Sales Advisory System: A Case Study in Applying Knowledge Representation Systems

Vorgelegt von
Diplom-Informatiker
Sunil Thakar
aus Sonepat (Indien)

Vom Fachbereich Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr. -Ing)

Genehmigte Dissertation

Promotionsausschuß:
Vorsitzender: Prof. Dr. Klaus Obermayer
Berichter: Prof. Dr. Bernd Mahr
Berichter: Prof. Dr. Hermann Krallmann

Tag der wissenschftlichen Aussprache: 01.02.2001

Berlin 2001
D 83

To
Little Baloo (The Bear)
Abhinav

To the Crazy Ones.

The misfits.

The rebels.

The troublemakers.

The round pegs in the square holes.

The ones who see things differently.

They're not fond of rules.

And they have no respect for the status quo.

You can praise them, disagree with them, quote them,

disbelieve them, glorify or vilify them.

About the only thing you can't do is ignore them.

Because they change things.

They push the human race forward.

While some see them as the crazy ones,

we see genius.

Because the people who are crazy enough to think

they can change the world, are the ones who do.

source: Apple Computer „Think Different" Ad-Campaign, 1987

Abstract

The presented case study establishes the relationship between the principles of knowledge representation and reasoning systems and their embodiment in working application systems.

The results of the conducted study fall under two categories: one which receive everlasting scientific interest (knowledge acquisition, knowledge representation and reasoning methods, validation and man-machine interactions etc.) and one which are supposed to fulfill the business challenge (application related subjects typically include maintenance, reconfiguration, integration with legacy systems and new applications such as e-commerce) posed to them.

The fact that the sales advisory domain is so highly knowledge intensive forced us to concentrate on the representation and reasoning aspects of it's application irrespective of mode/model we choose to support this process with a computer system.

Further, in order to use knowledge representation and reasoning systems in real-world knowledge-intensive applications, we have tried to bridge the gap between theory and practice. A logic-oriented representation system is presented which allows the representation of the terminological knowledge of our application domain of automobile sales advisory within taxonomic reasoning. Based on our analysis, we found that the description logics BACK, FLEX and CLASSIC have sufficient expressive power to approximately represent the automobile sales advisory domain.

Objects and constellations which typically describe the automobile sales advisory domain are represented in order to evaluate the method. Various complex descriptions of objects, aspects of classification and taxonomic reasoning in the sales advisory process are discussed.

The investigation is illustrated by using examples from the real world and supported by various case studies in different but related fields. These examples -by virtue of their object-oriented characteristics- support an experimental mode of domain modeling and allow for a representation study on a larger scale. The resulting modeling experiences lead to an assessment of the knowledge representation and reasoning systems presented here. Further, it helped us answer: how the chosen KR&R system can achieve the increased efficiency and improved quality in sales consultation.

The onset of e-business, e-commerce and web-based selling are changing the practice of doing business and the expectations of the customer too. Customers not only expect instant answers to their questions, they demand instant responses to their inquiries as well. In other words they want efficient and good quality consultation as service wrapped around the product.

Improving sales processes, pleasing the customers and making sales quicker simply demands more. The demand from the marketplace for customized products with a decreased quoting cycle time has lead to the introduction of configuration tools also known as interactive sales solutions or interactive sales systems or sales advisory/consultation systems.

This sets the prevailing context in which the present case study was conducted.

Zusammenfassung

Die vorliegende Fallstudie stellt eine Beziehung zwischen Wissensrepräsentations- und Schlußfolgerungssystemen her und zeigt ihre Eignung für den Anwendungsbereich der Verkaufsberatung.

Die Ergebnisse berücksichtigen sowohl wissenschaftliche Aspekte, wie effiziente Wissensakquisition und Wartbarkeit und Konsistenz der Wissensbasis, als auch anwendungsorientierte Aspekte, wie Flexibilität des Datenzugriffs, Stabilität und Wiederverwendbarkeit.

Da die Domäne der Verkaufsberatung äußerst wissensintensiv ist, konzentriert sich die vorliegende Arbeit auf die Aspekte der Repräsentation und Schlußfolgerung unabhängig von der computerunterstützten Umsetzung des Beratungsprozesses.

Es wird ein logik-orientiertes Wissensrepräsentationssystem dargestellt, in der das terminologische Wissen aus der Domäne der Verkaufsberatung in taxonomischen Strukturen repräsentierbar ist. Die Analyse von BACK, FLEX und CLASSIC hat ergeben, daß die Sprache aller drei Systeme eine ausreichende Mächtigkeit besitzt um das Domänwissen der Verkaufsberatung zu repräsentieren.

Zur Bewertung dieses Vorgehens werden Objekte und Konstellationen repräsentiert die für Systeme zur Verkaufsberatung typisch sind. Beschreibung komplexer Objekte werden untersucht und unter Aspekten der Klassifikation und des taxonomischen Schlussfolgerns im Verkaufsberatungsprozess diskutiert.

Die Ergebnisse der Untersuchung werden durch Beispiele aus verschiedenen verwandten Anwendungsgebieten belegt, die durch ihre objekt-basierten Charakteristika einen experimentellen Einsatz bei der Modellbildung unterstützen und damit eine Repräsentationsstudie in größerem Umfang ermöglichen; aus den Ergebnissen der Studie wiederum lassen sich Aussagen zur Ausdrucksstärke des vorgestellten Systems ableiten.

E-business, e-commerce und web-basierter Verkauf verändern die Geschäftsprozesse und das Kaufverhalten der Kunden. Um diesen Bedürfnissen der Kunden gerecht zu werden müssen effiziente und qualitativ hochwertige Beratungsprozesse als Zusatzleistung angeboten werden.

Die Nachfrage nach individuellen Produkten und möglichst kleiner werdender Angebotserstellungsphase führt zur Einführung computerunterstützte Konfigurations- und Beratungssystemen. Sie bieten einen neuen Kontext zur Umsetzung der Ergebnisse Dieser Studie.

Preface

Efficient knowledge representation and reasoning is an important component of intelligent activity and is a crucial aspect in the design of knowledge based systems. This dissertation is about the design of real world industrial scale knowledge based sales advisory/consultation systems. In it, we look closely at consultation systems as they exist today in the automobile industry and we set out new directions for future development.

All new technologies develop within the background of a tacit understanding of human nature and human work. This provided the main motivation for our investigations. Our research rests upon one such approach, socio-cognitive engineering viewpoint, which integrates the prevailing two notions of human knowledge: a techno-centric and other considering knowledge as a social resource. This approach of socio-cognitive engineering to design human-technology systems goes beyond „user centered design“ in that it involves analyzing how people perform complex tasks in organizational settings and then applying that analysis to development of systems that are not just usable, but useful and appropriate to the workplace.

The study and analysis of the sales advisory task as reported here was greatly influenced by this direction. Besides this new direction in system development philosophy, this research is motivated by the fact that our domain of automobile sales advisory is highly knowledge intensive and the structuring of the domain knowledge is crucial for the development of the sales advisory systems. From hereon, our investigations were focused on the knowledge representation and reasoning aspects of the application. Here, yet another set of motivation was to establish the relationship between the principles of knowledge representation and reasoning systems and their embodiment in working application systems, and to bridge the gap between theory and practice by actually using the knowledge representation and reasoning system in a real-world knowledge-intensive application.

One of the main research lines in knowledge representation, which is a field of artificial intelligence, has been concerned with the idea that the knowledge should be represented by characterizing classes of objects and relationship among them. The organization of the classes used to describe a domain of interest is based on a hierarchical structure, which not only provides an effective and compact representation of information but also allows for performing the relevant reasoning tasks in a computationally effective way. This was the principle underlying the development of first frame systems and semantic networks. However, such systems were in general not formally defined and the associated tools were strongly dependent on the implementation strategies. The logic-based characterization of such systems has been accomplished through the work on KL-ONE system. Here many ideas stemming from earlier semantic networks and frame-based systems were taken and given a logical basis for interpreting objects, classes (or concepts), and relationships (or links, or roles) between them.

Further, the research efforts to pursue the following two goals: first, the precise characterization of the set of constructs used to build class and links expressions and second to provide reasoning procedures that are sound and complete with respect to the semantics, lead to the origins of description logics. Most of the research in description logics has covered theoretical aspects, implementation of knowledge representation systems and the realization of applications by means of such systems in several areas. Here the key element has been a

very close interaction between theory and practice. Though a substantial amount of work and well-established material is there to support the successful use of description logics in various application domains but a comprehensive study regarding the effectiveness of description logics in "real" industrial application is still sought for.

Our investigations are a step forward in this direction and concerned about the question of "why and how a specific KR system (BACK, FLEX, CLASSIC) is best suitable for an existing specific real-world complex application (automobile sales advisory)".

Rapid and increasing advances in e-business, e-commerce and web-based selling have broadened the arena of research in challenging applications in the sales advisory domain and have provided us with the prevailing context to conduct this case study.

The research is demonstrated in a conceptual model of a knowledge-based consultation/advisory system (KBCS). The system helps a salesman in a communicative situation (sales talk/consultation) by providing him/her with the necessary technical information required to complete the sales (i.e. providing the customer with a product/product version which satisfies all his/her requirements).

In describing KBCS, I have tried to clarify the assumptions and scope of the system. I also explain how it works and, more generally, how important design decisions relate to its function as a knowledge-based consultation/advisory system. I hope that the conclusion drawn from this research will be useful to the design of future systems for knowledge-based computer-mediated/assisted consultation systems.

A guide for the reader

This work is a doctoral dissertation, which by its nature covers not only the specific topic in some depth but also anchors it to the many related areas of research. I have attempted to organize the dissertation so that it may be read sequentially or by sampling particular topics.

Chapter 1 introduces the problem domain of sales advisory/consultation, what we mean and understand by various terms and concepts used such as knowledge representation and reasoning systems and socio-cognitive engineering viewpoint. It also provides the rationale and motivation for our work. Further, it offers forward references to detailed discussions in later chapters.

Chapter 2 provides a narrated walk-through the main aspects of the case study. First it describes the domain i.e. a consultative situation in buying a complex technical product. Then describes the existing solutions, discusses system functions and analyzes these systems and mention the problems faced. Following that, the (re-knowledge-engineering) methodology , as adopted by us, is presented.

Chapter 3 discusses the field of knowledge representation & reasoning, which is also referred to as KR. Here, we not only present the fundamental ideas of this field but give a short overview of historical developments as well. Further, this chapter deals with not only the deficits of KR systems but suggests KR systems based on description logic as suitable candidates for our domain of sales consultation/Advisory.

Chapter 4 describes major issues involved in human communicative situation (consultation) and try to answer certain questions regarding the process of sales advisory and the need to provide computer aid to make this process more efficient. It provides a survey of ***state of the art*** consultation/advisory systems and presents some ***Aspects of Consultation*** as well.
Chapter 5 deals with the concrete modeling aspects of our domain. The three different sections present three different knowledge representation systems belonging to the same family, and describe how the specific knowledge of the various products (cars, coaches & trucks) of an automobile company can be represented. It deals with questions which arise while modeling a knowledge area in the terminological system. These questions can only be answered in the context of the developing application. To this counts such principle decisions like at what level we represent the information as objects and as concepts. Furthermore, different points concern the choice of appropriate language expressions and the granularity of representation, i.e. how „deep“ should be the modeling and the syntactic point of view.

Chapter 6 compares and analyzes our work to other related areas in order to discuss the differences, commonalties and their benefits.

Chapter 7, the final chapter, offers conclusions drawn from the above mentioned research and future research directions which may later on help improve the KR- and Consultation systems.

Appendix 1,2 & 3 list some sales advisory systems from different automobile manufacturers

Appendix 4 provides the syntax of the KR systems CLASSIC, BACK & FLEX.

