the manufacturing timescale of the Rover R17 model for both 5 and 4 door salon and the coupe version of the vehicle is shown in figure 86 (*The dates are hypothetical*).

The problem arises when an engineer by mistake specifies, for example, the front seat of the Rover R17 model with the "B69: seat heating" feature group at the build phase D01 (October 90) for the 5 door derivative whereas this feature group has been

A part of the Rover's R17 build phases

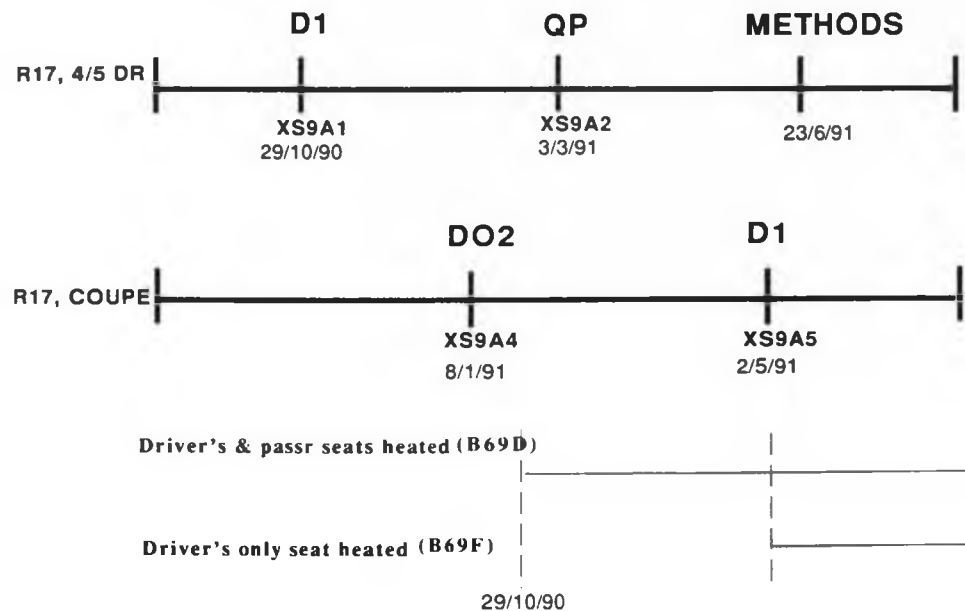


Figure 86: How time affects the specification of a vehicle

forecast by Specification Services to enter into the company at the DO2 build phase (January 91).

In another case if the usage statement of a part of the Coupe derivative is assigned to a date (XS9A4: January '91) later than the correct one (XS9A1: October '90), then this part will not be picked up by the Usage-Conditions-to-Parts system and will not be included in the PP&BLs (Preliminary Parts & Build Lists). Consequently, it will not be ordered for purchase from outside or

built in house. The lost of time in production can be imagined when in reality it concerns thousands of such parts.

The implementation of the time variable in the system is quite easy as only the higher level of the algorithm needs to be updated: ie. its planning process. According to the time requirement from the engineer to release his part (plus the part description and model type) the system could choose only the must and maybe feature groups (or features) with the correct effectivity.

At the moment, the system checks only the effectivity of the features by means of current existence. Implementation of effectivity to a more detailed level is straightforward provided Rover produces the documents which translate effectivity codes to dates. Note, the effectivity codes do not directly refer to a date but to major manufacturing events within the company.

The concept of time could be implemented within ROOVESP with other more effective ways if this were required. Such an implementation is feasible because ROOVESP works with Statice and OODBs are far more flexible than traditional databases for encapsulating the concept of time within their data c.f. the cache facility of Statice. More details on this subject can be found in the work of Thomas Atwood [75].

10.2.3 PROSPECTS and PIMS (PROMS)

The implementation of the **ROOVESP** coincides with a project currently under development within Rover called PROMS which is to integrate Land Rover's specification system (PROSPECTS database) with that of the previously known Austin Rover (PIMS database). A detail discussion for the PROMS system is beyond the scope of this chapter, however, the general idea of PROMS is to **replace** the existent PIMS and PROSPECTS data bases of the previously known Austin Rover and Land Rover companies respectively, with a single relational IBM mainframe data base (probably DB2). In that way both companies would operate with a single specification system and common data.

The replacement of PIMS by a new system gives rise to the possibility that all the information currently stored in PIMS will be lost. That is, not the actual data kept in PIMS but the **thought processes** which led to their creation.

Notice, there are various databases within the previously known Austin Rover, apart from PIMS, which operate as autonomous entities and service various other departments. For example, the COPICS database in Manufacturing and Purchasing department's data base. Up till now the Product Specification Concept within Rover has operated manually. Though the data are stored electronically, the actual **thought** mechanisms of product definition are implemented by the people. For 8 years people from the various departments of the company such as Specification Services, Auditing, Component Engineering, Manufacturing, Purchasing, Vehicle Directorate, have implicitly stored their knowledge and

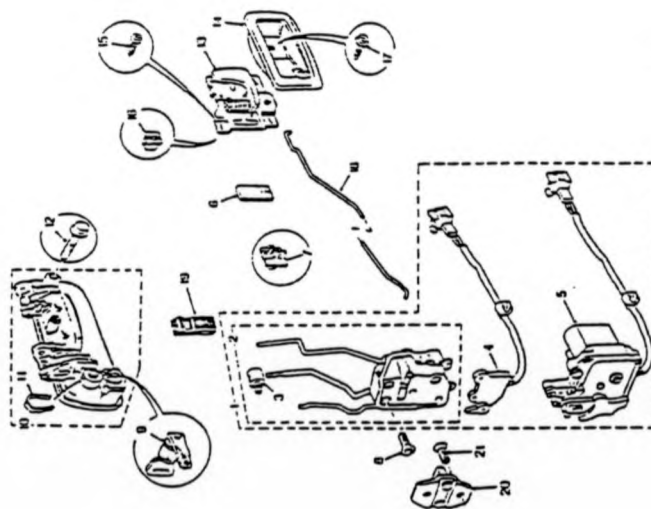
AP/IPC BASED PROTOTYPE PARTS LIST

GROUP 1

CHASSIS, SUBFRAMES, BODY SHELL, FITTINGS

FRONT DOOR HANDLE INTERIOR

APN NUMBER	PRIMARY DESCRIPTION	U.I.	SECONDARY DISCIP	UOM/DA NO.
13 1VC	Handle door release assembly		Inner panel	72120
14 1VJ	Finisher escutcheon		Door rel Assy	
15 0YP	Screw lapping		Semi std part	
16 110 AVH	Grommet screw NS			
17 0YP	Screw lapping		Semi std part	
18 110H	Link handle		Front door	
19 1JZ	Button stiff lock		Front/rear door	
20 100	Striker		Door lock	
21 0YP	Screw lapping		Semi std part	



ML 0811

ML 0811

Figure 87: A page from the Service Parts identification document (UNIPART) of Rover.

experience in databases.

The application of ROOVESP to the Auditing and Specification Services departments proved that such knowledge could be retrieved and analysed to well specified production rules which could form a kind of past experience (experience-link). A team of people within Rover, enthusiastic about the experience-link results, plan to apply the philosophy of the Meta-knowledge mechanism of ROOVESP (ie. combination of statistical analysis and specification patterns) in other databases, as well, and procedures are to be written in order to retrieve more information regarding the **part descriptions** within the company. Such a policy would serve two objectives:

Firstly, knowledge which may be lost with the implementation of PROMS will not only be saved but also refined and

Secondly, such a knowledge could support PROMS in its actual application within to Rover and ease the specification of the product. The rest of this section outlines thoughts by the author which ROOVESP could implement in the future.

The parsing of the APN catalogue and its consequent storage in Statice, for example, can reveal additional information for the source of the parts if appropriate statistical procedures apply (methods). It is well known, for example, that the windscreen of the vehicles is always bought from outside. If such a condition could be proved statistically then the system could automatically apply the source information required by a specification document (eg. CDC see figure 24).

In addition, there is much information coming from the APN

BB SIMS: Colour and Trim Chart Ref POINT Release No 6075 Issn 5th JUNE 1990 Printed on 07/01/94

A page from the Colour and Trim chart document of Rover R8 200/400.

catalogue data tape which is not yet analysed. For instance, statistical analysis on the service indicator attribute of the part-descriptions entities (ie. secondary descriptions-ID) in the APN catalogue could identify items which Rover services. Otherwise auditors have to work out both the the specification document and the UNIPART document of the model.

Figure 87 represents a page of the UNIPART document. The parts of the front door handle of the Rover R8 model which Rover services are listed. This document has more than a thousand pages for each model which shows the load of the work involved in a manual operation.

Statistical manipulation of the data from the Purchasing department could establish lead times for the purchasing of some parts which have been bought in the past and forecast the new time schedules. Similar analysis on manufacturing data could reveal forecasting lead times for the manufacture of all the parts in a specification package.

One more possibility is that the production volume could be automatically detailed into the colour level for each part description with statistical analysis on the past data in the Purchasing department. Remember that most of the parts have to appear in the assembly line with a colour (see table 4). For example, a front seat may have a colour. The colour indicator is added at the end of the part number with a 3 digits code (it is the three X's appeared at the end of the part number, xxx, see figures 23, 24). The colour information is added later in

FUTURE SPECIFICATION REQUIREMENTS 2

= FSU - D02 - PRODUCT ENGINEERING
& MANUFACTURING

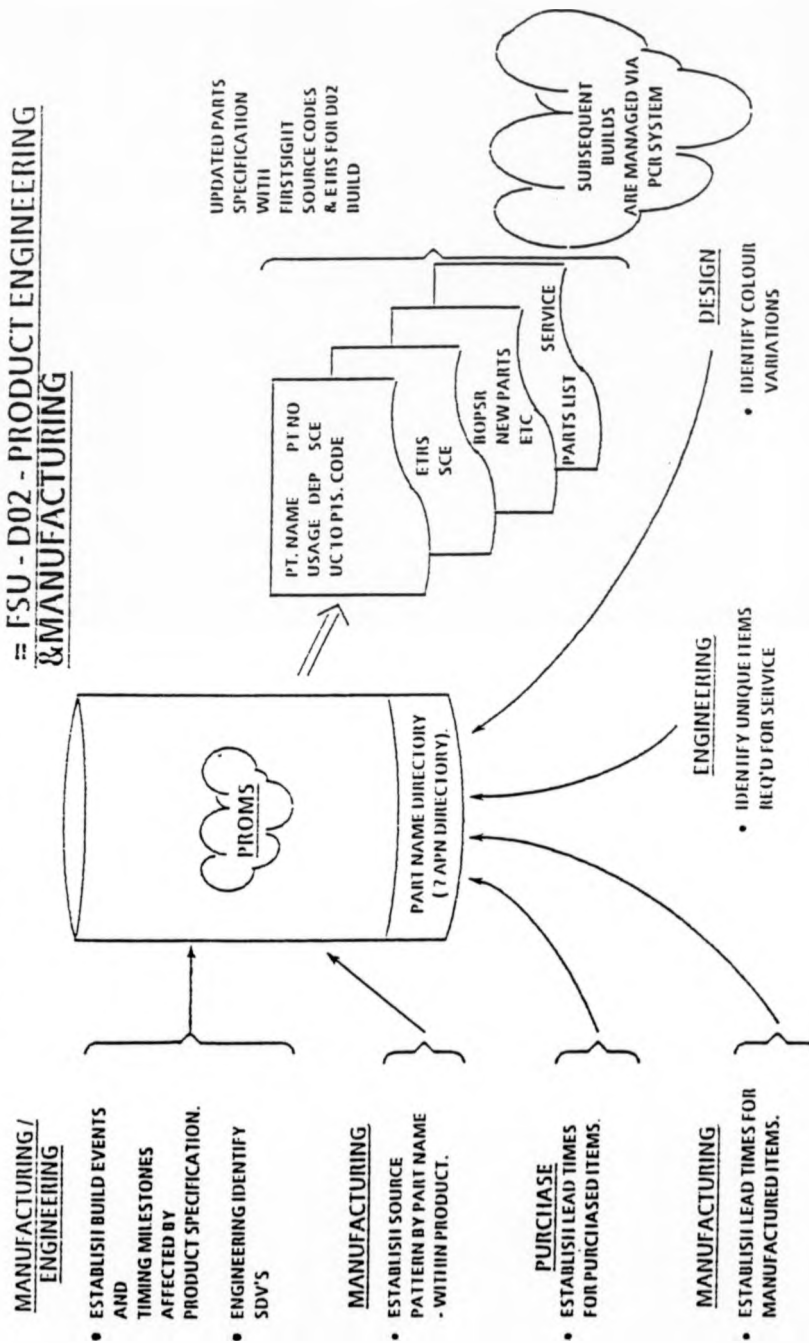


Figure 89: How ROOVESP believes that PROMS can be helped by the link between various part information and part description. From the presentation of Mr R. Mcleod to the PROMS team.