Appendix 5 discusses shortly the implications of the internet for automobile industry, in particular for the task of sales advisory/consultation.

Finally, Appendix 6 should help the reader to get acquainted with the future role of knowledge representation (in XML format) in the wake of web and electronic commerce

This text assumes that the reader is familiar with basic concepts in Artificial Intelligence, such as knowledge representation and search. It also assumes an acquaintance with knowledge systems - what they are and how they work - but it is not necessary to know how to build one.

Acknowledgements

The work that I present here is the result of my long years of involvement in research both at academia and industry. My research would have been impossible had I not received sound advice, encouragement, moral and technical support from various talented people to whom I owe a tremendous debt of gratitude.

I am greatly indebted to my principal advisor Prof. Bernd Mahr for valuable discussions, constructive criticism and helping me to clarify my ideas to present them understandably. I would like to thank my second advisor Prof. Hermann Krallmann for his knowledgeable comments, suggestions and insights in the field of today’s business and organizational environment. Thanks is due to Prof. Klaus Obermayer for accepting the chair of the dissertation committee and to Prof. Guenther Hommel for his invaluable encouragement and inspiration.

Some times less is more, thus in this sense, Joachim Quantz deserves a ***specia***l thanks because without him I wouldn't have been able to finish this work. His kindness and intellectual clarity inform this work in countless unattributed ways.

My thanks also goes to the management of DaimlerChrysler AG (DC), my present employer, and especially to Dr. Rolf A. Mueller, manager, Organizational Computing, whose greatest single contribution to this project has been the freedom to work (in his own words... "in research, you should represent your own independent view"). During all these years at Research & Technology Division of DC, I have benefited from discussions with various colleagues (past and present), especially Roderick Murray-Smith, Oliver Seils, Rudiger Klein, Klaus Goos, Ralf Escher and Anette Standfuss, who provided helpful advice.

Thanks are due to all the members (past & present) of the KIT-Group/Technical University Berlin, especially who were involved at various stages: Jan Thomson (for introducing me to KIT-Group and to the basic concepts of BACK) Uwe Küssner, Albrecht Schmiedel, Klaus Schild (for various discussions) and Christoph Peltason (for his constant intellectual support and permanent faith that something would come of it).

In my own field of Artificial Intelligence, I've enjoyed the intellectual companionship and personal friendship of Achim Hoffmann, who has been a constant and great source of encouragement and inspiration.

Jason Harman and Miriam Eggers provided excellent editorial suggestions and helped me express (in English) what I really wanted to. Remaining errors & omissions, of course, are entirely my own.

Last but not least, thanks to Manfred Heidbreder, a friend in need is a friend indeed.

This particular venture has not always been a bed of roses, I also had my share of rough experiences, moments of frustration, dejection and disappointments with different people, authorities and circumstances. Three people (without naming them), one who reproached me with "I can not deal with the subject of artificial intelligence unless I've read the German philosophers"; second, who underestimated me as "stupid" and finally, the third, who let all this happen in his presence, are worth the mention. I would like to thank them because their every denial was a step forward and motivation enough to achieve my goals. For me, they represent the three adverse human traits Arrogance, Ignorance & Incompetence, with which I do not want to do anything. Today, with these words, by putting the final nails in their virtual coffins, I would like to bury them for ever. On such moments of frustration my only source of encouragement and motivation were and still are the quotations provided in the text by some great human beings of our time.

All these years dealing with the German alien police department is also worth mentioning and wish no person with self respect will ever have to do anything with them.

My poor health in recent years was of very much concern not only to me and to my dear and near ones but to the whole medical team who took care of me on various occasions and still continue to do so. I am specially thankful to Dr. Hammerson (Neuro-Surgeon), Dr. Tuchelt (Endocrinologist), Dr. Disselhof (Cardiologist), Dr. Kempinski (Internist), Ms. Bäumer (Psychologist), Ms. Kobert (Physiotherapist) and their respective staff.

My family, my parents (Raj and Harbans), my god-parents (Shirin and Homi) and my friends have given me enormous inspiration and support throughout my career. My wife, Mini, and our young son, Abhinav, continue to help me to put everything into perspective. I am grateful to both of them for their patience, understanding and tolerating my not being available to them so often. Finally, I dedicate this work, with love, to my family, who gives me much more than I could ever provide in return.

Berlin, Feburary, 2001 **Sunil Thakar**

„wenn ich verzweifelt bin sage ich mir immer wieder daß in der Geschichte, der Weg der Liebe und Wahrheit immer gesiegt hat. Es mag Tyrannen und auch Mörder gegeben haben, die so schien es manchmal unbesiegbar waren aber irgendwann würden sie doch gestürzt“

Mahatma Gandhi
Source:Gandhi(motion picture)

All through history the way of truth and love has always won. There have been tyrants and murderers and for a time they can seem invincible but in the end they always fall,...think of it,.... always.

Mahatma Gandhi
Source: Gandhi(motion picture)

Table of Contents

Table of Figures

1 Introduction

To establish the relationship between the principles of Knowledge Representation and Reasoning Systems and their embodiment in working application systems, we undertook a case study in the domain of (automobile) sales advisory. Here, instead of inventing new techniques, we have tried to weave many conventional and experimental ones into a framework to design knowledge based consultation systems and also tried to demonstrate synergy among them.

This chapter introduces the problem domain, what we mean by Knowledge Representation & Reasoning systems and Knowledge Bases Consultation Systems (KR/KR&R and KBCS). It offers forward references to detailed discussions in later chapters.

1.1 The Problem Domain (Sales Advisory)

Traditionally, the process of sales advisory takes place orally and with the help of documents and manuals. The present situation in this field is that the salesperson does everything from requirement's analysis to product configuration; from present organizational and financial situation analysis to suggesting an appropriate solution plan manually. This task requires an enormous amount of knowledge and experience of a salesperson in various subfields ranging from product component and configuration knowledge to financial marketing etc.. Moreover, today's ever changing product development has made the products complex and financial market situations do not allow all salespeople to have the same degree of experience and knowledge about every subfield involved in the decision making, hence making the consultation task even more difficult. Computer support in this situation is thus getting more and more important. The purpose of our current work is not only to facilitate and accelerate the sales consultation process, i.e. to increase its efficiency but to improve upon the quality of the consultation as well. The quality of the sales consultation is defined by us in two ways:

1. large number of alternative solutions to be considered in the minimum time and
2. successful outcome of the consultation, i.e. how well does the sales object offered to the customer suit the customer's environment or specific needs.

The efficiency of the advisory process can be determined in terms of overall time required to provide all the necessary information to the customer to bring him to the point of decision.

The main aspect of the study was to determine why and how a specific KR system is best suitable for an existing specific real-world complex application (automobile sales advisory). Other aspects of the case study were:

- *easy maintenance of knowledge base (KB),*
- *efficient knowledge acquisition and*
- *consistency of the KB.*

The case study also considered such additional aspects:

- *uniformity,*
- *understandability (expressiveness),*
- *flexibility (of data access),*
- *stability and*
- *Reusability (changability).*

1.2 What we mean by KR&R:

Till today in the history of AI two fundamental and much talked about issues of importance are the acquisition and representation of knowledge. The notion of representing knowledge is at its heart an easy one to understand. It simply has to do with writing down, in some language or communicative medium, descriptions or pictures that correspond in some salient way to the world or a state of the world. In Artificial Intelligence (AI), we are concerned with writing down descriptions of the world in such a way that an intelligent machine can come to new conclusions about its environment by formally manipulating these descriptions.

1.3 What we mean by Consultancy/Consultation Systems:

Consultancy is not only acquiring answers for the future; it is also to get concrete solutions to problems which either do not exist or which are not the key to the future success of the specific business. The consultancy of the 21st century would be a distributed cooperative & communicative process involving a wider spectrum of people (possessing knowledge) employed in an organization; and who will have a broader understanding of business and of the inter-relations within it.

Moreover, today's ever changing product development has made the products complex and financial market situations do not allow all consultants (salespeople) to have the same degree of experience and knowledge about every subfield involved in the decision making, hence making the consultation task even more difficult. Computer support in this situation is thus getting more and more important. Thus any (knowledge based) computers system which supports (gives advice) the consultation process is termed as consultation/advisory system.

The proposed consultation/advisory system is based on several technologies such as deductive databases, knowledge processing and communication, multimedia & multilingual facilities. The proposed system is based on the client-server architecture and possesses adaptive features as well.

1.4 Socio-Cognitive Engineering Viewpoint

In this section, we provide the reader a short note about what we understand by socio-cognitive engineering viewpoint. The origins of this viewpoint lies in the observation that very few projects in AI and the domain of sales advisory have put equal emphasis on software, task, knowledge and organizational engineering. Some systems have demonstrated powerful techniques in knowledge representation, but virtually unusable. Others have attractive interfaces but do not address a clear need of the sales advisory domain. Yet others meet a well-identified need but do not fit easily into the workplace or the environment (sales organization). This approach to design human-technology systems is informed by studies of human cognition and social interactions. This approach of socio-cognitive engineering goes beyond „user centered design“ in that it involves analyzing how people perform complex tasks in organizational settings and then applying that analysis to development of systems that are not just usable, but useful and appropriate to the workplace. In another words, it draws

systematically (engineering discipline)[1] the cognitive, social and organizational sciences into the design process.

In this approach we see knowledge as a social resource but not as a commodity. It is seen here as more than a matter of technical innovation, neutral, objectified and separated from the social context Rather we see knowledge as the core resource for social interaction, a tool to support human cognitive tasks. Here, we are trying to design systems which complement human skills and recognize that such computer systems also structure relationships, not just information. Further, this viewpoint takes into account the various ways that actors and organizations are „connected together" with social relationships, as well as information flows and decisional authority.

In short, the term **socio-cognitive engineering** indicates the process of describing and analyzing the complex interactions between people and computer-based (supported, mediated or augmented) systems, so as to design the interactive systems which support the human cognition (learning, decision making, creativity etc.). This perspective extends the techno-centric notion of knowledge to the social dimension.

Most of the ideas considered here in „socio-cognitive engineering viewpoint" are based on the work being reported in different disciplines like software, task, knowledge, organizational engineering and some what newer disciplines like HCI (see [Becker & Buxton'87] for a historical survey & [Sharples'95] for an introduction), cognitive engineering and human centered systems design. For details on these and related topics we refer to the following: user-centered system design [Kling & Star'98], [Norman & Draper's'86]; soft systems methodology [Checkland and Scholes'90]; socio-technical and cooperative design [Catterall et al.'91], [Greeenbaum and Kyng'91], [Sachs'95]; socio-cognitive engineering approach & methodology [Sharples et al'97, Sharples et al'99]; cognitive engineering [Woods & Roth'88], [Rasmussen'86], [Norman'93]; engineering science of knowledge-based design[Tong'87] and the application of ethnography to system design [Rogers and Belloti'97].

The research reported here is based on integration of all these ideas, which constitute the two view points namely Socio-Cognitive and Engineering viewpoint, resulting into a single broader Socio-Cognitive Engineering viewpoint. The idea to integrate the two viewpoints was to avoid the mistakes made while considering each of these viewpoints individually. In other words researchers following one viewpoint were tending to forget the other; thus resulting into the same problem of acceptance of an expert system. For example, take an engineering expert system which helps a salesman - in a consultative situation - by giving him the exact technical configuration of the complex end product. In this situation, if we consider only the social context and neglect the engineering (problem solving process) aspect then the resulting system may have all the features which are required to integrate the system into the social environment (work place of an expert system user) but still may not be fully used/accepted because of its weak or not so appropriate technical solution to the problem posed .

In an another situation, if we have technically sound (provides technically optimal solutions) system and one has spent little or no time on the social aspects of its integration in the

[1] An engineering discipline lays out principles for designing, analyzing and building certain artifacts for particular purposes; in our context, the artifacts we are constructing are knowledge-based advisory systems and the purpose is to provide aid during the sales consultation/advisory.

working environment of its user, then system tends to face the same problem of acceptance because of its complexity.

To overcome this dilemma, I suggest an integrated viewpoint, (Socio-Cognitive Engineering Viewpoint) where neither of the above mentioned two views is emphasized/neglected at the cost of other. Here, what counts, is to have a system developed which shows strong problem solving behavior (engineering viewpoint) and simultaneously integrated completely in its users working environment (Socio-Cognitive viewpoint). The research is demonstrated in a knowledge-Based Consultation/Advisory System called KBCSsce. The system helps a salesman in a communicative situation (sales talk/consultation) by providing him/her the necessary information required to complete the sales (i.e. providing the customer with a product/product version which satisfies all his/her requirements).

1.5 Organization: (Forward references to detail discussions)

This work, by its nature covers the specific topic in some depth but also anchors it to the many related areas of research. I have attempted to organize the work, so that it may be read sequentially or by sampling particular topics. Chapter 1 introduces the problem domain, what we mean by knowledge representation and reasoning (KR&R), knowledge based consultation/advisory systems and how we understand the "socio cognitive engineering viewpoint". It offers forward references to detailed discussions in later chapters. Chapter 2 provides a narrated walk-through the main aspects of the case study. Chapter 3 & 4 deals with the introduction to KR&R and sales advisory. Section 3.3. presents description logics as a potential knowledge representation system, its application to KBCS and assesses its strengths and weaknesses as well. Section 4.2 presents the state of the art in consultation systems section 4.3 describes major topics covered by a human communicative situation (consultation). Chapter 5 discusses the representation of knowledge in KBCS and the details of the architecture. Chapter 6 surveys related areas of research in KR & knowledge-based consultation /advisory systems and adds an analytical evaluation of KBCS and discusses the implications of the results. Included are analyses of its sources of power, scope of applicability and major assumptions illustrated with examples. Chapter 7 offers a few general conclusions and conjectures about KBCS that have resulted from this research.

1.5.1 Case Study (main aspects)

This chapter deals with the main aspects of the case study. Section 2.1 describes the domain i.e. a consultative situation in buying a complex technical product. In another words, the domain of „sales advisory" in particular a system which helps a sales person during the sales consultations to sell an automotive product (Car, Truck etc.) of an automobile manufacturing company. The next section describes the existing solutions (XYZ & ABC). This section not only describes the system functions but analyzes the systems based on the following criteria: system development philosophy, knowledge organization/ structure/representation, knowledge acquisition, knowledge base consistency/maintenance/sustenance, problem solving methods/ inference mechanisms, human computer interaction and problems faced. Following that, we present the (re-knowledge-engineering) methodology adopted by us. Re-Knowledge Engineering is closely related to domain analysis and reverse knowledge engineering.

1.5.2 Knowledge Representation & Reasoning: Description Logics

This chapter presents an introduction into the field of knowledge representation & reasoning, which is also referred to as KR. Here, we not only present the fundamental ideas of this field but give a short overview of historical developments as well. Further, some important issues

in knowledge representations are dealt with. In the next section state of the art KR systems are presented, which lead to the general purpose hybrid logic-based KR systems. These systems are compared with other KR-systems and pros and contra of such systems are given. This section of chapter 3 closes with the deficits of KR systems.

Another section of this chapter deals not only with the deficits but suggests KR systems based on description logic as suitable candidates for our domain of sales consultation/advisory. Just about every current AI program has what is called a „knowledge base“ containing symbolic descriptions represented in some „representation scheme“. For implementing the knowledge, an operational description of rationality is needed and logic has always been considered an important aspect of rationality. Logic can be used to generate new knowledge based on given knowledge. Because of the importance of Logic this chapter provides an insight to the description logics and discusses the reasoning techniques. The exposition starts off with basics and it proceeds to a rather detailed exposition of technicalities.

1.5.3 Sales Advisory/Consultation

The problem is to help the sales person by providing him with an electronic assistant (computer) during the sales advisory task (i.e. while he/she is involved in a consultation dialog). The sales person's job is to understand the requirements/specification of the customer and then accordingly suggest a product, its alternatives and a comparative analysis of suggested products with its alternatives.

State of the art technology in the field of advisory/consultation offers lots of solutions, either by providing the sales person with an access to a product info base or simply a data base where all the product catalogues are stored. Now, maybe the product relevant information is digitized but the sales people seldom make use of such facilities or totally reject such aids. Why is that so? Why have researchers not developed an electronic aid/ assistant - for the sales person - which improves not only the quality of advice but also makes this process more efficient? Finally, why do the sales advisory systems developed today experience less success? In this chapter we try to provide answers to all these questions and based on these results we provide an architecture of knowledge-based consultation systems and further try to infer an architecture of generic knowledge-based consultation/advisory systems.

The next section provides us with the **state of the art consultation systems**. Here, we try to answer such questions as, why are the presently deployed sales and marketing management systems not sufficient to fulfill the present day’s demands? Why is computer aid unavoidable in today’s ever changing product development and financial market situations? Finally, why is there a need to develop the knowledge based sales advisory (consultation) systems?.

In this final section of this chapter 4, we try to define consultation in general as a communicative situation and its implications for Computer Aided Sales Advisory, in other words **aspects of consultation**. Furthermore, we also try to define what all - e.g. Agent’s, their functions their roles, task distribution, cooperation & communication and its type, and finally the medium of communication among them - is involved in the process of sales consultation. Thereafter, we will try to define a sensible division of labor between the human and the machine involved in this process of Sales Advisory/Consultation. The final three sections of this section provide the design criteria, general principles of the design and special requirements for representing consultation knowledge.

1.5.4 Domain Modeling

After presenting the basic concepts of the knowledge representation with the help of a terminological formalism (Description Logics) in the previous chapter, in this chapter we describe how the knowledge of a Coach-world was represented. First we would deal with some basic questions and then explain in detail the modeling of the domain. While modeling a knowledge area in the terminological system there arise lots of questions, which can only be answered in the context of the developing application. The next section deals with some such questions. To this count such principle decisions like, at what level we represent the information as objects and as concepts. Furthermore, different points concern the choice of appropriate language expressions and the granularity of representation, i.e. how „deep“ should be the modeling and the syntactic point of view.

1.5.5 Evaluation & Discussion (related work)

Once we have laid down the Framework for Knowledge base consultation systems, where the system is based upon two major hypotheses, one which says that success of any counseling/Advisory system pre-supposes a strong and descriptive product model, thus a descriptive knowledge representation and second it has to be integrated into its environment. In the penultimate section, we provide the specifications for such a consultation system . We compare and analyze the related work in these areas in order to discuss the differences, commonalties and their benefits.

1.5.6 Conclusion & Conjecture

In the final section of the paper we draw conclusions from the above mentioned discussion and present an analytical evaluation of a presented sales advisory system and discuss the implications of the results in the context of KBCS, including its sources of power, scope of applicability and major assumptions illustrated with examples. Finally, here we propose some research directions which may later on help improve the KR- and Consultation systems

The real voyage of discovery consists not in seeking new lands
but in seeing with new eyes.

Marcel Proust

2 Case Study

This chapter deals with the main aspects of the case study. Section 2.1 describes the domain i.e. a consultative situation in buying a complex technical product. In other words, the domain of „sales advisory“ in general & in particular a system which helps a sales person during the sales consultations to sell an automotive product (car, truck etc.) of an automobile manufacturing company. The next section describes the existing solutions (XYZ & ABC). This section not only describes the system functions but analyses the systems based on the following criteria: system development philosophy, knowledge organization/structure/ representation, knowledge acquisition, knowledge base consistency/maintenance/sustenance, problem solving methods/inference mechanisms, human computer interaction and problems faced. Following that, we present the (re-knowledge-engineering) methodology adopted by us. Re-Knowledge Engineering is closely related to domain analysis and reverse knowledge engineering.

2.1 Domain Description (Current Situation/Scenario):

A customer presently owns a small transport company and is close to exceeding the load/Volume capacity of his present fleet. He can increase his load/volume capacity by either:

1. Buying/Adding new trucks or
2. Leasing some more trucks or
3. A sensible combination of 1 and 2 or
4. Reorganizing his present activities + Buying/Leasing new trucks

Considering the above four situations; if he decides to buy new trucks then probably the cost of buying is more than the benefits and if he leases them all, probably in the long run its an expensive solution and a combination of the two is also not so optimal. Then what should he do? Choose the fourth option; here again he is not able to produce an optimal mix on his own. In this situation what he requires is an able consultant.

He walks into a local sales office of a leading truck manufacturer. There, he is received by a friendly sales representative who is ready to advise him on his current problem. Once, they have settled themselves down in a comfortable corner then the sales representative proceeds systematically in the following way:

Q. What type of company are you?
A. I own a Transport Fleet

Q. What is the capacity of your present fleet?
A. Trucks in no. = 4
Capacity Load = 500,000 Kg
Capacity Volume = 500,000 m^3

Q. What do you transport ?
A. Food Stuff
Q. What kind of Food Stuff ?
A. Perishable

Q. What food item in particular do you want to transport?
A. Fresh North Sea crabs

Q. From where to where, how often & what quantity ?
A. From Hamburg to Munich ; Everyday ; 5000 Kg

Q. What do you do with the crabs once in Munich?
A. Distribute them to retailers in Munich

Q. Any specials/extras to take care of while transporting ?
A. They should be kept under -70 C and should survive 20hrs

After going through this quick question/answer session the salesperson formulates the present job in the following words: 5000 Kg of fresh North Sea Crabs are to be transported everyday under -70 C of temperature from Hamburg to Munich and then delivered to the retailers in Munich. For this whole operation less than 20hrs of time is at disposal.

On the basis of the above information the sales person gathers a preliminary picture of the customers requirements and formulates them in technical terms like:

A new truck of type XYZ is required which has a minimum loading capacity of 5000 kgs and good refrigeration facility to keep these 5000 kgs of food stuff under -70 C for at least 20hrs and over 1000 Km of distance. In other words:

A truck with these specifications:

Capacity Load = 5000 kg

Capacity Volume = 5000 m^3

Refrigerating system capacity= 5000 m^3

Strong Engine to transport 5000 kg for 1000 km in less than 20hrs

Strong Dynamo to provide electricity for refrigeration

Large Fuel Tank to travel at least 1000 km in a day (by less no. of refueling).

Before the sales person starts mapping all these requirements to the actual product pallet of his company he tries to get some detailed information about the present situation of the company in order to suggest to his customer something which best fits his present business environment. The information gathering process is completed in a question/ answer session of the following type:

Q. What is the present fleet structure in terms of load & volume capacity ? (This may help him suggest a reorganization)

A. 1 Truck of 10000 kg & 10000 m^3
1 Truck of 5000 kg & 5000 m^3
2 Trucks of 2500 kg & 2500 m^3

Q. Who plans the daily tours (drivers or office personnel)?
A. Drivers

Q. How do they plan their tours ?
A. According to "nearest first" principle

Q. Can you please explain the work load distribution of your present fleet explicitly?

- Truck with 10000 Kg capacity shuttles between Amsterdam and Munich 3 times weekly and brings dairy products.
- One truck with 5000 kg capacity travels to Nuremberg every day.
- Another truck with 5000 kg capacity covers the area surrounding Munich travels app. 250 kms/day.
- Rest of the two trucks with 2500 capacity operate within Munich and travel average 100 kms/day each.