manufacturing by another team of auditors who have to validate against the Colour and Trim Master Chart Release Document (figure 88). The system could specifically establish the volume of parts required in production for all the different colours of the assemblies. In summary, the general idea is that all this information, retrieved from statistical analysis of past data could be stored in a database and support PROMS "up front". In this way the knowledge gained from the Meta-knowledge mechanism of **ROOVESP** could support PROMS in the specification of the product directly from its design phase to the final production line.

Figure 89 shows how statistical analysis at the part description level (as proposed by the author) can help PROMS by investigating all the existing sources of such information ie. colour, services, Purchasing, Manufacturing, etc.

10.2.4 SUMMARY ON THE CHAPTER

In this chapter was discussed the future developments of **ROOVESP** both from the software and design point of view required in order for the system to tackle the whole PSC in the company and also help existent systems under development for the same purpose.

IPL represents the first phase (or investigation) of interfacing images in the specification packages with which an engineer could

work. PROMS (PROSPECTS & PIMS) represents the second phase. IBOM (Illustrated Bill Of Materials) the third.

ROOVESP stands among all them in order to give ideas for the appropriate software tool required and designed and also has proved that intelligence can be put to the PSC in Rover, rather than just electronic facilities.

11. DISCUSSION ON ROOVESP

IPL has been previously mentioned and the fact that ROOVESP was guided to similar objectives with that of IPL. As IPL is already an operational system within Rover, it seems a good opportunity to compare the two systems in order to make clear the achievements of ROOVESP.

The first section of this comparison gives the opportunity to discuss in more detail the benefits of Object Oriented Programming and OODBs in relation to relational databases which were mentioned in chapter 7 but also revealed during the development of the system.

The second section of the comparison deals with the conceptual design of ROOVESP in relation to IPL.

contribution to knowledge:

- An in-depth investigation (and comparison with the traditional way of programming and data representation) of the merits of the Object Oriented Paradigm.

- Proof for the integration of the existing systems in a single master database which can store both images and texts and increase performance.

- Enhancement of the traditional audit way.

- Refinement of the design engineering knowledge and

experience in the company.

- ROOVESP can benefit from existing systems in the AI field.

11.1 comparison of IPL with ROOVESP

IPL cost to Rover much more money than ROOVESP. < D

11.1.1 right software choice for the characteristics of the problem

In this section the software aspects mentioned in the comparison of OOP and OODBs in chapter 7 will be discussed in more detail.

11.1.1.1 Data abstraction and semantic overload

11.1.1.1.1 1st APPROACH: NORMALISATION

Consider the relational IPL model in figure 49. This model has been 'normalised' (see [70]). For example, the table **Design Feature Group/ Vehicle Part Group** has been implemented as the *intersection entity* (see [66]) of both tables **Vehicle Part Group** and **Design Feature Group** which are related to a *many-to-many* relationship.

There may exist many more conceptual relationships in the model (or any other relational model in general) than only intersection entities which are shown in the diagram (figure 49). These relationships are known only by the designer as they are not explicitly stated anywhere. For example, there is an implicitly expressed *class-subclass* relation between the feature groups and features in the system. However, although this relationship is obvious, it cannot be explicitly stated in the relational model. Instead, it only exists in the mind of the designer (in an OOP environment such a relation could be hard-wired). Additionally, sometimes two or more tables may express the same set of objects but with different table-names because of the need of different concepts in the design of the system. A more detailed explanation of this subject can be found in the work of R. Hull and R. King [66].

Such relationships tend not to be explicitly stated anywhere, because there are only two ways to represent relationships in the relational model: fields (or attributes) within a relation or by using the *join* operator between the entity tables. For example, consider a simple query in the normalised IPL environment of accessing the *design features* of the product "seat-front complete manual". The ORACLE's SQL query would look like as in the following:

```
select Feature-Description
from DESIGN-FEATURE
where DESIGN-FEATURE-GROUP.feature-group-code in
```

```

select DESIGN-FEATURE/VEHICLE-PART-GROUP.feature-group-code
from DESIGN-FEATURE/VEHICLE-PART-GROUP
where DESIGN-FEATURE/VEHICLE-PART-GROUP.product-code in
      select PRODUCTS.product-code
      from PRODUCTS
where Products.product-name = "seat-front complete manual"

```

All the relations between the tables are represented with equality in the values of their equivalent fields. Consequently, because many different relationships in the data base are represented only in two different ways, the data base is semantically overloaded. In addition, the lack of documentation of those relationships by the model itself (by appropriate primitives), makes the representation of the knowledge in the system more obscure to the outside programmer.

In addition, the normalisation of the data base resulted in the creation of new logical tables (intersection entities). This adds to the work of software development of the system. That is, at any stage of the development of the system when information from this table is required, the programmer has to know the internal structure of these tables in order to retrieve information from them. For example, if the programmer needed to find the feature groups which are associated to the part code "HAD12345xxx" he should know the way in which information is stored in the **Design Feature Group/ Vehicle Part Group** table. The procedure in SQL would like:

```

select DESIGN-FEATURE/VEHICLE-PART-GROUP.feature-group-code
into .....
from DESIGN-FEATURE/VEHICLE-PART-GROUP
where                DESIGN-FEATURE/VEHICLE-PART-GROUP.part-code
                                = "HAD12345xxx"

```

From the software point of view this decreases data abstraction as the programmer must be constantly aware of the details of the data structures, every time he uses the system.

On the other hand, in the OOP, as mentioned in chapter 7, methods attached to specific classes of objects represent the communications protocol of the data base and the programmer need not be concerned with the way in which the objects are implemented in the data base.

In order for this to become clearer let's consider the **secondary-description-ID** defined in the Statice data base (fig 90).

The attachment of the feature groups in its internal structure resulted in the creation of a new logical entity, as well, named ~~the-reference-to-Rover-cars----~~actually-Austin-Rover, but the association of the **secondary-description-ID** object with this new logical entity is more abstract than the relational model from the programmer's point of view:

Two methods **get-musts** and **get-maybes** were defined to work with that logical entity and they were accessing the values of the **must** and **optional** feature groups of the part description it was linked to. The programmer did not need to know afterwards the

internal representation of the entity in order to access the feature groups, every time they were needed. Notice, the functions themselves do indeed involve a lot of detail such as the specification of the model, the translation of vectors to lists and then to feature groups, etc. All this detail is hidden and consequently the program development becomes more rapid.

The equivalent implementation in a C function, for instance, to do the same job in the relational model does not offer the same level of abstraction. Firstly, because this function would not be local to the **secondary-description-ID** entity but generic to the whole program. Secondly, it could not be inherited in the knowledge representation hierarchy of the data base. Consequently, it could not be organised structurally and classified.

The example of defining and accessing data from the relational model continues and reveals even more complexity.

The must and maybe feature groups in the the-reference-to-Pover-cars---actually-Austin-Rover entity are implemented with two matrix-like variables (in reality two sequences of vectors). The

(define-entity-type **secondary-description-ID**

()

(part-description string :unique t :inverse-index t)

(emission-indication string)

(services-indicator string)

...

```

(Has-negative-features      vector)
(Has-must-feature-groups    Rover-cars--actually-Austin-Rover)
(Has-maybe-feature-groups   Rover-cars--actually-Austin-Rover)
(picture      raster-array)
...)
```

```

(define-entity-type Rover-car--actually-Austin-Rover
  ()
  (Rover-R8-musts      vector-of-integers)
  (Rover-R8-maybes     vector-of-integers)
  ...
  (Mini-musts          vector-of-integers)
  (Mini-maybes         vector-of-integers)
  ...)
```

Figure 90: The secondary-description-ID entity implemented in Statice.

matrix format is required in order to map the design features with the vehicles specification. Such an implementation in the relational model can be done in two ways:

Firstly, by creating two new tables (must and maybe feature groups) and linking them using *aggregation* (intersection entity) with the part description (products). This has already been discussed earlier and it was shown to result in greater space requirements and lack of naturalness in the representation of the knowledge in the system.

Secondly, by using the relational model in a way that it could simulate *object-oriented* behaviour. For instance, the *non-first-normal-form* (NF^2) model [97] and as stated in [69] "the non-first-normal-form model allows nested relationships whereby hierarchical relationships among subobjects can be modelled".

11.1.1.1.2 2nd APPROACH: NON NORMALISATION

For example, using the NF^2 model, the part description entity, assuming that up to 4 (either must or maybe) feature groups are needed to specify its usage statement, could be defined as:

```
create Part Description
[product-name: string(40)
 product-code: string(11)
 has-feature-groups: {          /* ordered list of axes */
    [ musts:                    [DH_MATRIX:          /* 4x4
                                Denavit-Hartenberg
                                Matrix                */
                                < 4 FIX [COLUMN: integer,
                                VECTOR: <4 FIX real>] >
    ]
    [ maybes:
        [DH_MATRIX:            /* 4x4 Denavit-Hartenberg
                                Matrix                */

```

```

        < 4 FIX [COLUMN: integer,
            VECTOR: <4 FIX real>] >

    ] )

end.

```

This example has used the syntax of the AIM-P prototype which represents an implementation of the NF^2 relational model [98] (relations are denoted in AIM syntax by the pair of brackets $\{[...]\}$). The reader for more details can refer to [98] or [69]. The definition of the *Part Description* relation, more specifically, has been borrowed from the relation which defines an industrial robot in [98]. However, it can be used as a general example to demonstrate, that even the second approach would result in the lack of data abstraction in the database model.

The insertion of the must and optional feature groups into the nested NF^2 structure of *Part Description* would look like:

```

insert
    [ COLUMN: 2,
      VECTOR: <0, 1, 0, 0>]
into Part-Description.has-feature-groups.musts.DH_MATRIX[2]
from .....