Q. How many drivers do you have ?
A. 4 one each for four trucks

Q. What do you do in case of absence ?
A. Hire one temporarily

Q. Do you get them easily ?
A. Sometimes yes, sometimes no

Q. What are the driver's activities during the tour ?
A. Deliver goods
Get the receipt
Book orders if any
Return, hand over all the receipts and new orders in the office

Q. How many staff are in your office ?
A. Two secretaries; one for order booking and the other for financing

Q. What is the mode of operation in your office?
A. Manual

Finally, the sales person asks the customer the following question in order to get the financial constraint:

Q. How much would you like to invest?
A. Less than DM 400,000 for a truck

Now, at this point the sales person starts the actual mapping of the customer's technical requirements to the actual product/components and offers him a basic type of truck which could be considered a possible solution to the above mentioned problem. Let us suppose he proposes

1. A truck type D5000 which has got the loading capacity of 5000kg

2. D5000 is actually designed for local travelling but the customers second requirement calls for it to do long distances. So that can be easily done by taking another type of motor called LDM700-D

3. The LDM-700-D types of motors require another type of Fuel Injecting System FIS-4000T which provides extra fuel efficiency and is economical as well but costs extra(these two additional benefits are used by the sales person later to justify the extra cost)

4. The next requirement calls for a good refrigerating system, for a D5000 Truck manufacturers provide a RS5T as standard but different versions like RS5T-AD, RS5T-XT, or RS7T, and RS7T's different versions RS7T-V1, RS7T-V2, RS7T-V3 are also available at different prices and/or benefits accordingly. Initially, sales person selects a standard RS5T type of refrigerating system but makes a mental note of other available alternatives for use at some later stage.

5. The RS5T being standard equipment requires also a standard electricity supplying unit ESU5000 but one can make use of ESU6000 and ESU7000 accordingly.

6. One last requirement is of a fuel tank which also comes in various capacities and sizes like FT300, FT400, FT500, FT700 and different extensions of these models are also available like FTXT100 and FTXT150 and only FT300 can be delivered as FT300-2 which is a double tank. As standard Factory fitting D5000 has a fuel tank FT300.

The initial technical configuration of a truck looks like this:

Type	D5000 – Original SDM-300 motor	300,000 DM - 30,000 DM
Motor	LDM-700-D	50,000 DM
Fuel Injecting System	FIS4000-T	7,000 DM
Refrigerating System	RS5T	25,000 DM
Electricity Supplying Unit	ESU5000	15,000 DM
Fuel Tank System	FT300	5,000 DM
Total Cost		**372,000 DM**

In the above mentioned configuration, if we change suppose the fuel tank to FT700 which requires a lot of space and costs more. It actually triggers two kind of constraints one the financial and other spatial. If we do not consider them then the resulting configuration looks somewhat like as given below:

Type:	D5000 - original SDM-300 motor	300,000 DM - 30,000 DM
Motor:	LDM-700-D	50,000 DM
Fuel Injecting System	FIS4000-T	7,000 DM
Refrigerating System	RS7T-V3	60,000 DM
Electricity Supplying Unit	ESU7000	35,000 DM
Fuel Tank System	FT700	12,000 DM
Total Cost		**414,000 DM**

Now if the financial constraint is applied, then RS7T-V3 can be replaced with RS7T-V1 which actually costs less but its bulk triggers the spatial constraint. To overcome this situation we replace FT700 with FT500. But his violates one of our prerequisites (long distance with less refueling). Ultimately, we find a compromise when we choose the combination: FT500 + FTXT100, RS5T-XT, ESU5000. Finally the optimal technical configuration may look like as follows:

Type	D5000 - original SDM-300 motor	300,000 DM - 30,000 DM
Motor	LDM-700-D	50,000 DM
Fuel Injecting System	FIS4000-T	7,000 DM
Refrigerating System	RS5T-XT	35,000 DM
Electricity Supplying Unit	ESU5000	15,000 DM
Fuel Tank System with extension	FT500 + FTXT100	7,000 DM +1,500 DM
Total Cost		**385,000 DM**

Once the technical configuration is over; the sales person considers business and organizational aspects of the requirements. Then presents first the simplest solution out of all the possible solutions. Which may look like this:

"For your present expansion plan you need only a minimum of 2 trucks of type D5000 with the above mentioned configuration. That makes the investment of 2 x 385,000 = 770,000 DM

Before suggesting some alternatives, he asks the customer:

Q. How do you plan to finance the additions to your fleet?
A. 40% own investment and 60% financing

Q. Have you got your own source of financing or would you be interested in "in-house" financing ?
A. I would be interested in "in-house" financing.

The sales person calculates the financial plan and hands it over to customer. PLAN1 doesn't seem to be lucrative enough. On realizing that, the sales person offers the customer with following options:

1. PLAN1 with some discount

2. Instead of buying two trucks he should buy 1 and lease 1 (shows how leasing can be cost effective)

3. Lease both trucks and reorganize his company by introducing computers and buying the necessary hardware & software from the manufacturers.

Once the customer decides on what exactly he wants, a formal deal can be struck

The scenario mentioned above tries to sketch one typical present day consultation situation. A close look at this scenario description shows the complexity involved in delicate decision

making situations where no "best" answer exists when developing appropriate buying/acquisition plans that satisfy customers requirements while meeting applicable financial and organizational objectives. Here, the question is not simply to optimize some specific objective but any proposed solution will be the result of balancing competing goals. Furthermore, suggesting merely a plan would be insufficient; rather explanations are required to persuade and convince the decision maker that the proposed solutions are reasonable.

The present situation in this field is that the sales person does everything from requirements analysis to product configuration; from present organizational and financial situation analysis to suggesting an appropriate solution plan etc. manually. This task requires an enormous amount of knowledge and experience of a sales person in various subfields ranging from product component/configuration knowledge to financial marketing etc. Today's ever changing product development and financial market situations do not allow all sales persons to have the same degree of experience and knowledge about every subfield involved in the decision making. Hence, making the consultation task even more difficult. Computer support in this situation can be of great help.

2.2 The Two Representative Systems

The next two sections describe the two representative expert systems XYZ & ABC. In our view XYZ represents the technical-oriented view and helps the sales person to sell a truck, whereas ABC represents the social-oriented view and helps a car sales person during the sales talk with a customer.

These two sections not only describe the system functions but analyze both the systems based on the following criteria: System Development Philosophy, Knowledge Organization/Structure Representation, Knowledge Acquisition, Knowledgebase Consistency/Maintenance/ Sustenance, Problem Solving Methods/Inference Mechanism, Human Computer Interaction, Problems faced, as well.

2.2.1 XYZ: The Sales Support System (Truck Configurator)

At the corporate headquartersof a leading truck manufacturer, in the EDP-department, since early 80's a computer system was developed to automatically check and verify the sales order forms for a particular product. (The name of the computer system and the product has been classified; thus I shall not be using names). The computer system performs two jobs:

1. Basic Check: checks whether the product components and ordered product features are allowed/valid, and
2. Multi-Dimensional Check: checks whether the ordered components are configurable with each other.

In other words, not allowed combinations in an order form are deleted or signaled with a warning and necessary but not specified component combinations are either automatically inserted or warned. In short the functionality of the system is comparable to the classical example of the configuration system XSEL [Mcdermott 82a] & R1/XCON [Mcdermott 82b].

In the beginning, experts used to specify the rules daily on a form (piece of paper) which were then passed over to the EDP-department. First these rules were fed in manually and then during the night a Batch-Procedure would translate these rules into a Cobol-Check Module. This module would then be linked to the Cobol-Check Program in order to execute the whole

thing. Since 1986 engineers have been working on a new version (called XYZ) of this system where if-then-else Cobol-Rules are substituted by logical (dependency) chains. Furthermore, new flexible data-input was designed to handle extra logical queries such as: production-date, color etc. Since 1990 this version is operational to its full extent.

This system is completely integrated into the sales transactions. Along Order-Form generation and verification one can determine the delivery date, calculate the price and Order Confirmation & Order Slips. Further interfaces are provided to production planning and material disposition.

2.2.1.1 Facts & Figures:

- At present, XYZ checks and configures 900 new order-forms and 1800 order-changes
- XYZ's knowledge base contains 10000 complex rules which are further divided into groups independent of each other.
- XYZ uses 2000 product model related rules per order-form check/verification.
- System XYZ run's on a centrally located IBM3090 under the operating system MVS
- XYZ is written in Cobol
- The processing of order-forms takes place during the night as batch-run
- Through an online interface a direct access to the system XYZ is possible for all the 39 inland and 7 foreign sales branches.
- 18 engineers maintain and sustain the system XYZ
- Every year approx. 30% of all the rules are either changed or written new
- On an IBM3090, XYZ needs 1,2 CPU-secs to check an order-form.
- Total cost of computing on this system is ca. 1.62 m DM/year.

[Hinzman et al. 90]

2.2.1.2 Existing Solutions

In the past 10 years system XYZ has been an object of several case studies and optimization experiments. In every investigation system XYZ was examined for two criteria:

a) whether through new concepts the system can be optimized?

b) whether the systems computing time/costs can be reduced.

Till date, no attempt has ever been made to improve upon the structure of the knowledge encoded in the system. Reason being, that system XYZ, in its present state, is in position to process 95% of the order-forms correctly, even though it has to run through the Cobol rule-base 7 times sequentially by oscillating between delete and insert-rules and causing an increased CPU time.

One possible critic to the concept of XYZ was the pre-defined inference strategy of 7 fixed sequential runs through the Cobol rule-base, which theoretically does not always reach a correct solution. Apart from that this strategy leads to a sub-optimal CPU-Resource utilization.

In the following section we will describe the results of two prior trials (COBOL & KBMS solutions) to improve upon the performance of System XYZ based on the above mentioned two criteria.

COBOL solution:

in this first attempt (1986-1990), the complicated if-then-else Cobol rules were replaced by logical (dependency) chains. XYZ now consists of a rule-based knowledge base which is based on propositional logic. It contains 10,000 such rules fully independent of each other.

- *Knowledge Representation:* all the rules in XYZ have a conditional part and have one or more action parts. Some restrictive conditions may have been further assigned to an action part. The action part will only then be executed if along with the argument of the condition part these restrictive conditions are also fulfilled. Rules with multiple actions are built in two steps. The top step (called ARG) contains the condition which is valid for all the following actions. The lower step (labeled as 001 & 002) contains all the conditions which along with the ARG are only valid for this particular action.(see Fig. 2.1)

```
ARG      BM = 617003.

         UN    CO = X12#
001      CO = X12
   U     LD > 399#
002      CO = X13
   U     LD < 400
   UN    CO = A06 A40 A41#
```

means:
IF in the order-form the first 6 digits in an 8 digit production pattern contains 617003 and the order-item X12 is not included
THEN insert accessory X12 in the order-form only if the ordering country has Identifier greater than 399
OTHERWISE insert X13 in the order-form as long as the ordering-country's identifier is less than 400 and none of the accessories are already present in the order-form.

Fig. 2.1: A Rule in XYZ and its Interpretation

- *Control Mechanism:* Due to maintenance reasons, rules are developed independently of each other and stored in a flat knowledge base (without hierarchies or any other kind of structure). All the rules are processed sequentially. In order to make this processing independent of the order of rules; the complete knowledge base has to be processed several times. The present version of XYZ requires 7 inference cycles(runs) to get the desired results. By each inference cycle we mean a complete sequential run through a block of either „delete rules“ or „insert rules“ respectively, where every possible (applicable) rule in these blocks will be fired. XYZ's present inference strategy is as follows:

  - 1st Run/Cycle: Run through the delete rules
  - 2nd Run/Cycle: Run through the insert rules
  - 3rd Run/Cycle: Run through the insert rules
  - 4th Run/Cycle: Run through the delete rules
  - 5th Run/Cycle: Run through the insert rules
  - 6th Run/Cycle: Run through the insert rules
  - 7th Run/Cycle: Run through the delete rules

- *User Interface:* the user interface for the experts is completely independent of the system's internal representation and that's why there exist two data bases. The user's data base

contains rules in a form as shown to experts on a display unit (fig. 2.2). The data base for system internal representation contains practically the same information but is optimized for inference purposes. New or changed rules can immediately be included in the user's data base and furthermore through a parser can be converted into the syntax (internal structure) of the system's data base.