```

The example demonstrates that the nesting of the relations in the *Part-Description* structure is reflected from the nesting of the operations which commit a transaction. As A Kemper, P. Lockemann

and M. Wallrath state in [69] "this is by no means trivial and requires an *intimate* knowledge of the representation of the entities in the data base". The example also, shows that though this approach gives a better flavor in the 'meaning' of the representation of the data in the data base, it decreases drastically the data abstraction in the model.

11.1.1.1.3 3rd APPROACH: IMPLEMENTING OBJECT-ORIENTED FEATURES IN CONVENTIONAL DATABASES

For the reasons mentioned above, there are currently ongoing prototypes - or even prototype design database methodologies, see [37] which try to implement Object Oriented software facilities on the top of existent RDBMS. Those facilities would specify new datatypes in a more abstract way and they would be isolated from the internal structure of the data in the data base. Such an example is the ADT-INGRES [69], or POSTGRES [65], [78] implemented on the top of the existing DBMS INGRESS, or R²B² (Relational Robotices Database System) [69].

The problems which arise in this approach are:

(i) since all abstract datatypes have to be mapped on a flat conventional database domain the abstract structure of a data objects is only superficial and the actual benefits of OOP get lost.

(ii) the designer has to become familiar with two quite

different systems. In the ADT-INGRES for example: (i) the database language QUEL and (ii) the programming language C.

(iii) it is quite difficult to implement *transitivity* on the new defined abstract types. For example, it is hard to define a new matrix datatype on the basis of the before defined vectors datatypes.

On the other hand, Statice provides the facilities to the designer to specify his own *logical* or *physical* datatypes. In the case of the *physical* datatypes, actually, the designer can program his own way how he wants the specific datatype to store or retrieve its information from the data base. For instance, he can define a matrix of strings rather than only matrices of numbers, or a matrix of both, or still an adjustable matrix rather one with specified dimensions. The development of such a code is easier and faster than using conventional programming, as the designer is only concerned with a single software environment, GENERA, rather than two or more plus their required interfaces.

In addition, Statice provides the facility for building new database structures on top of those which have been previously defined by the programmer (*transitivity*) eg. the definition of matrices can be built on the definition of vectors. This extends even further the concept of data abstraction during the development of the database and debugging.

Finally, it must be noticed, the need of high variety of abstract datatypes in the engineering environment. This is because of the

need of specialised data in this area such as CAD/CAM (solids or two dimensional objects with physical characteristics) or CIE data (images), rather than the stereotypical data required in the commercial world, strings, numerics, dates etc. In the work of R. Martin [58] it is illustrated the inability of the relational model to directly represent one of the simplest no-commercial datatypes, the array, though it represents a widely used and well-understood data structure.

11.1.1.2 Uniformity and extensibility with OOP

The environment for the extensibility of ROOVESP to a full system for Rover is the same, the GENERA environment, whereas this is not the case with the relational model. IPL had to use a new language C in order to interface the needs (graphics and search) of the system with ORACLE. In this respect OOP is more *uniform*.

The primary benefit of OOP is improved productivity because of the higher level of modularity it embeds which eases maintenance and reuse of code. The internal inferential integrity of the FLAVORS or Static environments guarantees higher consistency in the updated code whereas the special debugging mechanism for object oriented databases such as the FLAVOR EXAMINER [4], existing in Lisp machines, can result to even more modular extensibility of the code.

11.1.1.3 efficiency during the system development

It was planned for IPL and ROOVESP to interface with each other sometime on February 1990. However, this did not happen as unanticipated complexities due to problems arose in the program of IPL. For example, in one case, it took 2 months and more than 20 pages of code to program the deletion or insertion of a new part in the data base. That was because of all factors which needed to be taken into account: the parts which it is linked to, its image reference, etc. The programmer had to work with at least three different languages, SQL, ORACLE, 4th DIMENSION, plus their interfaces.

In ROOVESP, on the other hand, the deletion of a part for example, was direct and natural due to the way in which Statice works and its embedded inferential integrity mechanism. In addition, only one software environment, GENERA, was required. Notice, of great importance to the fast development of ROOVESP was the contribution made by the incremental compiler which Lisp Machines support. In this way, only the changed part of the program needed to be compiled each time rather than all the code. More details on this subject can be found in [4].

11.1.1.4 Database performance

11.1.1.4.1 creation and run time performance

The major problem with IPL is run-time performance. This is because when a VPG is considered by the engineer, the relationships of all of its parts are created 'on the fly' from the 'flat' structure of the relational database. Secondly, the reference to the images database needed in order to retrieve the appropriate scanned files to the text data slows down the whole process. In addition, the multiple interface protocols necessary because of the lack of a common hardware platform (PIMS, ORACLE, 4th DIMENSION) loads the network processes and slows down the program's performance even more.

On the other hand, it was found that the creation of the actual Statice database takes a long time to complete. This is because of all the complex relationships of the data objects that need to be created during its initialisation. However, once the OODB has been created its performance during the actual operation of the system which is the generation of the usage conditions of the parts, is higher than that of the relational model. The superiority in the run time performance of OODB is because the objects in the database are linked by identity (the objects themselves), not state (key values) as in the relational model (More details on the subject can be found in the work of R. Hull and R. King in [66]). In the latter, at least one address translation is required to get from the key (attribute) value to the location of the tuple.

In OODB however, the objects in the program memory have attributes whose values point direct to other objects. (ie. *has-secondary-descriptions*, *Belongs-to-the-VPG*, ... see the

definition of **Secondary-Description-ID**. See also, photos A14 and A15 in appendix 10).

11.1.1.4.2 indexing and clustering routines

Relational databases are considered to have the best indexing routines. Statice and other OODBs also support such routines. This can be shown in the following.

Looking at the **primary-description-ID** entity there is an index which is used to speed up a regular accessor function. This index increases the speed of locating the *has-secondary-descriptions* attributes in the data base. In reality, the performance improvement happens in both the reader and the writer functions of this attribute as such functions are used by the system when it builds up the APN catalogue database automatically. Working with Statice it can be assumed that its indexing routines can be quite efficient, especially when they are used in relation with some of the unique features of the OODBs. In order to understand this better let consider the **secondary-description-ID** entity in Statice.

The **part-description** attribute of the **secondary-description-ID** entity has as inverse function the *the-part-description-entity-type*. This function is created automatically by the Statice system when the **:inverse** option is tagged to the **part-description** attribute. When this function is applied, it accesses the object (entity) whose value of its attribute matches the query, from the

database eg. a query such as "seat-front-complete manual-driver" accesses the Statice entity with such a part-description value. Notice, that each part description attribute is unique in the data base, consequently only one Statice object will be returned after the evaluation of the function. In general every Statice reader function, including the inverse ones, can be characterised as *one-to-one, many-to-one, one-to-many, many-to-many*.

The inverse reader function is one of the additional facilities of Statice that no traditional database systems offer and when is used with indexing (:inverse-index) can result in high performance in the data base. This is because the actual objects in the data base are sorted with the index routines and the values of the various attributes can be directly retrieved. To clarify this consider the inverse reader *the-part-description-entity-type* discussed earlier.

The way in which this function, it finds the entity for the part description named "seat-front-complete manual-driver" is by examining each entity of type **secondary-description-ID** (including entities whose type inherits from type **secondary-description-ID**) and checking the value of the part-description attribute to see whether it equals "seat-front-complete manual-driver". This inverse function although it is a unique facility offered only from OODBs, the performance of the database would be slower and slower as it grew, taking time linearly proportional to the number of **secondary-description-ID** entities (a significant problem if considered the almost 15,000 part descriptions within Rover). In fact this is what happens if there is not an index.

The index can increase the performance of the database rapidly without the inverse function facility be lost.

In [58] R. Martin supports that the clustering techniques of the OODBs can be superior of those in the relational model. In the automatic creation of the APN catalogue database, because of the hierarchical connections at the VPG group level of the data after analysed from the VAX tape, it shown that clustering is a efficient characteristic of Statice and could increase performance drastically (see section 8.2.4.2.3).

In summary, the direct (by 'identity') relation of the data in an OODBs has been compared with the relational model where this happens with equality in the values of the attributes of the data in 'flat' files (by 'key'). In addition, it was shown that Statice supports the standard indexing routines which can be quite efficient when they are used with inverse readers functions which can directly access the objects from the data base.

Statice is slow during the creation of the database but it is superior than the relational model at the run time. The long time required for the creation of the data base can be overcome by overnight processing and needs to be performed once. However, the performance of a data base during the execution of the program is crucial - as it affects the company if it slows down - because it is needed daily.

11.1.1.5 representing assembly structures

Here some conclusions which were drawn by developing the Intelligent Networks prototype using Object Oriented Programming and Statics are discussed.

The way that arcs (assembly links) and nodes (physical parts) are linked in the internal memory of the computer ie. **direct** referencing to the object which are linked to, it was shown in chapter 9. Considering the amount of data involved - 180,000 physical parts PLUS at least as many arcs - the Object Oriented Database representation offers the highest performance over any other means of storing data.

Performance proved to be higher than that of the relational IPL model. This is because of the direct relationship of the user interface with the database itself within the Object Oriented environment. For example, when the engineer at the command level requires the deletion of a node (part), the application program expects the indication of a node (object) by the mouse. When the engineer "clicks" on the node it actually "clicks" on the **object** of the data base itself. The node in the screen is only the "illusion" of the actual information kept in the object.

The facility of OOP of **presenting** objects in specific ways (presentation types) depending on the programmer's need proved to help development time rapidly. For example,

(i) the way to present the arc object as

<#arc:....from node1 to --> node2>

increases naturalness which results in faster conceptualisation of the results of the developing code.

(ii) as the programmer can pick up the bulk of information he needs by just clicking the mouse in one of the objects, and pass it as an argument to a function, development time decreases and debugging becomes much simpler.

11.1.1.6 Storing images

Statice offers the facility to store images permanently as real values (raster arrays) of an entity type attribute in the data base. In this way the relationship between the textual and non-textual information is direct as it belongs to the same object. As another benefit it increases naturalness in the system.

In the IPL example, however, it has been shown that the system complexity and the fact that the images exist in a different database contribute to the low performance of the system. This was because of the need to link together the textual and nontextual information by using two more software tools besides Oracle, the Pacer and 4th Dimension and interface of different hardware platforms.

11.1.1.7 User interface facilities

ROOVESR is the result of the third approach to the problem (ie. the Audit function and the PSC), totally different from the previous ones. The first approached the implementation of the

Features List and the second the use of multiple dynamic windows (Base and Additional features, explanation facilities, backtracking etc) updated at run time accordingly to the user choices. The involvement of many different departments in the problem (Engineering, Auditing, Specification Services, Business System, Manufacturing, Purchasing etc) showed that early prototype was to be the only solution for its implementation in order to co-ordinate their different perspectives based on the results of the program. Early prototyping also, helped greatly in the development of the second and third phases of the Audit function resulted to new way of thinking.

The use of the GENERA's presentation types to graph the truth maintenance mechanisms embedded in the system helped the engineers/auditors and the knowledge designer to communicate among themselves as both the internal structure of the data base and the inference procedures of the program could be seen on the screen. In the former, in particular, the embedded inheritance mechanism of the Object Oriented Paradigm helped greatly. In this respect, Object Oriented Programming aided the "planning in a hierarchy of abstractions" technique. The different subsystems of ROOVESP (ie. the validation algorithm and the Intelligent Networks prototype) could be viewed from different angles, as the user interfaces were relatively easy to implement and different thoughts could be demonstrated to the people in Rover. This also proved that when Object Oriented programming is used together with the "Planning in a hierarchy of abstractions" technique, the development environment is capable of exhaustively searching

many possible alternatives for the solution of the problem being programmed.

On the other hand, in the relational model, the implementation of user interfaces would require the need for additional software tools such as the C programming language or GKS for the VAX system, or both.

11.1.1.8 general issues in the selection of OODB

The choice of Statice to store the vehicle data was chosen on the basis of some general characteristics of the problem, as well. For example, the hierarchical design of the data by using an OODB was closer to the reality of the way in which the data were stored in Rover at the time. ie. the APN catalogue (or EJA18V) although has a flat structure in the VAX tape, it has been processed to describe hierarchies of VPGs. OODBs are efficient at maintaining hierarchical data structures (classes, inheritance, etc) while relational databases represent a totally different philosophy. Similarly, the structure of the AFCs in a simple hierarchy of feature groups and features results promotes naturalness in the system:

(i) the database represents the format of the hardcopy file which the auditors use, and

(ii) the designer of the system (in this case, the validation algorithm) can 'think' in a similar manner with that of the auditor or engineer ie. feature groups which are related to the part description and their feature restrictions which create the usage condition of the part.

In summary, up till now the software issues of **ROOVESP** in comparison with **IPL** have been discussed. In the following the reasons for which the design of **ROOVESP** makes it a better environment to tackle the Audit function and the PSC in the company than that of **IPL** is discussed.

11.1.2 conceptual analysis of **ROOVESP**

11.1.2.1 automation

IPL, with the way it currently works is limited to the existence of manual processing in some stage of its function: the engineer can make changes using the front end facilities of 4th Dimension such as the deletion of a part, update of the part's usage condition etc. but all these changes do not directly update the PIMS database. Instead, they are hardcopied and passed to the Specification Services/Auditing department people which manually update PIMS. Only the existing data tables which are already filled in ORACLE are updated automatically (see section 7.5.1).

On the other hand, **ROOVESP** with the use of the Intelligent Networks prototype can directly update Statice which holds all the parts and consequently the PSC can be fully automated.

11.1.2.2 multiplication of the information

As previously mentioned the duplication of information in the AS (Applications System) and PIMS databases, already exists from the way the PSC in Rover operates. IPL generates a third multiplication of the information: the ORACLE database.

All three data bases must be 'on line' all the time without mentioned the update of the images database in an 'on line' basis, as well.

On the other hand, in ROOVESP there exist only a single database, Statice. The images, as well, are directly connected to the parts themselves which makes even easier the software development of the automatic 'on line' update (see photos A8 and A9 in appendix 10).

In IPL it is difficult if not impossible to write software which will simultaneously update text and image data because of the separate update of the images from the rest of the data (the engineer has to go out of the 4th Dimension software, in a scan or a drawing package) and also is manual.

11.1.2.3 intelligence

IPL applies only a basic 'checking' on the data. This is only a

checking in the type of data expected in each field amended for the Rover's case; four characters with the fifth to be a number, etc. In reality, IPL plays the role of a "library" where the data are classified by VPG, Sub-VPG codes, etc. instead of doing any kind of validity test. For example, the engineer can delete a part (a part number), or change its usage statement without this be checked at all.

In one case only, IPL tried to apply a kind of 'intelligence' in its procedure. This was the association of feature groups with specific VPGs. This knowledge was **manually** obtained from a seat engineer who was appointed to work in a prototype project for the seats of the car. The gradual acquisition of the knowledge proved that experience is vital and the complexity of the specification can be extremely hard to maintain if is dependent only on the expertises conceptualisation of the problem. Even on that high level of analysis, the VPG, the association of VPGs with design feature groups proved to be a complicated exercise for IPL.

On the other hand, however, **ROOVESP** with the OOP, was able to **automatically** analyse the data in the part description level (and any level higher) by actually manipulating the lowest level of information available in the company, the physical parts.

In addition, there was still the inclination for mistakes in IPL as the engineer can choose for a wide area of feature groups which correspond to the equivalent VPG. The subtrees of the VPG family, however, even the nodes themselves sometimes do not share common design engineering "behaviour" (see figure 65).

Consequently, the engineer can still make mistakes and choose a wrong feature group to specify his part.

In the following section the relation of **ROOVESP** to the other systems which were described in chapter 7 is discussed and some ideas of how methodology of other existing systems could contribute to the further implementation of **ROOVESP**.

11.2 **ROOVESP** in relation to other existing systems in AI.

11.2.1 Hierarchical planning

It was mentioned in chapter 7 that unclear procedures and complexity were met during the investigation of the Audit function and the PSC. In this perspective, it seemed vital that a planning technique be used which would allow the design of each phase to be tackled independently but also allow any adjustments that may be needed by the whole system to be easily implemented.

Planning with abstraction provides such characteristics and the whole design can be updated if needed by simply modifying the interactions protocol of the system's subcomponents. This is obtained by using *satisfaction posting*. The same technique, abstract planning of *almost hierarchical decompositions*, has been used extensively within the first audit phase, as well (see

figures 51, 67, 69).

Although the philosophy of using constraints for algebraic planning, the one which **CONSTRAINTS** adopts, has been used in the design of the validation of the usage conditions in the first audit phase, **MOLGEN**'s philosophy of viewing constraints as the elements which define the subproblems interactions has been adopted in the design of the overall system.

R1 probably represents the best example in literature that resembles the application of this thesis ie. it configures a system (VAX computers) of components with relationships among themselves. The difference is that it does not use combinatorial search but the Match method instead.

The Match method, when it is considered as a searching method, it represents a weak (general) search technique (see [89]). Consider, for example, the below piece of code which represents a match function with a wildcard qualifier such as the star (*) (see [99]).

```
(defun Match (pattern data)
  (cond ((and (null pattern) (null data)) t)
        ((or (null pattern) (null data)) nil)
        ...
        ((equal ((car pattern) '*))
         (cond ((Match (cdr pattern) data))
               ((Match (cdr pattern) (cdr data)))
               ((Match pattern (cdr data))))))
```

The interesting case here is the wildcard qualifier star (*) which when is going to match against a form of data it has to try out recursively all the possible combinations. When many wildcard qualifiers are involved in the searching of the algorithm this requires a huge amount of processing which in another case may lead to combinatorial explosion. As McDermott, the author of R1 states, "the search space for Match is the space of all instantiations of the variables in a form, ... and Match is not always sufficient for the complete task."

The combinatorial search (as the validation algorithm discussed in this research) has been preferred from many researchers in AI (see [57] and [90]) as heuristics can be introduced in order to guide the path of the search by the means that more intelligent choices are allowed to be taken by the program rather than only pure chance. Furthermore, heuristics could be used to optimise the solution.

Regarding R1, the simplicity of its domain configuration (VAX) relatively to the vehicle configuration discussed in this thesis, the rules of composition are only few (480) which have been derived manually from the experts. In this project however, as it will be shown later, that is practically impossible and automatic transfer of knowledge was required.

11.2.2 Meta knowledge

During the investigation of the problem it became apparent that the implementation of the system should need *design engineering knowledge*. Such knowledge, however, was not documented anywhere in Rover. For this reason the author resorted to the development of an *automatic theory formation* program which would recover such knowledge and which would represent the learning element of the system. Although this program uses a similar approach to Meta-knowledge representation, frame-like structure of data, as the other systems in literature discussed earlier, it differs from all of them for different reasons.

NASL stores all the information (rules and data) in the same associative data base and resembles a script-based system rather than an object oriented one. Similarly, R1's knowledge representation is closer to hierarchical rather than OOP.

PIP, CENTAUR and PROSPECTOR basically represent only prototypes and they do not use a real OODB for their application.

META-DENDRAL shows the closest resemblance to ROOVESP - not at the knowledge representation level but in the way which the meta-knowledge information is obtained. That is, it tries to abstract the information by generating principles of the behaviour of the data in the sample. Even the two programs INTSUM and RULEGEN of META-DENDRAL can find equivalents in the design of the learning element of the system (see section 7.3). The motive of the automatic transfer of experience in the DENDRAL system, matches with the problem faced on this project: the lack of experts with enough specific knowledge to make a high performance problem solving program [42]. Finally, notice that META-DENDRAL, as the

learning mechanism **ROOVESP** does, takes account of the noise in the data sample.

The only major design difference between **META-DENDRAL** and **ROOVESP** is that **ROOVESP** uses the advanced technology in object oriented data representation (ie. **FLAVORS** and **Statice**) to 'abstract' the knowledge from the data rather than simple data triples (object, property, values). In data triples, type inheritance can only be coded with primitive nested lists whereas such relationships are hard-wired with the use of advanced OOP environments such as **FLAVORS** or **CommonORBIT** (see [99] for more details).

Another difference of **META-DENDRAL** with the learning system of **ROOVESP** is that the size of its research space can be extremely huge. For example as stated in [42] "the size of the space to consider for subgraphs, containing 6 atoms, each with any of (say) 20 attribute-value specifications, is 20^6 possible subgraphs. This makes it to be used only for "successful demonstration of scientific capability".

On the other hand, **ROOVESP's** learning system 'cuts down' the research space by applying statistical markers (derived after empirical analysis of the data and format of the **EJA18V** document).

11.2.3 Interactive transfer of expertise

Another way of "handcrafting" knowledge bases for expert systems apart from the automatic theory formation programs which have been discussed up till now (ie. **META-DENDRAL**, **PROSPECTOR**, etc), is *interactive knowledge transfer* programs. These programs facilitate the interactive transfer of knowledge from a human expert to the system using a high level dialogue. Examples of such paradigms are **TEIRESIAS** [32], **EXPERT** [29] [30], and **SEEK** [33].

The most important characteristic of these systems is that they embed explanation facilities for the conclusions of the system which could guide the user in the way that the new inference rules were acquired (a kind of a "debugging mechanism").

Especially, **SEEK** - used in rheumatology - is of great importance. It uses *case experience* data, in the form of stored cases with known conclusions to *refine* the rules of the model. It does this by using either *generalisation* (weaken the rules's IF part) or *specialisation* (strengthen the rules's IF part), if the results are not the desirable ones. **SEEK** distinguishes three different levels of confidence of the rules acquired: *Possible*, *Probable* or *definite*.

TEIRESIAS, **EXPERT** and **SEEK** give useful ideas for the further development of the Meta-knowledge mechanism of the system. The interactive transfer of knowledge matches with one of the fundamental objectives that led to the implementation of the learning mechanism of the system. That was, not only to support the overall system but also to represent the starting point of the documentation of the missing information in Rover.

The further implementation of the learning mechanism of the system to interactively advise the engineer for the cause of a conclusion derived, could help in its rule refinement. Notice, that the concepts of *possible*, *propable* and *definite* as they have been used in the **SEEK** system are also used in this system, for the specification of the design feature groups in relation to the part description. Consequently, the implementation of similar ideas in rule refinement such as used in **SEEK** have major potential for use in **ROOVESP**.