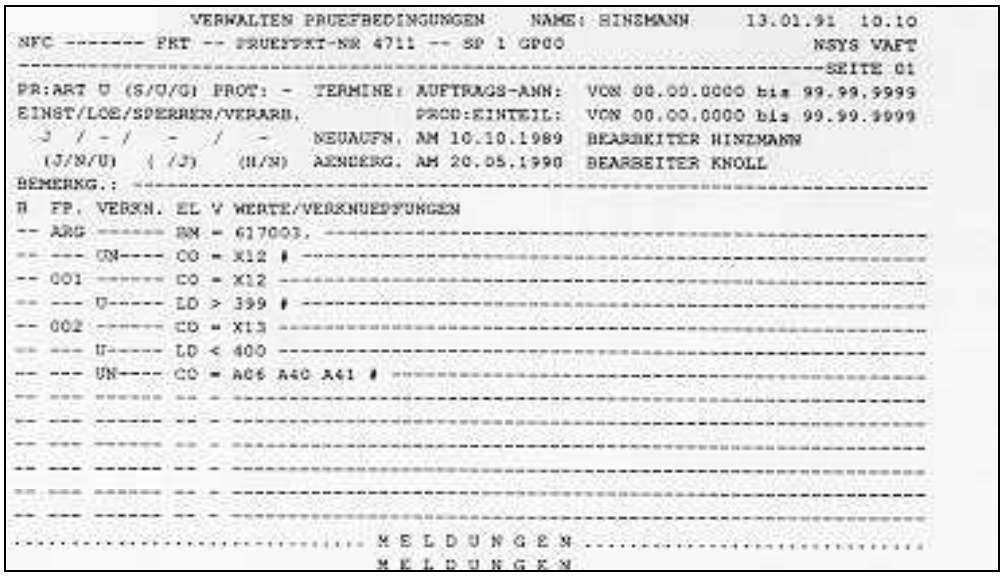


Fig. 2.2: Previous example shown in a user interface of XYZ

- *Inference Optimization:* the optimization of the inference mechanism is achieved through the high rate/number of inferences. To achieve fast inferences all the rules are saved in a form of logical pattern. For rule evaluation truth values of the rule premises are calculated. By an AND connection among the rule elements, the truth values are multiplied and by OR connection added. To resolve the conflict between possible dependencies among individual rules the knowledge base is run through many number of times. During which it is checked whether there has been a change since the last run and if this is not the case then the inference mechanism terminates. To reduce the search space the whole rule base is subdivided into modules which are fully disjoint amongst each other.

  The central division criterion is the type of production pattern. One such search cycle considers all the rules from the knowledge base for one particular order-form just on the basis of the first three digits of the production pattern.

- *Explanation & Documentation Components:* the changes in the order-form which are caused by the knowledge base are automatically logged. Ordered components before and after verification/configuration are listed in the log file. All the changes are labeled according to which rule the components have been inserted in or deleted from the order-form.

KBMS solution:

in this case study the goal once again was to find out :1) How good the XPS development tool KBMS is for such a large application with respect to development and maintenance and 2) How is its performance with respect to a large number of rules. In this case study, from the complete Cobol-solution only modules which insert and delete the component codes in the order-form were reimplemented in KBMS.

First, a conceptual model was developed which maps all the functionality of XYZ's Cobol-solution to KBMS. As KBMS is a rule-based XPS development tool, once again a rule-based paradigm was chosen. Due to the large number of rules it was difficult to rewrite all the rules manually. Therefore, a conversion concept and program was developed to convert all the Cobol-rules - in a syntactically & logically right form - to KBMS-rules syntax. Due to certain limitations of KBMS rule syntax on average 1 Cobol-rule was converted as 3.3 KBMS rules, increasing the total number of rules to the same factor (i.e. 10,000 * 3.3 = 33,000 rules).

- *Knowledge Representation:* the KBMS-model contains simple objects with an object-attribute-value structure. Each object contains one or more attributes and attributes can have either single valid values or a range of values. Some of the following objects were also defined in KBMS-solution: the Order-form was divided into two objects. First object ORDER contains all the features as its attribute which occur only once like Ordered-Production-Type (alphanumeric, 8-digits), Ordering-Country (numeric, 3-digits) and the Production-date(date format). The second object (CODES) contains all the features as attributes which have multiple occurrence e.g. Ordered Component called CODE (alphanumeric, 3-digits), and Status of Ordered Component called STATUS. STATUS has „always-available" as default value and „inserted" & „deleted" are allowed also as values. Apart from that this object contains attributes RULE and FOLLOWING-CONDITION to protocol the rule and condition in case the system has changed a certain ordered-component automatically in the ordered-form. There is no relational link between the two objects ORDER & CODES. As per definition they always belong to same order-form and thus need not use any sort of inheritance mechanism.

  The other objects like VARIABLES and HELPFIELDS were also created to have attributes which are needed to measure time and to manipulate component.

  Rules in KBMS are typical production rules and have the following syntax (Fig. 2.3).

Fig. 2.3: Rule Syntax of KBMS

- *Modularity of KBMS-Model:* principally it is possible to have all the rules in one knowledge base and to process them there. However, that could lead to a very long response time as a large number of rules have to be checked. Through a modularity concept the number of evaluating rules can be reduced as long as one can find some differentiation criteria. One possibility would be to find parts which are independent (logically disjunct) of each other in their contents or parts which could be separated as logical super or subsets of any rule set. In this KBMS application like its Cobol version the same criterion „Production-Pattern-Type" was used. In a module or a packet all the relevant rules for one particular „Production-Pattern-Type" are grouped together. This 3-digit „Production-Pattern-Type" is used to refer to different rule packets. They could be tested through the AGENDA CONDITION of the Rule Packet. Foe example rules which have their highest common Production-Pattern-Type beginning less than 3 digits (i.e. 99 or less) must be stored redundantly in different packets.
- *Inference Strategy:* as an inference strategy classical Forward Chaining was used which has proved in many configuration tasks to be efficient and appropriate.

2.2.2 Domain Problems

Like any other rule based system XYZ also has problems. XYZ has a very large number of rules and the frequency of change in the rule sets (30% per year or 10/15 rules every day) creates lots of problems in maintaining and sustaining the system. Moreover, 18 knowledge engineers, who at present maintain XYZ in a physically decentralized organization is in itself a problem not only during the knowledge acquisition but also during the representation of knowledge.

In the following section, in short, we will sketch out the problems faced by XYZ in its structure and representation of knowledge (Rules), which causes further difficulties in knowledge acquisition, representation and maintaining the knowledge base consistent at every stage.

2.2.2.1 Present Knowledge Structure

At present XYZ's knowledge base is divided into two parts. The first part contains all the rules which delete all not acceptable components from the order form (Negative Rules) and the second part contains all the rules which insert all the valid components into the order form (Positive Rules). The rules in both cases are not sorted because of the advantage that new or changed rules can just be added at the end of respective rule-packets and there is no need to check the effects of these new additions or changes. These rule packets are then run through several times for inference and to resolve eventually all the forthcoming dependencies (Fig. 2.4).

In a file with positive rules; two following rules are saved:

Rule 1 IF Code A is present in the order-form Then insert Code B
Rule 2 IF Code B is present in the order-form Then insert Code C

Now, in the order-form only Code A is present. Then by the sequential run through the rule set first Code B is inserted by Rule 1 and then by Rule 2 Code C is inserted. Thus after this sequential processing of rules an order-form will contain Code A, B and C as well.

As described, all the rules can have any sequence. Thus it is possible that both rules can be written in an interchanged sequence:

Rule 1 IF Code B is present in the order-form Then insert Code C
Rule 2 IF Code A is present in the order-form Then insert Code B
If in an order-form once again only Code A is present, then the first rule does not insert any component because Code B is not present and only the second rule then inserts Code B. After the first sequential run the order-form contains only Code A and B. Thus only after the second sequential run with Code A and B in the working-memory one would be able to come up with the right results.

Fig. 2.4: Example for inference via 2 sequential runs through the rule set

Dependencies as shown in the above example can occur in any depth. Thus, theoretically by n unsorted rules maximum n sequential runs are necessary but in the practice logical dependencies are small. One possibility to evaluate this type of rule system would be a multiple constant number of sequential runs. The disadvantage would be that theoretically there is no proof of its correctness as inference - even after the final run - could still be incomplete. On the other hand by a small number of nesting eventually redundant sequential runs could take place.

Another possibility to terminate this evaluation process would be that if two consecutive sequential runs do not change the contents of the working-memory. In this case the number of inference cycles would always be one more than necessary. If the process terminates then for sure the inference is complete. If not and some conflicts occur then theoretically this inference process can go on for ever. The process can then be terminated if „number of inference cycles“ is greater than the „number of rules“ in the rule base.

The third possibility would be not to determine the set of all executable rules by sequential runs through the rule base but to produce them beforehand through a special table supporting branching of dependencies of all the conditional parts of the rules and with which - during the run time - the set of all the currently executable rules can be determined quickly. In this process similarities in the conditional parts are exploited in order to build the rule groups which could then be processed together. Thus the set of all the executable rules is build in a first run and in following steps only their logical implications on this set is processed (Agenda-Mechanism). This kind of a process, for example, is used in Rete Algorithm. The only disadvantage in this case is that it is computationally very intensive and needs a very large memory.

The resolving of dependencies between the rules under certain conditions cannot be achieved dynamically during the run-time, but should be made possible in advance through a static sorting process. Rules are brought into a static order which enables - for every possible start situation only one sequential run through the rule base - a complete inference. The advantage of a static sorting process could be an optimum inference during the run-time. This process is only possible if a rule base meets the conditions which are described in the next section.

2.2.2.2 Conditions to conduct a static sorting and their problems

The condition is that the order of the rules in a rule base should not have any effect on the results as long as all the conditional parts are logically valid and are used any time during the inference process. A simple way of achieving this is the method of multiple sequential runs through the rule base as described before.

If we want to achieve this only in one sequential run through the unsorted rule base then the above mentioned condition does not hold anymore because it is never sure that all the conditional parts which are logically valid are actually going to be used sometime during the inference process. Through sorting the rules according to the logical dependencies between

the conditional part of one rule and the action part of another, the deduction can be made sure of. But there should not arise any conflict in the sorting sequence. For example the sorting sequence may contain conflicts if a rule due to different dependencies is placed once before and at the same time after a specific rule.

Following problems were detected during the sorting process:

Loops:

A loop in a directed graph means a backward dependent reference of a rule premises to the conditional part of another rule. This is necessary to deduce the rule premise.

Example:
The following two rules are given
1. IF Code A is present Then insert Code B
2. IF Code B is present Then insert Code A

In the above example we see a direct one step loop (equivalent). In a sorting process this leads to a conflict as both the rules are dependent on each other and in an action part of one rule a component is inserted which builds the conditional part of another. Despite conflict during the inference process either via single sequential run or via multiple sequential runs the same result is reached, i.e. if A or B are present then insert the missing one. In XYZ knowledge-base a lot of such equivalent rules appear.