11.2.4 general discussion and summary on the learning element of **ROOVESP**

Lenat in [33] distinguishes three kinds of theory in heuristics: 0th order, first order and second order. The 0th theory says that the relationship *APPROPRIATENESS(Action, Situation)* is continuous and time-invariant. This means that given an action, the appropriateness of this is always valid for that situation.

The first-order theory says that "the 0th-order theory is often a very useful fiction. It is cost-effective to behave as though it were true. Notice that the 1st-order is itself a heuristic !" [34].

The second-order theory says that heuristics are compiled with hindsight. As new empirical evidence accumulates, it may be useful to 'recompile' the new hindsight into heuristics (synthesize new heuristics and modify old ones).

Summarising, it can be said that the learning element of ROOVESP can be further implemented and reach the **second-order** theory of heuristics, as expressed by Lenat, by adopting a mechanism with which the engineer could view the acquisition of the inference rules of the system and appropriately modify it in a way similar to the **SEEK** example. The learning element of ROOVESP reaches at least the first-order theory of heuristics: when the project started the state of design engineering knowledge was very incomplete. Currently, such knowledge is well understood and, as mentioned on the above, it is open to further correctness of its acquisition. In between, the heuristic search of the learning mechanism is a useful paradigm.

Another relevant paper to the learning of heuristics field is that of D.A. Waterman [35].

11.3 How ROOVESP affected the existent environment of the PSC in ROVER.

This section discusses a series of "thoughts" or concepts which were existent in the company and clarified during the investigation in this thesis.

ROOVESP was the system which enabled all those "thoughts" to be refined and linked together with the new concepts that were invented to overcome the various inconsistencies of the existing

PSC in Rover. In this perspective, ROOVESP represents the means by which old and new concepts are linked to form Rover's new vehicle specification conceptualism: That is the automatic validation of the usage statements of all the parts of the vehicle and the ultimate aim for the automation of the whole PSC.

The last section of the chapter discusses the new concepts which were introduced in the company during this research and as a result of it.

In this section only the "thoughts" which were existent and which were clarified during the research are discussed. In addition, some other contributions of ROOVESP to the functionality of the various departments in the company such as Auditing, Specification Services, Business system, and Component Engineering are discussed.

11.3.1 Auditing department

11.3.1.1 methodical approach of the various problems

- for the first time the audit function was analysed methodically and three phases were clearly distinguished whose internal operations defined unambiguously (chapter 5).

- a validation algorithm was methodically defined and actually implemented, which was empirically tested and proved to be correct.

- the generation of the boolean qualifiers PLUS (+) and MINUS (-) in the final usage statements of the parts of the vehicle, were studied in a structured way and algorithmised for the first time.

11.3.1.2 enriching of the traditional auditing way

- the analysis carried out for the design of ROOVESP enhanced the existing audit function itself. It reformed its process which till now was tied into the traditional way of duplicate documentations (Features List, Territory Code Index Report, etc) and the struggle of understanding the same information in different formats.

11.3.1.3 ROOVESP matures the Auditing function.

The audit function has grown throughout the years. In the early days of its application there were many cases where auditors or engineers used to specify vehicle parts based on the model trim levels. That is the worst case because no real specified information is generated. That is, it is well known that every part is dependent on the trim level. Rover's specification concept has passed this level but is still hampered by heavy functionalism as stated in [20]: "My specification is better than yours". With ROOVESP the way of specifying the parts of a vehicle becomes standard.

11.3.1.4 **ROOVESP** removes the need of high degree of specialisation within the Auditing department.

5%	Experts in the majority of Component Areas
15%	Expert on one particular Component Area - eg. Power Unit, BIW
30%	Reasonably general knowledge in this group
20%	Novices
30%	Shortfall of Supply v Demand

Figure 91: The percentages of experience within the Auditing department

Figure 91 shows the distribution of knowledge and experience (and shortfall of experts) within the audit department as stated in the company's report [8].

This no longer needs to happen with the application of **ROOVESP** as the design information (coming from statistical analysis of the past data) is clear and available to everybody (ie. no need for auditors to be specialised in particular areas of the car such as seats, mirrors, harness etc.).

11.3.2 Business system

11.3.2.1 Feedback on the quality of the stored data.

The Business Systems department which maintains all the backup data tapes used for the implementation of ROOVESP, was feed-back with information regarding the wrong quality, of some of the tapes. For example, it was found that the APN, AFGIR, EJA18V and AFC formats of these tapes are not compiled thoroughly by ISTEEL. Sometimes these tapes even maintain corrupted data. This emerged because it is the first time that these tapes have been used by Rover for reasons other than hardcopy backup.

11.3.2.2 proposal for the correct implementation of the part descriptions database.

The analysis of the APN catalogue showed the Business system people the confusion which is created by the way in which they implement the APN database. That is that a feature description should not be inserted in the description of the part. The usage condition should discriminate the parts and not the human conceptualisation.

11.3.3 Specification Services

11.3.3.1 ROOVESP 'documents' the way for the logical compilation of the AFC of the model.

- ROOVESP proves the need for a specific standardisation of the compilation of the Additional Feature Charts of the model. That brings to an end the controversy which has existed in Rover for many years. Feature groups which are offered with more than one feature availabilities in a model, should cover all the trim levels of this model in its AFC document. On the other hand, if only one feature availability exists for the feature group, it is advisable for the specification people to create its logical negation (let say, not heated seat) and compile it as a real feature availability of the AFCs of the model. This negative availability will then cover the rest of the trim levels in which the feature group under consideration is not offered.

- Although ROOVESP can tackle the unambiguous ways in which the restrictions of the features are coded in the AFC, it is advisable these restrictions follow a standard pattern in their compilation (see section 5.1.1).

11.3.3.2 refinement of the knowledge and experience in Specification Services.

- Specification Services people use their experience to compile the Base and Additional Feature Chart of a model. For example, if heated seat is introduced in the company for the current R8 model and this option is offered across to all Rover R8 derivatives, it sometimes happens that the specification people do not compile a new discriminating feature (ie. heated seat). However, if the company's policy was to change in the

future, the specification people have not only to create a new feature but to re-compile the old BAFC of the model, as well (a major task).

The analysis of the must and maybe feature groups provides a clear picture of the design engineering knowledge situation inside the company. It refines the past knowledge by the means of identifying the design feature groups which they are inclined to change throughout the time and the design feature groups (musts) which should always be compiled in the AFC of the model regardless of their standardisation across the whole range of the derivatives.

11.3.3.3 cost effectiveness for the company

By using the EJA18V and VPG tapes to extract past experience, ROOVESP makes these documents to worth their value which until now were used only for a report basis. Ironically, the actual EJA18V tapes till now have kept overwriting each other which makes their compilation from ISTEEL even a less cost effective investment for the company.

11.3.3.4 computer based update of the data and knowledge

Using ROOVESP's maintenance mechanism the update of the Rover's data such as the APN catalogue, the BAFC of the models, the K87 feature directory etc. are supported in a database logic, rather

than implicitly through the generation of new backup data tapes.

11.3.4 Product Development

11.3.4.1 **ROOVESP** can reduce the large amount of combination variants by instigating a better understanding of the product itself within Product Development (or Vehicle Directorate).

As stated in a survey by Aachen university [6] "one approach of reducing the number of variants is neglected: Reducing the number of the product variants starting in the design phase. Less variants in the design reduce the amount of work in every other department".

This survey introduces a method called Variant Model and Effect Analysis (VMEA) to control the number of product variants and when it was applied to an Audi 100 car pedal system, it was proved that appropriate management in design changes resulted in a reduction of 29.8% parts and 62.5% variants. A cost analysis showed a reduction of approximately \$2 per pedal system in a volume of 550 systems per day. Finally, the survey concludes that a 16 percent cost can be saved on the total product.

The design engineering knowledge which can be derived through ROOVESP can become the basis for such an analysis of variants at the design level. Remember, Japanese automobiles work on this principle and have reduced design time drastically (see table 5).

In addition, the Intelligent Networks prototype would greatly contribute to the rationalisation of assembly and subassembly structures (the Bill Of Materials of the products) as all the knowledge of the vehicle and the new changes are shown in the screen.

The survey [6] states, that when questions such as

- When does the variant first occur and which parts are involved in the variant?

- In which subassembly is it possible for a certain variant to occur?

are answered correctly, this would contribute to the correct analysis of the BOM of the product and consequently decrease its multiple derivatives.

In **ROOVESP**, the knowledge of the system stored in Statice represents the knowledge domain itself which also can be graphed in the screen. In that perspective **ROOVESP** can contribute to the better understanding of the product. This is discussed in more detail later.

As a matter of fact this survey - the only one which found in the literature to consider even implicitly the PSC in the automotive industry - came to the same conclusions as the research in this thesis, although from a totally different angle (ie. variants analysis):

- Firstly, **Object Oriented style**: The survey supports that "each unit represents a functional entity in the variant tree which is not necessarily devoted to a single function",

- Secondly, graphical breakdown of the assemblies is required to support design analysis: "Graphic representation of the information is preferred ... the variant tree tool is able to support the design, also" [6].

11.3.4.2 **ROOVESP** brings together all the various departments which are involved in the Product Specification Concept by clarifying their contradictory concepts and creating new concepts when is required.

Figure 92 shows the current situation in the company where many departments with their subsidiary databases are viewing the same information (ie. the product) from different angles. As stated in [20] "Today, across the company, many departments are receiving PIMS paper output, analysing, reformatting and re-keying data into stand alone systems. This, at best, is a duplication of efforts, or worst it leads to errors and continues the proliferation of autonomous rather than integrated systems".

In addition, with **ROOVESP** a common theory of communication is created (ie. must and maybe feature groups to the part description, validation procedures, assembly links algebra etc.) which guarantees to be and practically supported as one can refer to the existent results of the application software.

11.3.4.3 **ROOVESP** contributes to the implementation of existing

projects in Rover.

The relationship between ROOVESP and IPL has been discussed ie. the most appropriate software tool for such a type of problems and the use of intelligence in the field. IPL represents the first of a three stage project in Rover. PROMS represents the second stage where all the existing data together with the Land Rover's data will be collected into a common data base system. ROOVESP helps PROMS by supporting the existence of design engineering knowledge - as well as other knowledge - at the part description level and it has been proved that such knowledge can be derived automatically from past experience data. The IBOM (Illustrated Bill Of Material) project represents the third stage of the overall project. This includes the interface of the existing IPL (images and usage conditions) and PROMS (for Land Rover, as well) data with Rover's CAD system. The choice of the data base for the IBOM project coincides with the experience gained from this research ie. the appropriateness of Object Oriented Data Base for such a type of problems.

Figure 93 shows the intended implementation of the whole BOM project. ROOVESP falls somewhere inside the Product Definition field, whereas the intersection of the fields define the duplication of information and effort as the three different projects are under parallel development. For example, it is shown that PROMS overlaps with IBOM. This is because data are input both in the OODB (the one chosen by IBOM) and DB2 (chosen by PROMS). However, because a relational database such as DB2 has

been considered incapable of handling the CAD data (CADDs in the figure), the IBOM project is created which intends in the future to cover the whole BOM area in Rover (as the arrows indicating outwards).

This proves the validity of the search in this thesis in two ways:

- Object Oriented Programming and OODBs have been proved to be the most appropriate environment for the PSC in Rover,

- Secondly, the problem illustrated by the PSC has several generic features therefore the approach used here can be used in other problem areas that exhibit the same characteristics.

In addition, ROOVESP's feasibility study of the Object Oriented Paradigm and more importantly the implementation of its data base in a real OODB system such as Statice can help both in the investigation of this new field and more naturally guide the development of Rover's PSC to an Object Oriented environment.

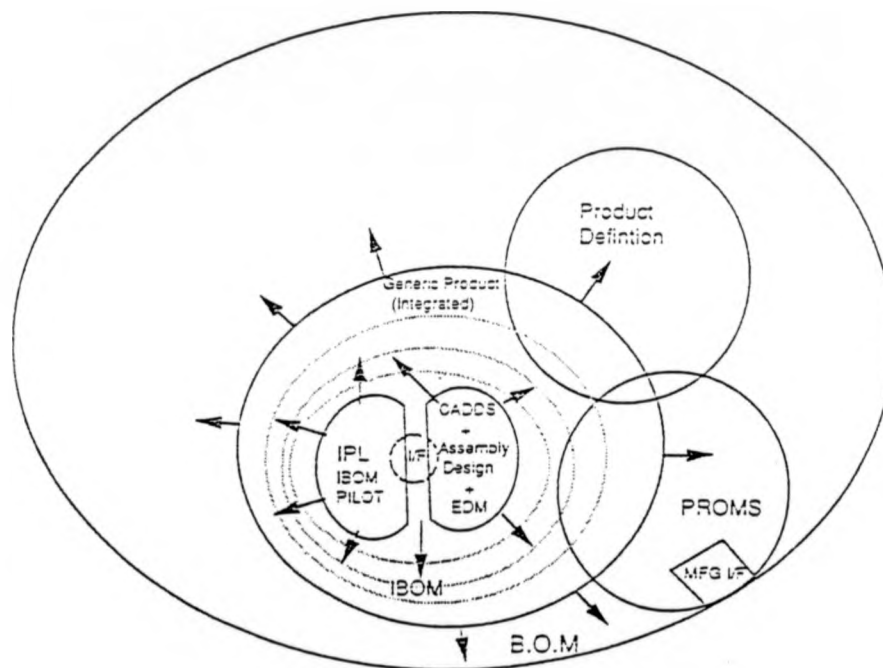
11.3.4.4 Decrease in the Specification time of a product

Decrease in the time spent in the specification of the parts of a vehicle is expected potentially to drop drastically:

Firstly, because of the automation which has been implemented on the Product Specification Concept and secondly, because the engineers do not need to liaise with the auditors. That is because all through the implementation of ROOVESP, engineers were considered to be the prime authors of the

Illustrated Bill of Material (IBOM) Project
Single Workstation for Text & Graphics

- * Interaction Between CAD & The BOM
- * Intuitive Graphical Front End (Windows-based)
- * Latest Technologies



Initial development focused on IPL functionality, then core CADDs, Assembly Design and EDM functionality and data enriched to encapsulate corporate BOM functionality.

Figure 93: The proposed for future implementation in an OODB Rover's PSC.

specification of the parts.

11.3.5 Component Engineering

11.3.5.1 **ROOVESP** makes interchangeability a 'visible' concept.

Interchangeability is a concept which has been discussed for many years in Rover and its definition is still under question (see section 9.2.1.1). This is because *interchangeability* faces different requirements for the various departments such as the Component Engineering, Manufacturing, Purchasing, Services etc. The prime author, however, for the definition of the interchangeability of a part, is the component engineer.

The use of the interchangeability concept in the *Intelligent Networks* prototype for the first time illustrates this concept to the engineer by breaking down to the lowest level the assembly structures of every single part. In this perspective, interchangeability becomes visible to anyone and easier to maintain.

11.3.5.2 **ROOVESP** points out for discussion the correct need for the creation of new physical parts.

The *Intelligent Networks* prototype discusses the needed conditions for the creation of new parts (new part numbers) in

the assembly of the new designed part. It advises that care is taken and new parts are created only when necessary [100]. For example, is not need for the creation of both a new heated-overlay and new heated-squab (see section 9.2.1) which adds confusion in the maintenance of the APN catalogue.

Having in this section discussed the various concepts and or "thoughts" existent in the company and the way they affected from the development of **ROOVESP**, the following section summarises the objectives of this thesis, as drawn during the search. It also, discusses the new conceptualism which **ROOVESP** introduced in order to overcome inconsistencies of the existent PSC and automate both this and the audit function.

11.4 The new conceptualism introduced in the PSC by **ROOVESP**

When the project was launched there were strong doubts about the successful application of computer technology in the field. That was due to the nature of the problem domain itself which is highly based on the "common sense" and "experience" of the people. Thus, the initial objective of the project was restricted in replacing the paper and pencil of the auditing function by electronic equipment. There was little attention to substitute the auditor's capabilities, instead the approach was to provide as much support as possible, partially using a rule-based system to apply basic validation procedures and partially by just

providing an appropriate user interface. The central part of the Auditing process based on experience and intelligence viewed on the time to remain the task of the human beings, however.

The implementation of the rule-based system to validate the specification packages coming from Component Engineering was faced with two problems which from early on showed that the Audit process cannot be isolated from the more general Product Specification Concept in Rover which it supports.

Firstly, no clear audit process existed.

Secondly, its computerisation would need design engineering knowledge which was not documented anywhere in the company.

These two problems when examined closer uncovered inconsistencies in the Rover's PSC which would need to be corrected in order for the environment to reflect the futuristic automation of both the audit function and the PSC. This naturally led to the study and understanding of the various vehicle specification concepts or thoughts - some of them discussed earlier - which had to be clarified and corrected when required. When the inconsistencies of the PSC could not be solved by clarification or correction of the existent concepts, new ones were developed.

The application of software in both the audit function and the PSC could then be designed on the basis of this new vehicle specification conceptualism.

Thus, it can be concluded that the research of this thesis not only examines the application of computer technology in a

traditionally human oriented environment such as Auditing and Vehicle Specification, but also analyses and "documets" these environments in order that they can be automated afterwards.

11.4.1 unclear audit process

The unclear definition of the audit function forced the investigation of the project to be viewed from a new perspective. That was the association of part descriptions with design engineering feature groups at an abstract level.

The Business system department [9] was of the view that no such relationship existed but rather simple Manufacturing quantitative links based on the assembly structures of the parts.

This research however, has proved that the Features List document is not required and the parts specification can be extracted solely from the BAFC. Especially, as missing parts information from this document (normally presented in the Features List) can be retrieved.

Thus, as the complexity of the audit function gradually simplified, it became clear that in reality feature groups do relate with physical parts at the part description level. The only problem was the huge quantity of the design engineering knowledge which should be documented manually from the experts to support the system.

11.4.2 need for design engineering knowledge

At this stage it became apparent that the original search should be expanded outside of the scope of the Auditing department and encompass the whole Product Specification Concept, including all the functionalities and the concepts of the departments which are involved in its process.

The ~~past~~ and ~~maybe~~ feature group concepts introduced in the company together with a statistical algorithm which was implemented to manipulate the internal representation of finished product documents such as EJA18V (specification patterns and data analysis). It proved that design engineering knowledge could be obtained for the company on the basis of this new conceptualism.

This design engineering knowledge represents past experience accumulated in PIMS as long as PIMS is in existence. Its quantity is huge - ie. 10 (models) * 5,000 (parts) * 200 (feature groups). ROOVESP in a further step beyond the statistical analysis of the part description level obtains design engineering knowledge for higher levels in the parts data hierarchy (ie. families, VPGs) by abstracting from what it already knows.

This Meta knowledge element of the system represents the starting point for the documentation of this knowledge in Rover. A major achievement considering that the literature survey in automobile industry has revealed that no such knowledge is anywhere documented for any company.

11.4.3 A real system

The third problem which was faced during the investigation of this thesis was the implementation of a real life system to support Rover in an everyday basis.

The implementation to a real system led in the investigation of both software programming environments and software database technology which would support both the Auditing and the PSC efficiently and flexibly. That resulted in the choice of the AI environment: Object Oriented Programming and actually the latest software technology in the field: real Object Oriented Databases. Real Object Oriented Databases when compared with traditional database approaches it was proved that they support more closely the complicated relationships of the problem and still can be efficient and accurate.

Statice, a relatively new OODB product from SYMBOLICS, was explored and chosen to support the PSC in Rover.

After, the full implementation of the first Audit phase, it proved that the objectives of the project not only successfully met but still could be more ambitious for the future:

The system does not only replace pencil and paper but intelligence, as well, and can be a realistic working system for Rover. The following step, under development, is the full automation of the Product Specification Concept and

its integration to the rest of ROOVESP.

The work which has been done in the *Intelligent Networks* prototype proves the feasibility of the latter objective and represents that such an achievement is a matter of completeness of the data in the database and time rather than research.

The following section discusses the general aspects which were learnt during this research and some figures are given which both help the better understanding of the contribution of ROOVESP to a realistic application for the company and the need for it.

11.5 What can be learnt from ROOVESP

11.5.1 integration of the existing systems

The PSC in the automotive industry represents a typical example of problems which exist in big companies. The large amount of information which is involved in the running of the business gives to the proliferation of many different systems - possibly computerised - which are developed in a such a specialised way as to solve only different individual areas of the same problem without necessarily integrating naturally with each other (see figures 92, 93 existing and future implementation of the PSC in

information, usually in different formats. It leads to different interpretations of the same source of data knowledge, the high specialisation of the various departments involved and finally the gradual isolation of one from the other.

The original difficulty of solving the specific problem within the company is transferred to the management of complexity which constantly grows by specialising the solution of its subproblems.

In such situations, especially when multiple computerised systems exist where each one maintains its own data base, one solution to the problem is the integration of the systems to a single database and with different procedures in the application program meeting the various functional requirements the different needs of the various departments rather than the representation of data within each database. The representation of the data in the new data base, however, should represent the domain in the most natural way in order to ease the development of such procedures and help the designer to globally maintain the changes done in the database from any angle.

11.5.2 Object Oriented Databases are suited for interactive update problems

The need for the natural representation of the knowledge of the

domain finds as one of the best software environment candidates, the Object Oriented Paradigm. In addition, the specific characteristics of the PSC and generally any other problems of this nature suggests the use of the OOP. This is the need for interactive processes in the data base between the user and the data.

The PSC falls in the category of CAD/CAM like exercises. As discussed in chapter 7, the PSC has been invented to help specify new derivatives on the basis of the existing ones. Consequently, not only in the re-design of already existent parts for which market research has discovered inefficiencies but even in the case of the insertion of a new vehicle, the whole process of the PSC is dependent on changes based on already existing data.

The Object oriented databases, more specifically Statice as used in this research, are designed to work for a such class of problems. The user can refer directly to a specific section of the database under consideration, as only the relevant data is 'pulled out' because of the 'by identity' relationships between all the data of the section. The data can then be updated as required on the screen, by manipulating their presentation types. This represents the most natural way of the interactive update of the data base as the user thinks not in the terms of the internal unfriendly representation of the data but their connotative information which has been conveyed into graphical representations such as a part, an assembly etc. This information, however, is the actual data themselves. Consider that in an engineering environment, engineers are not computer

experts which makes OODBs the most useful because of its naturalness to describe data and express graphically connovative information.

In the traditional way of storing data, those data bases have been designed to work with queries which search all the data in the database rather than just a specific section.