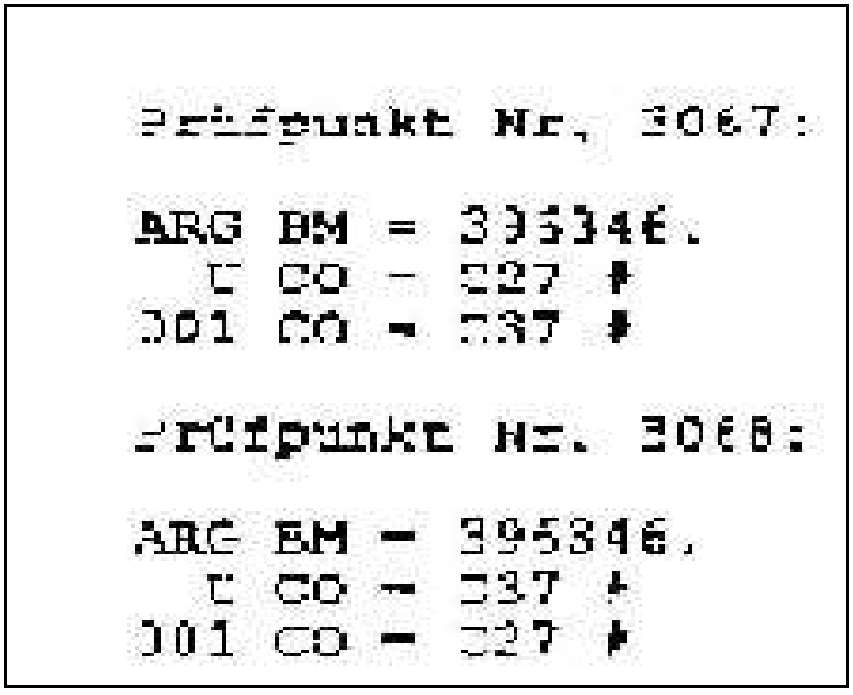


Fig. 2.5: Single step loop in XYZ rule-base

In contrast to single step loops, multi-step loops create problems e.g.

Example:

1. IF Code A is present Then insert Code B

2. IF Code B is present Then insert Code C
3. IF Code C is present Then insert Code A

such multi-step loops lead as well to a conflict during a sorting process and do not come always to the same result as compared to single-step loops via either a single sequential run or a multiple sequential runs. In the above example if only Code C is present then via single sequential run only Code A is being deduced and via multiple runs Code B & C as well. In order to achieve this in single-step, the conclusion parts of all the rules will have to be extended for all the alternatives. The following type of multi-step loops were not detected in the investigated version of XYZ (Fig. 2.6).

Example:
1. IF Code A is present Then insert Code B & Code C
2. IF Code B is present Then insert Code A & Code C
3. IF Code C is present Then insert Code A & Code B

Fig. 2.6: An example of a multi-step loop

Sequence Check

In the XYZ-implementation one of the specialties of the rule syntax is that a lot of sub rules can be put together in a single rule. During a sorting process this leads to a problem of cross-references (fig. 2.7), which says that one sub-rule is to be sorted out at a different place to the other one and leads to conflict.

The following two rules are given and each rule contains two sub-rules
Rule 1:
IF Code A is present Then insert Code B
IF Code E is present Then insert Code F

Rule 2:
IF Code B is present Then insert Code C
IF Code D is present Then insert Code E

Fig. 2.7: Cross-references in a rule with multiple propositions

If one analyses the above two rule then one can detect the following dependencies:

A => B => C
D => E => F

Thus one can conclude that it is not possible to determine the exact sequence of the above two rules because in the first chain Rule 2 is dependent on Rule 1 and in the second chain Rule 1 is dependent on Rule 2. This conflict can be resolved if all the sub- rules are converted into individual rules.

In an actual XYZ Knowledge-Base one finds such cross reference as shown in the example below:

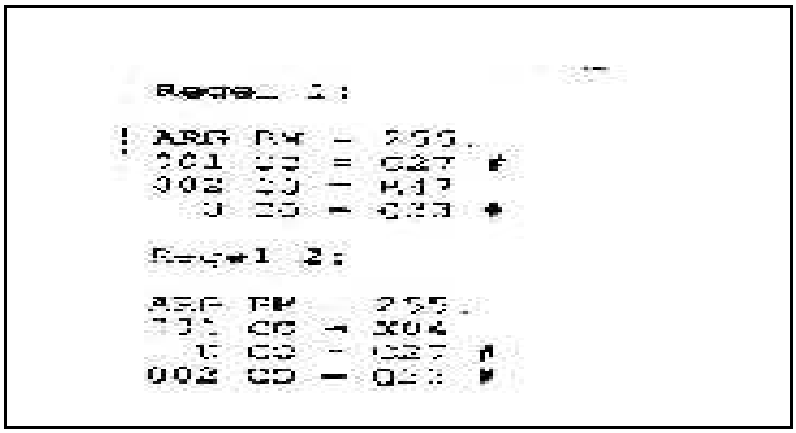


Fig. 2.8: Example of cross-references in actual XYZ Rules

Variables:

By introducing variables, the propositional logic can be extended in the direction of predicate logic. But it would be very difficult to find that there are some specific variable allocations which change the logical value range of a preposition. If the variables are instantiated with the concrete values, as done in the dynamic rule processing during the run time, then it is easy to determine the propositional value. But in a static sorting one must in advance check which intersections of the propositional-logic value-ranges of rules have a relation with other rules and whether through that some conflicts arise or can be avoided, respectively. In the XYZ Knowledge-Base, only Production-Pattern and Ordering-Country are introduced as variables, thus the value-ranges are to be determined. The investigation of value-range of variables in XYZ can be used to constrain the logical dependencies between rules so that conflicts can be avoided during the sorting process.

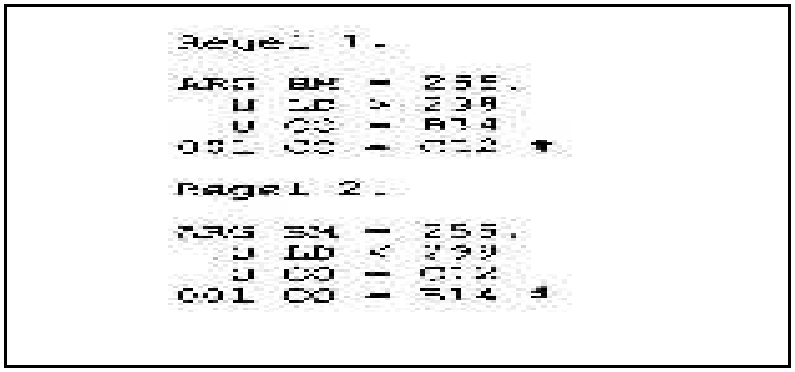


Fig. 2.9: In the above rules, by investigating the value-range of variable Ordering-Country, one can make sure that both the rules are independent of each other

Contradictory Premises:

Contradictory premises are defined as rules which have a contradictory conclusion part yet simultaneously their premises can be accomplished. Such contradictory premises, in a sorting process, do not lead to conflict situations. For example:

Example:
Following two rules are given
1 IF Code B is present Then insert Code A
2 IF Code C is present Then delete Code A

Fig. 2.10: Rules with contradictory premises which do not lead to conflicts

In order-form, Code B as well as Code C are present.

In the above example, even though a straightforward contradictory set of premises is given, during the sorting process it would never be identified as blocks of insert-rules and delete-rules are sorted separately without investigating the dependencies among them. Even after a single sequential run through both the rule-packets it cannot be detected. Thus, with the help of an action list one must make sure that insertion then deletion (or vice versa) of the same code leads to an error notification. If according to the present strategy of multiple sequential runs the inference is not terminated then it indicates a conflict.

2.2.3 Intermediary Conclusion:

After a careful analysis of the solutions presented in two different case studies the system XYZ still faces lot of problems. A completely different approach to tackle all these problems would be to bring in the structure to present XYZ's knowledge. Thus it would be advisable that during the design phase one should structure and represent the knowledge in such a way that during the maintenance and sustenance of the knowledge base there is no change in the structure but only in the objects and rules changes are required. Moreover, this will help in acquiring knowledge as well. In the following section we would present a new knowledge structuring and representation scheme keeping all the above mentioned criteria in view.

2.3 ABC: The Automobile Selling System (Car Consultant)

Description of an XPS (ABC) based purely on Social View: Before we describe the system called ABC, we would like to make an introductry remark

Introductory Remarks: We begin this section with a series of quotes ....."At the focal point of the conception - is the complete/full integration of the communication process of a sales person with his/her customers" ..."With the support of ABC the sales person concentrates once again purely on sales-strategic aspects and the technical restrictions are automatically taken care of by the system ABC"..."To achieve this, fundaments and methods of AI, knowledge sociology and cognitive psychology are considered/used".

Sales Situation: Presently a sales person has lots of documents -in paper form- (e.g. car-sales handbook, car-accessory brochure, car-price lists and competition supportive strategic statements) at his/her disposal. From these documents during the sales talks he/she is supposed to make an offer to the customer. While he/she is doing so the salesperson must consider not only the sales tactical aspects but the technical restrictions as well.

That means, for a large range of sales offers the sales person requires more time and the complete range of offers is not always clearly visible to the customer. Furthermore, price calculations have to be done separately with the help of an ordinary calculator. In case of change of product models and accessories one has to do all the price calculations once again and complex technical tables are also to be considered. Immediately after the sales talks neither for the customer nor for the salesperson is there any clean document which confirms

the customers order. At present the sales branch office secretary types per hand and at a later stage the customer receives the printed order confirmation.

During the above mentioned administrative work, the sales person deviates from his sales activities like negotiations, bargaining etc. All this leads to providing support to a sales person while he/she is counseling the customer.

2.3.1 Functional specifications for ABC:

A sales supporting consultation system should relieve a sales person from the following tasks or at least support him/her actively:

Basic Tasks:

- Complete offer for a customer
- includes type/model selection, accessories selection, price calculations and a printout.
- Automatic consideration of all technical dependencies
- includes technical restrictions, prices and incompatibility
- Separation of models alone with the accessories according to their contents and optically as well.
- By change of model, automatic adjustment of factory-fitted and customizable accessories accordingly.
- Information about the available factory-fitted and customizable accessories.
- Continuous price calculation and view of selected options
- Inclusion of special arrangements.

Further, for the comfortable usage of the system certain extra functions are required e.g.
Service Tasks:

- Creation and processing of order confirmation
- Saving the offers(according to customers) archiving system
- Connection to different systems within the sales transactions

The prerequisite for the acceptance of such a system is an actual database and maintenance of such systems should also be possible by the sales personnel and without any special knowledge of EDP. For this the following modules are required:

Central Tasks[1]:

- Changes in Prices (of accessories and models)
- Inclusion and exclusion of accessories and models
- Maintenance of technical restrictions between all components
- Realization of factory-fitted accessories
- Creation of product information

2.3.2 System Development Philosophy:

once we have the functional specifications of ABC, here at this point we would like to present the underlying goals and systems development philosophy.

[1] **Note:-** All these programs for data maintenance - under Central Tasks - are not part of a system which can be accessed by a sales person. They are exclusively centrally accessible but must fulfil the same functional specification for comfort and user friendliness as the actual consultation system.

Goals: the primary goal of the research work presented here is to contribute in the field new forms of distributions of cognitive tasks between human and computer. In other words the optimum interaction between human and technology; especially within an organization. This work is about the techniques of representation, processing and distribution of knowledge. Yet, in another words, it is about the creation of a new type of intelligent medium for communication of knowledge to support planning and consultation processes within an organization.

An example of such a work process is the task of a sales person. Here, today not only the technical restrictions of production and manufacturing but also the sales strategic conditions are to be considered by the sales person during the sales talk. Decisive - in the task of a sales person - is to communicate with the customer under skillful sales negotiations, which at present suffer because of the technical restrictions of the product. Thus shifts the focal point towards achieving the correct manufacturable product and having the consequence that the whole scope of variations offered by the modern production methods cannot be fully exploited.