It has been estimated by Rover (confidential [93]) that "the inability of the current system in the company to respond to change (mainly because of the way in which the data are stored in the data base) not only hinders performance at the development phases, but also impinges on the company's ability to exploit market trends."

Further investigation on the current system has shown that the unnatural way in which the data are stored in the data base leads engineers to re-design already existing parts in the data base because it is difficult to access the section of the information with which they are concerned and consequently they have no full knowledge of the database contexts (see section 9.2.1).

11.5.3 The new world of OODBs

Statice is a typical example of the new database technology: real OODBs. It has been found that it can be a realistic database

system (ie. making use of pathnames, concurrency control, etc.) and work in a real life application when the indexes and clustering techniques are implemented properly in the system.

There are reservations regarding the amount of time that is required to create a large and complicated database (for **ROOVESP** that has been estimated to be 5 days). The performance issues which were discussed earlier in this chapter were based on only a representative amount of data; not a large one. In this case it was found that Statice and hence OODBs in general are much faster than the traditional databases. On the other hand, for large mainframe applications Statice shows lower performance than the traditional databases (it compares to the performance of relational databases 3 to 5 years ago). However, the author is of the opinion that OODBs (which have been in the market long enough) are still preferable for such type of problems as **ROOVESP**. This is because nowadays the software technology is moving in this direction and consequently more efficient software will be available in the future. In addition, the main reason for performance, ie. hardware, is changing rapidly and on-going improvements in CPU and disk speed which constantly will make OODBs much faster. Even with the existing performance the implementation of a system in OODBs is faster than rewriting an existing application, especially as much time can be gained during the development phase because of the naturalness in OOP.

Lufthansa which uses Statice in a real life application has created 50 thousand objects which correspond to 50 Mega Bytes of storage memory. These objects are massively interrelated and no

problem in performance has arisen. They support that a future implementation of 300,000 objects in the Statice database (corresponding to 100 MegaBytes) with equally complicated interrelationships will not make any difference in the performance of the data base [92]. SYMBOLICS supports that up to 2 Giga Bytes can be handled by Statice. Rover's current PIMS data base is almost 1.2 Gigabytes [92]. Consequently, there is strong indication that Statice could replace PIMS or at least a major part of it (ie. the most complicated areas) and still maintain higher performance than the traditional databases approach. However, this still remains a challenge for the future.

11.5.4 experience and knowledge acquisition

In most big companies there is a huge amount of data which is stored usually in different kinds of computer data bases. It can be said that these data reach the second order of knowledge representation: "information". They do, in reality keep information for specific areas in the company. However, using the expression stated in [84], these data are not knowledge, any more than an encyclopedia is knowledge. This information can be translated to knowledge by understanding it. This means the study of the **internal structure** of the way in which the data are organised in the flat files and the way in which the data is

maintained. Consequently, though the data in the flat files does not intentionally represent knowledge data, its format in the data base can be used in combination with rules which govern its maintenance and general rules of the way in which company run its business.

For example, when the internal structure of the data of a finished product (EJA18V) was studied, it showed that the need for such information to cover various needs of Rover, included identifiers which might help in the refinement of the "meanings" of this information such as dates, code replacements of one part from another, feature combinations etc. (format information). Consider the above knowledge with the fact that a part represents a unique design intention of mutually exclusive alternatives at a particular point in time (company's general knowledge) and knowing that a part's usage statements are date stamped (document maintenance knowledge). This can enable one to derive that if two different usage conditions also exist for the part, at the same point in time and one of these includes less design feature groups, and it can be proved that the missing feature groups can be added in a valid combination, then it could be determined that this feature group is a *maybe* for the design of the part (design engineering knowledge, see section 8.2.2.1.7) where an example is stated).

This understanding of the internal structure of the data in combination with the various rules results in the ability to use additional tools for a more thorough investigation. In our case it was the validation algorithm (checking for combinations), the

invention of specification patterns and statistical analysis.

11.6 The new conceptualism and Rover

11.6.1 How Rover reacted to the new concepts introduced by ROOVESP in the PSC.

ROOVESP was implemented on the basis of the business knowledge extracted during the investigation of the PSC in Rover. Naturally, the new conceptualism which was introduced, it was expected to be acceptable from the people in the company and it did. In fact, the new concepts which were introduced did exist implicitly within the various departments but it was difficult to be expressed or justified without the existence of an application environment ie. a computer system.

For example, the must and maybe feature group concepts, as it discussed in section 8.2, are implemented implicitly within the BAFC document. In a similar manner, implicit reference to such concepts has been expressed within the company during the specification of a part or the retrieval of the required volume of production (see section 11.3.3.2 and [100]).

The inefficient way in which the description of the parts is compiled in the APN catalogue, has been discussed over the years in Rover [88], [7]. Thus, the "negative to the part description"

feature has been accepted by the company. Similarly, the compilation of the BAFC in the way which ROOVESP suggests (see 11.3.3.1) is, also, globally acceptable.

The integration of all the systems to a single data base is an issue which has been proposed overtime from certain departments within the company ([7], [18], [91]). In fact, the present tendency of Rover towards an OODB (see 11.3.4.2) shows the acceptance of the ROOVESP's philosophy and/or software environment.

The Intelligent Networks prototype of ROOVESP coincides with the objectives of a supplementary to ROOVESP project, which was under consideration at the beginning of this research [18] and now is in progress (see 11.3.4.1): *the rationalisation of the Manufacturing assemblies*. For more details in this subject the reader can refer to [6] and [13].

Auditing, in general, has accepted ROOVESP because of its user interface facilities, the automatic validation of the part specifications and also due to the shortfall of expertise demand in the department. However, some auditors are concerned of the changing role of the auditing function within the PSC in Rover.

Engineers, as it was discussed in section 8.3.1 [100], have expressed enthusiasm for ROOVESP. Now they can see that liaison with auditors (with the consequent drawbacks ie. delays, misunderstanding, need of their profession, etc.) becomes obsolete.

In summary, it can be said that ROOVESP was met with surprise and sometimes enthusiasm rather than scepticism.

11.6.2 Recommendations to Rover.

The recommendations of this research to the company are:

(i) Rover should turn towards the integration of all the existing systems to a single master database.

(ii) Knowledge should be attempted to be retrieved from existing data base for the various aspects of the whole business ie. Design Engineering, Purchasing, Services, Manufacturing, etc.

(iii) Simply, a user friendly system is not sufficient (as it was proved with IPL) to tackle the complexity of the PSC. Intelligence should be incorporated and coded within the procedures which run the business.

(iv) ROOVESP should be extended to encompass the rationalisation of the Manufacturing assemblies. Such knowledge should be used "up-front" in the "Design to Manufacture" circle of each new product ie.

(v) The prime author of the specification of the parts should be the engineer and only him.

(vi) OOP and OODB is the recommended software environment to support the complexity of the PSC in Rover and the ongoing developments required for the integration of the BOM data with the CAD data.

In summary, expert knowledge can be derived by using existing information which only vaguely exists on a research field such as the format of a document, interrelationships among pieces of information and with general knowledge of the companies business. The full understanding of such business and the correct association of the different pieces of information can ease the further development of knowledge abstraction of the information space under investigation by methods which are already existent such as practical results (statistical markers), statistical analysis, or even with outside procedures (validation algorithm).

In the following chapter the conclusions of this research are discussed.

12 CONCLUSIONS

Below are listed the contributions of ROOVESP to the company.

ROOVESP could potentially replace the load of paper with which auditors or other departments work. This load of paper can easily be compared with twice the whole Encyclopedia Britanica.

ROOVESP standardises the method of the specification of parts irrespectively of subjective interpretations.

The validation algorithm of ROOVESP not only validates Usage Conditions for the parts but actually creates them.

ROOVESP proves that the Product Specification Concept (and its accompanied Auditing function) can be automated.

With ROOVESP the time for the specification of a vehicle will eventually drop to 90%.

Whilst developing ROOVESP it was found that existing information could be translated to knowledge valuable to the company.

ROOVESP generates design engineering knowledge which currently is

documented nowhere else. The techniques developed in ROOVESP can be extended to extract knowledge for other areas of the company eg. Services, Purchasing, Manufacturing, etc.

A major contribution of ROOVESP to Rover is that it assigns the various tasks to the departments clearly so that the maintenance of the design engineering knowledge is assigned to the Specification services, the creation of the usage statements of the parts in the engineers etc. Consequently, when a mistake occurs its responsibility can be easily tracked back to its source and more easily corrected which increases the discipline in the PSC.

Categories of problems such as CAD/CAM need data types other than the traditional ones in the conventional world such as strings, numbers, etc. Especially there is a need for images and the internal natural representation of new data types embedded in the OOP brings such a software development environment even closer to Engineering and Manufacture.

OODBs will not replace data models for traditional mainframe applications such as payroll and accounting. Rather, the object oriented data bases will replace data models required many data types such as knowledge based systems, CAD/CAM, CIE, and workstations in general.

13. REFERENCES AND BIBLIOGRAPHY

13.1 References

1. Ciccarilli Eugenie, Presentation based interfaces, MIT Press, USA, 1984.
2. Bromley Hank, Lisp-Lore: a guide to programming the Lisp machine. Cluuea Academic publishers, 1986
6. Becker Torsten & Schuh Günther, Variant Mode and Effect Analysis: A new approach for reducing the number of product variants. Laboratory for machine tools and production engineering (WZL) of Aachen university, 1988.
7. Rover Group & Advanced Technology Centre (ATC), PIMS:UC TO PARTS INTERFACE PROJECT - BUSINESS ISSUES AND PROCEDURES, 1990.
8. Kirby Aidan, Specification Services: Auditor Dilemma, Rover Group, June 1988.
9. Interview with Mr. John Layes from Rover Group, Business System, June 1989.

10. Briggs W. & Chester L. & Mustard R., Product Derivative Complexity, Coventry Business School, October 1989.
15. Kirby A, Reports Available from PIMS. Rover Group, July 1988.
18. Rover Group (confidential), CIE database project, November 1988.
19. King CD, Specification Release Procedure, August 1989.
20. Naylor A, Conflicts in Specifications. Rover Group, April 1987.
21. Apple, 4th Dimension, Apple, 1988.
22. XEROX, Second City Systems Group, 1988.
23. Simmonds D, PIMS, Rover Group, March 1985.
24. Rover Group & ISTEI, Colour & Trim Master Chart Release, June 1990.
25. Rover Group & ISTEI, APN/TPC BASED PRPTOTYPE PARTS LIST, 1990.
26. IBM seminar, WERS (Worldwide Engineering Releasing System), September 1986.

29. Barr Avron & Feigenbaum Edward, The handbook of AI: volume 1 - SEMANTIC NETWORKS pg. 181 - 221. HeuristECH press, 1983.
30. Weiss Sholom & Kulikowski Casimir, Proceedings of the Sixth International Joint Conference on AI - volume 2 - EXPERT: A SYSTEM FOR DEVELOPING CONSULTATION MODELS - pg. 942-947, Tokyo, August 1979.
31. Allen John, Anatomy of Lisp, McGraw-Hill Computer Series, 1978
32. Davis Randall, ARTIFICIAL INTELLIGENCE magazine - Interactive transfer of Expertise: Acquisition of New Inference Rules - Vol 12 -pg 121 -157. 1979.
33. Politakis Peter & Weiss Sholom, ARTIFICIAL INTELLIGENCE magazine - Using Empirical analysis to Refine Expert System Knowledge Bases - vol. 22 - pg. 23 - 48. 1984.
34. Lenat Douglas, ARTIFICIAL INTELLIGENCE magazine - The nature of Heuristics - vol 19 pg. 189 - 249. 1982.
35. Waterman D.A., ARTIFICIAL INTELLIGENCE magazine - Generalisation Learning Techniques for Automating the Learning of Heuristics - vol 1 - pg. 121 - 168. 1970.