The goal of a sales supporting consultation system is to provide a car sales person with a PC based instrument to ease his job while selling. Here, top priority is the functional embeddedness of the system in the work process of the sales person. This leads to the following specifications for the system:

- Easy operation of the system
- Lucid/clearer partition of the display/screen
- Possibly complete information about contents, catalogues etc.
- Immediate orientations, possibility about recent activities
- Total mobility of the system(Laptop)
- Complete freedom of sales person's work (with the PC)
- No system stipulated interruptions during the sales dialogue

Such an instrument should relieve the sales person from present complicated technical and administrative routine jobs.

2.3.3 Implementation Philosophy

primary goal during the Implementation of the prototype was to achieve operative fairness to the sales person. This includes not only the layout of a visible user-interface on the screen/display but also the reaction of the system to the different operational situations during the sales dialogue. The communication between the sales person and customer during the sales dialogue should at no occasion be disturbed by the demands of the system towards its user.

All this reflects back on the detailed construction of the operations and the reaction of the system:

- The selection of alternatives should occur with fewer and easily placed keys on the keyboard

- The localization of the selected alternatives on the screen should easily be possible with the naked eye and without any help e.g. from the index finger etc.

- The screen should have a continuous, smooth and clear design

- The functional associations of contents and statements should always take place in the same area of the screen

- The expression form used to represent the objects and alternatives (catalogues, models, features, instructions etc.) should always be done in the language used by the sales person.

- The system should on no occasion order the user to a forced step

- The user should be able to carry out any action at any time

- The system reacts tolerantly towards the false/erroneous operations (i.e. it starts processing again without interruption immediately before a false input occurs)

- The system recognizes the erroneous technical configurations but prompts the user until all the components have been put together (Thus the sales person has a chance to revise the erroneous inputs given due to certain sales strategies).

The selection of operational continuity wished for by the user is not controlled by the function keys in ABC. This has been purposely avoided in order to achieve a direct control over the contents displayed on the screen. Thus a user need not have to think in two dimensions:

- Selection of function based upon the contents
- Controlling the system's progress

The function keys are used only for help functions (e.g. help texts, displaying extra information about selected objects etc.) which do not influence the system's actual processing.

The system is developed in such a way that it activates the moments of playing among its users (salesperson and customer). That means that a salesperson is animated by the possibilities of the system to vary the models and accessories and the customer plays all the alternatives through, which were till now not considered, as the system provides correct accessories and a price.

This opens - for the sales person - a new scope for negotiation which has till now been used seldom by him/her because of lengthy and complicated calculations.

2.3.3.1 Knowledge Representation/Organization/Structure:

in the very first feasibility study the initial design was realized on a LISP-machine (Symbolics) and with the help of LISP(Fig. 2.11). The approach, adopted here, to represent the necessary knowledge was based upon a simple input-output model, where necessary inputs were given through the extensive usage of built-in menu functions and the constraints used to configure all the accessories of a car were realized through the conditional statements in LISP.

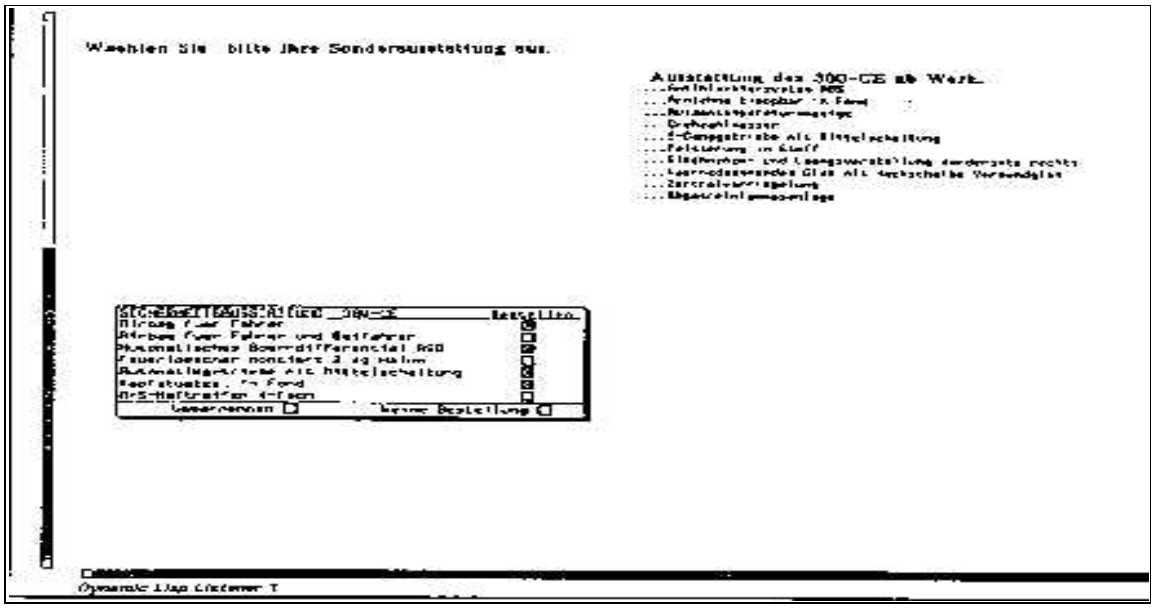


Fig. 2.11: LISP version

Due to unsatisfying user-interface development possibilities, another version of ABC was developed with the help of Knowledge Craft (Fig. 2.12). Here, the data (e.g. basic models, accessories and price) was encoded into CRL-Objects and the restrictions were represented as production rules in OPS5. The interaction among the objects was realized through the Demons (KC-facility).

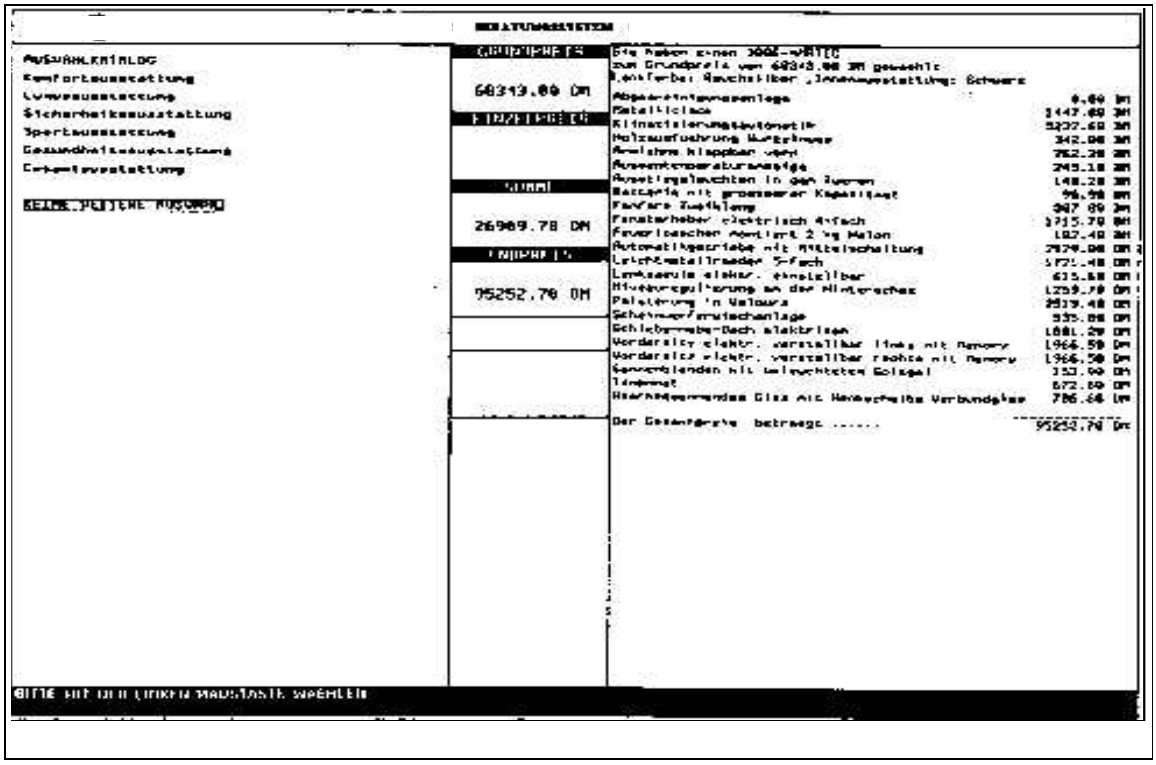


Fig. 2.12: KC-Version

After experimenting with KC-version, ABC was reimplemented on a PC (Fig 2.13).

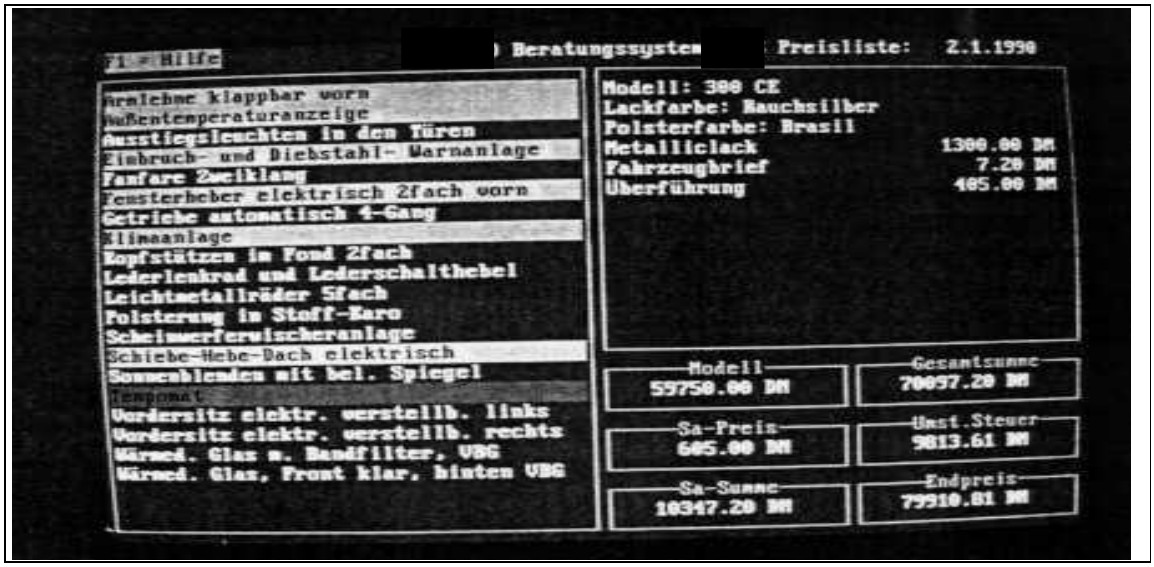


Fig. 2.13: PC-Version

Here, TURBO-PROLOG (now PDC-PROLOG) was used as an implementation language. The program logic (design) and the partition among the rules and data were adopted without any change from the first KC-version. Here, Rules and Data both were represented as PROLOG-facts. As arguments to the facts the slot values from the KC-version were used. The relations among the facts could be built through the pattern matching. In order to achieve necessary run-time optimization and flexibility, a rule interpreter was developed for processing the manufacturing constraints. Five types (Fig. 2.14) of rules with internal “and” and “or” connections are allowed and following are the valid types: only in connection with, not in connection with, only one-of, includes and substitute. All these types are differently processed.

```
p("()","u2",["325"],["310"],[],"")
p("%","ex1",["330","331"],[],[],"")
cp("ersetze","e02",[],[["531"]],[],["538"],"")
cp(">","v0",[],[["550"],["651","650","653","951","950","952","953"]],[["480","489"]],[],"")
cp("?>","ev30",[],[["AusSL"]],[["442"],["471"],["551"],["580"],["880"],["283"]],[],"")
cp("!","w1",[],[["300"]],[["310","325","325+326","323","323+326","355"]],[],"")
c("$","c1-Lsp",[261],[["750","751","537","538"]],[262],"")
sa("AA-Datum","",10000," 3. 6.1991",0,0,0)
sa("Abgasreinigungsanlage","",203,"620",985,0,0)
ob("",0,[],[],[],[],"","")
ob("190D",32300,[],[20,79],[],[6,.... ,596,597,1000,1011,1003,1101],"Limousine\n5-Sitzer,
4 T#ren\n4 Zylinder-Dieselmotor, 1997 ccm\n55 kW/75 PS\n18565 R 15 87 S\nRadstand:
2665 mm\n","")
kataloge("Kompaktklasse",["190 D","190 D 2.5","190 D 2.5 TURBO","  ","190 E
1.8","190 E 2.0","190 E 2.3","190 E 2.6","  ","190 E 2.5-16"],4)
kataloge("... nach Code suchen",["suchen"],4)
kataloge("Sonstiges",[" ","   inz","   ","  ","260"],4)
kataloge("Weiter",[],4)
```

Fig. 2.14: Knowledge representation and organization in PDC-PROLOG version of ABC

2.3.3.2 Problem Solving Methods/Inference Mechanism:

In the KC-Version, where rules are encoded in OPS5, the obvious inference mechanism or problem solving method used is to forward chain the rules and make use of the efficient RETE algorithm [Forgy '82] for the processing.