36. Melle William, Proceedings of the Sixth International Join Conference on AI - A DOMAIN-INDEPENDENT PRODUCTION-RULE SYSTEM FOR CONSULTATION PROGRAMS - vol. 2 - pg. 923 -925, Tokyo, August 1979.

37. Gaschnig John, Proceedings of the Sixth International Join Conference on AI - PRELIMINARY PERFORMANCE OF THE PROSPECTOR CONSULTANT SYSTEM FOR MINERAL EXPLORATION - vol. 1 - pg 308 - 311, Tokyo, August 1979.

38. Aikins Jan, Proceedings of the Sixth International Join Conference on AI - PROTOTYPES AND PRODUCTION RULES: AN APPROACH TO KNOWLEDGE REPRESENTATION FOR HYPOTHESIS. Tokyo, August 1979.

39. Szolovits & Pauker, Readings in Medical AI pg 219 - 229, WJ Chancey & EH Shortliffe, Addison-Wesley Publishing Company, 1984.

40. Barr Avron & Feigenbaum Edward, The handbook of AI - PROSPECTOR - volume 2 pg. 155 - 162, HeuristECH press, 1983.

41. Barr Avron & Feigenbaum Edward, The handbook of AI - Meta-DENDRAL - volume 3 pg. 429 - 437. HeuristECH press, 1983.

42. Buchanan Bruce & Feigenbaum Edward, ARTIFICIAL-INTELLIGENCE magazine - Dendral and Meta-DENDRAL: Their Applications dimension - vol 11 - pg 5 -19, 1978.

43. Tanimoto Steven, The elements of AI - chapter 7 pg 239
Probabilistic reasoning, Computer Sciences Press, 1987.
44. Lenat Douglas, ARTIFICIAL-INTELLIGENCE magazine - EURISCO: A
program that learns new Heuristics and Domain Concepts - vol 21 -
pg 61-97, 1983.
45. Barr Avron & Feigenbaum Edward, The handbook of AI -
AM - volume 3 pg. 439 - 451, HeuristECH press, 1983.
46. Tanimoto Steven, The elements of AI - chapter 4 pg. 89 -
Knowledge representation, Computer Sciences Press, 1987.
47. Sussman Gerald & Steele Guy, ARTIFICIAL-INTELLIGENCE
magazine - CONSTRAINTS-A language of expressing almost
hierarchical descriptions - vol 14 - pg 1-39, 1980.
48. Sacerdoti Earl, ARTIFICIAL-INTELLIGENCE magazine - Planning
in a Hierarchy of Abstraction Space - vol 5 - pg 115-135, 1974.
49. Fikes Richard, ARTIFICIAL-INTELLIGENCE magazine - STRIPS: a
new approach application of theorem proving to problem solving -
vol 2 - pg. 189-207, 1971.
50. Barr Avron & Feigenbaum Edward, The handbook of AI -

PLANNING and PROBLEM SOLVING - volume 3 pg 515 - 562,
HeurisTECH press, 1983

51. Korf Richard, ARTIFICIAL-INTELLIGENCE- magazine - PLANNING
AS SEARCH: A quantitative approach - vol 33 - pg. 65 -87, 1987.

52. Winston Patrick & Brown R, AI: an MIT prospective - vol 1 -
pg 35 -49, MIT press, 1979.

53. Barr Avron & Feigenbaum Edward, The handbook of AI -
MOLGEN - volume 3 pg. 551 - 562, HeurisTECH press, 1982

54. Stefik Mark, ARTIFICIAL-INTELLIGENCE magazine - Planning
with Constraints (MOLGEN: part 1 & 2) - vol 16 - pg 111- 168,
1981.

55. McDermott John, NASL, 1977

56. McDermott John, ARTIFICIAL-INTELLIGENCE magazine - RI: A
Rule-Based confogurer of computer systems - vol 19 - pg. 39 -88,
1973.

58. Martin Robert, EXPERT SYSTEMS USER magazine - Object
Oriented databases, May 1990.

59. SYMBOLICS Inc, message mail from USA SYMBOLICS

Inc(confidential) a search upon real OODBs, 2/12/90.

60. Shipman David, ACM Transactions on Database Systems - The Functional Data model and the data language DAPLEX - Vol 6 - No 1
- pg. 140-173. March 1981.

61. Smedt Koenraad, AI Programming environments - Object Oriented programming in FLAVORS and CommonORBIT - pg. 157-196,
Robert Hawley, John Wiley and sons.

62. SYMBOLICS Inc, release 1. 22/8/1989.

63. Stahl J. & Timmermann, AMS - An Aircraft Management System
confidential. SYMBOLICS Inc, 1989.

64. Rajan Tim, EXPERT-SYSTEMS magazine - Object Oriented Programming Techniques - pg 12-15. December 1988.

65. Stonebraker Michael, IEEE - Object Management in POSTGRES using procedures, 1986.

66. Hull Richard & King Roger, ACM computing Surveys - SEMANTIC DATABASE MODELLING: Survey, Applications, and research issues -
vol 19 - pg 201-259. 1987.

67. Hudson Scott & King Roger, ACM Transactions on Database Systems - CACTIS: A Self-Adaptive, concurrent implementation of

an Object Oriented Data Base Management System - vol 14 - pg 291-321, September 1989.

68. King Roger & Hudson Scott, ACM - Object Oriented Database support for Software environments - pg 491-503, 1987.

69. Wallrath Mechtild & Lockermann Peter & Kemper Alfons, ACM - An Object Oriented Data Base System for Engineering Applications - vol 19 - pg 299-311, September 1987.

70. Kent William (IBM), Communications of the ACM - "a simple guide to five normal forms in relational database theory" - Vol 26 - No 2, February 1983.

71. Hartzband David & Maryanski Fred, IEEE - Enhancing Knowledge representation in Engineering databases - pg. 39 -46, 1985.

72. Smith John & Smith Diane, Communications of the ACM - Database Abstractions: aggregation - vol 20 - number 6 - pg. 405-413. Wegbreit B, June 1977.

73. Grosky William & Mehrotra Rajin, IEEE - Image Database management. December 1989.

75. Atwood Thomas, IEEE - An Object Oriented DBMS for design

support applications - pg 299-307, 1985.

76. Maier David, IEEE - Why OODBs can succeed where others have failed. 1986

77. Blaha R. & Premerlani William & Rumbaugh James, Communications of the ACM - Relational database design using an Object Oriented methodology - vol 31 - number 4. April 1988,

78. Damon Graig (Ontologic), SIGMOND RECORD - VBase Object oriented database system - vol 17 - No 2 - pg 96. Report on the OODBs workshop held in conjunction with the "Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '87) Conference"; Thatte Satish Workshop chairman, (Artificial Intelligence Laboratory Texas Instruments Incorporated, June 1988

89. Winston Patrick, LISP. Berthold Klaus & Horn Paul, Addison Wesley, 1989.

92. Interview with ISTEEL & SUMBOLICS (February - 1991).

93. Rover IT strategy group, Report on task group survey: Rover BOM review. Rover, September 1989.

94. Interview with Rover's Forecasting and Finance people (J

Dalton, M. Hanttsinson, C. Bayley), June 1990.

95. Interview with Rover's Product Control Complexity team.

96. Polya, How to solve it, Princeton UP - 2nd edition, USA, 1957.

97. Schek H.J. & Pistor P., Proceedings from the VLDB conference: Datastructures for an integrated Data Base and Information Retrieval System, Mexico city, 1982.

98. Dadam P, ACM SIGMOND conference on Management Data: A DBMS prototype to support extended NF² relations. An integrated view on flat tables and hierarchies - pg 376-387, 1986.

99. Markomichalis Panayiotis, An exercise in Machine intelligence and rule generator using ART and LISP, MSc. thesis, Warwick University, Nov 1988.

100. Dmonsrtations of ROOVESP to design engineers Geoff Cowan (harness) and Paul Stevens (seats).

13.2 Bibliography

3. SYMBOLICS Inc, manuals for Statice, USA, 1988.
4. SYMBOLICS Inc, manuals for the Lisp Machine - 16 volumes, USA.
5. Auditing course
11. Interview with Mr Bill Jobshon from Rover Group, Canley, Product Change Control, March 1990.
12. Interview with Mr. DJ Wall from Land Rover, Lone Lane, Solihull, June 1990.
13. G. J. Burke & J.B Carlson, DEA at Ford Motor Company, Ford Motor company, 1988.
14. Auditing department, The PIMS database, Rover Group, May 1988.
16. Rowel J, Product Change Control, Rover Group, 1987.
17. Austin Rover (The Open Learning Unit), Open Learning: 3 textbooks, 1988.
27. Ford Company, The Ford Specification Concept, 1980.

28. Sacerdoti Earl, Problem Solving Tactics. Proceedings of the Sixth International Joint Conference on AI - volume 2 - pg. 1077-1085, Tokyo, August 1979.
57. Lauriere Jean-Louis, ARTIFICIAL-INTELLIGENCE magazine - A language for stating and solving combinatorial problems - vol 10, pg 29 - 125. 1980.
74. Alty JL & Coombs MJ, EXPERT SYSTEMS: concepts and examples. NCC publications, 1984.
79. Ullman D. Jeffrey, Principles of Database Systems pg 21-79. Pitman Publishing ltd, 1980.
80. Baddeley Alan, The psychology of memory - semantic memory - pg 317 -346. David Alan, Harper & Row, 1985.
81. Barr Avron & Feigenbaum Edward, The handbook of AI - Semantic Networks/Frames and Scripts - vol 1 - pg 180-220. HeuristECH Press, 1983.
82. Charniak Eugene, TINLAP-2 -Theoretical approaches in Natural Language Processing-2 - With a spoon in hand this must be the eating frame - pg 187-193. University of Illinois at Urbana-Champaign, 25-27 July 1978.
83. Schank Roger & Abelson Robert, Scripts, Plans, Goals and

Understanding. Lawrence Erlbaum associates Publishers, 1977.

84. Barr Avron & Feigenbaum Edward, The handbook of AI - representation of knowledge - vol 1 - pg 140-221, HeuristECH Press, 1983.

85. Schutzer Daniel, AI An applications oriented approach - pg 27-163. Van Norstand Reinhold company, New York, 1987.

86. Rich Elaine, Artificial Intelligence - Structured representations of knowledge - pg 201-243: Basic problem solving methods - pg 75-107. Mc Graw-Hill book company, 1985.

87. Interview with Mr. John Saulders for the Honda and Ford's PSC, Rover Group, June 1989.

88. Interview with Mr. Reg Darlington for the Chrysler's PSC, Rover Group, June 1989.

90. Fikes R, Artificial Intelligence magazine - REF-ARF: a system for solving problems state as procedures - vol 1 - pg 299, 1970

91. Engineering Services, Time for Change. Rover Group, 11 May 1988.