In the PROLOG version, where all the rules are divided into the above mentioned five different types, a customized approach was adopted to process these rules. The first two types were used to display a special text and in case there was no text a standard text was automatically generated and displayed. „only one-of" types of rules process errors due to carelessness. Here, the system asks for the user's decision. If this error occurs again, while processing the rules, the system removes this conflict automatically. If the rule type "substitute" and "include" cannot reach to a display then the system resolves all the conflicts quietly (automatically) from the data. The rules in this customized approach are processed sequentially and make use of Prolog's in-built facility of backtracking in a controlled way.

2.3.3.3 Human Computer Interaction:

In this section, we would like to describe how a user could interact with ABC. In the beginning of the consultation the display screen divides into three parts and remains throughout the consultation at the user's disposal:

One big window on the left side displays catalogues for selection. Here, the user can steer the system according to his/her wishes.

1. Top right, in this window all the selected options are displayed and also some extra information is provided.

2. Bottom right, in this window the actual price of a configured automobile is displayed all the time (Principally, an intelligent calculator of a sales person).

Further information, in special cases, is displayed in pop-up windows and in some cases through this mechanism the input of texts or numbers is sometimes asked for. This type of representation was chosen to have increased attention by the user.

The selection of the items in the catalogues is always made by cursor keys „up" and „down". The selected item is immediately displayed in inverse video so that it is visible to the eyes. The catalogue selection is principally made by pressing the „return" or „enter" key. Leaving or finishing of a menu is done through different keys (e.g. „POS1", „HOME", „END", „F10", and „ESC") according to the habits and comfort of the user. Within a catalogue one can move around line wise with the cursor keys and page wise with „PgUp" and „PgDn". The Tab-key is used to jump from left to right window and vice versa.

With the help of function key „F1" a help text can be displayed at any time. With the help of „F2" one can always ask for the actual code of an accessory and, if available, extra textual information about the selected accessory can also be displayed.

2.3.4 Problems Faced

Here, we would like to present some analysis of the system ABC and its implications to the structure/organization of knowledge, knowledge acquisition and Knowledgebase consistency/maintenance/sustenance.

2.3.4.1 Knowledge Structure/Organization:

In the previous section we mentioned that the data (e.g. basic models, accessories and price) was encoded into CRL-Objects but there seems to be no effort put into organizing this data into some kind of hierarchical structure (missing hierarchy). Thus, the facility of inheritance was not used at all (no inheritance) leading to redundancy in the knowledge base. In other words a lot of information - in case of hierarchical structure, which could have been represented in general concepts and later inherited by their respective specialization's - is now represented explicitly in each and every concept and is thus redundant (redundant information). This redundant information further leads to run time problems (inefficient run time behavior).

As the data is encoded into objects independent of each other, there is no way to browse through them in a graphical interface -an essential part in the expert system development tools and also provided in Knowledge Craft - which further reduces the transparency (reduced transparency) of the system. In this case one is totally dependent upon the system developer's capabilities to handle everything in an editor (one person one system). That means during the whole development cycle involvement of a non EDP- person (e.g. domain expert) is not possible or at least difficult. Furthermore, reduced transparency leads to acceptance problems among the end users and the domain experts, as the structure and organization of knowledge does not correspond to their mental organization of knowledge.

In the TURBO/PDC-PROLOG version ABC, the objects and also the rules are encoded as prolog facts and have a flat structure as mentioned above and therefore show all the above-mentioned drawbacks. Furthermore, while representing knowledge the developers of ABC have not differentiated between the different types of knowledge (Fig. 2.15).

```
p("()","u2",["325"],["310"],[],"")
sa("AA-Datum","",10000," 3. 6.1991",0,0,0)
sa("Abgasreinigungsanlage","",203,"620",985,0,0)
ob("",0,[],[],[],[],"","")
```

Fig. 2.15: Representation of different types of knowledge

Now, if we glance through the four prolog predicates shown in Fig 2.15, we are not able to differentiate between a rule and an object. Even if we somehow can differentiate between rule (predicate p) and an object (predicate sa, ob) we are unable to understand the underlying semantics e.g. why „AA-Datum" - which represents a date - is being represented as a fact and further more has the same syntax and semantics as „Abgasreinigungsanlage" which represents a component of an automobile. Similarly a dummy object „ob" is also represented. May be the developers achieved through these representations some kind of computational efficiency but from the knowledge representation point of view it creates lots of confusion.

Now let us consider the internal structure of the predicates shown in fig. 2.14 and interpretation of their syntax is shown in Fig. 2.16

```
p("Function", "Rule-Name",["Condition"],["Action1"],["Action2"], "Text")
sa("Name", "Explanation", ID-Number," SA-Code", Price, "Option", "Flag")
ob("Name", Price,[Series-Code],[ID-Code],[SA-Code],[],"Text", "Text")
```

Fig. 2.16: Interpretation of the predicates represented in Fig. 2.15

Now, if we look back at Fig. 2.13, what it shows is that there exist certain predicates, which have a forced syntax and semantics. That means all the information is encoded in such a way that it fits into the predefined predicate structure (syntax) even if it is not required. For example, in Fig 2.13, the instances of predicate „Kataloge" are the examples of how different pieces of information are represented in a forced syntax and how this leads to confusions, e.g. one interpretation suggests „kataloge" represents the class (collection) of objects and the second suggests it calls up a function to handle either a search procedure or a procedure to manipulate the user-interface.

Furthermore, the forced and strict syntax of the predicates leads to reduced flexibility while acquiring, representing and maintaining the knowledge and also causes problems in keeping the knowledge base consistent.

2.3.4.2 Knowledge Acquisition:

Let us now analyze both versions of ABC from the knowledge acquisition point of view. As the developers of ABC - in the KC-version of the system - have decided to go in for a flat structure of the knowledge represented, thus each new piece of knowledge has to be encoded in that form. This means while adding a new object into the knowledge base one has to be careful about its internal structure i.e. all the properties of the object have to be explicitly defined even though it may be a specialization of an already existing object. Same is true if we need to delete an object's specific properties. Furthermore, an addition of an object into the knowledge base may cause an addition or editing of a rule and/or a method, which leads to an overhead in computation. Thus, one has to keep an exact behavior of the system in mind during the processing and for that reason one is dependent upon an individual who is either the system developer or a trained and skilled programmer of the tool/language used to build the knowledge base.

The same reasons apply to the prolog-version of the system too, because the underlying principals of knowledge representation here are also the same as above i.e. a flat structure of objects. Here, additional confusion occurs when facts and rules are both defined as prolog predicates. So, one should be aware of what predicate defines a rule and what predicate defines a fact. Things get worse while defining the rules, when the rules don't have unified semantics e.g. in prolog version of ABC there are five different types of rules. Not only that, their internal structure (syntax) too is different from each other.

Further, these rules are processed by a rule interpreter, which is a constant, thus in order to do the proper processing the rules have to have exactly the same structure in which they can be processed by the rule interpreter. This causes further consistency problems.

2.3.4.3 Knowledgebase Consistency/Maintenance/Sustenance:

From this point of view, the more redundant information there is, the more difficult it is to keep the knowledge base consistent. The KC- and the PROLOG- version are both victims of this principle. Furthermore, due to the missing hierarchies in the KC-version each object requires a rule or a method for its manipulation or processing, which means at each addition or deletion of an object one is forced to manipulate the corresponding rule or method.

Now, a careful analysis of the Prolog knowledge base suggests the following types of consistency errors:

Facts consistency

I. *Simple prolog syntax (data structure) errors:* Fig. 2.17 shows how the data structures for facts in prolog version of ABC are declared.

```
Domains
  list,liste,integerliste = integer *
  stringliste,secliste,pliste,saliste,serienliste,xliste,
  yliste = string *
  cliste,oderliste = stringliste *

database - mydba
  sa(string,string,integer,string,real,integer,integer)
  kataloge(string,stringliste)
  c(string,string,liste,cliste,liste,string)
  cp(string,string,liste,secliste,oderliste,integerliste,string)
  p(string,string,pliste,xliste,yliste,string)
  ob(string,real,serienliste,liste,saliste,liste,string)
```

Fig. 2.17: Declaration of data structures in prolog version ABC

Here, three kinds of lists have been declared (e.g. a list containing integers, a list containing strings and finally an embedded list containing a list of strings).

Furthermore, for example a fact representing „SA“ is defined as:

sa(string,string,integer,string,real,integer,integer)

This kind of arrangement - throughout the knowledge base construction - expects from the developers/system maintainer not to make any kind of typing or conceptual error in their representation. If by any chance, one makes a mistake while representing the facts then the following are the possible mistakes (Fig. 2.18) and - in a large fact base - they are neither easily detectable nor correctable without the help of an automated syntax checker.

```
#
1101 Integer erwartet (Bitte Integer eingeben).
#
1102 Reelle Zahl erwartet (Bitte Reelle Zahl eingeben).
#
1103 Anf#hrungszeichen erwartet (Bitte Anf#hrungszeichen eingeben).
#
1104 Hochkomma erwartet (Bitte Hochkomma eingeben).
#
1105 Listen Anfang erwartet (Bitte "[" eingeben).
#
1106 Listen Ende erwartet (Bitte "]" eingeben).
#
1107 Functor in Domain nicht gefunden.
#
1108 '(' erwartet (Bitte Klammerauf Zeichen eingeben).
```

Fig. 2.18 Possible syntax errors in ABC (ABC’s input/output is in German)

II. *Length of text less than 38 characters:* This particular error is due to the compromise made by the developers who would rather have a constant size of the text display section than to have a variable display mechanism which changes according to the length of the text or having a mechanism where the text automatically adjusts to the size of a display.

III. *A SA-Code cannot have two different texts*

IV. An Accessory SA cannot have two different SA-Codes

V. A SA-Code cannot have the same identity number

The error III, IV and V actually fall under the category of semantics chosen for the facts representing but occurs due to the manual input of the facts into the fact base and are also difficult to remove manually from a large knowledge base.

Rule Consistency:

In this section we present the possible errors which can occur in ABC's rule base.

- *Rule Syntax:* The ABC's developer chose the convention that all the rules must have a unique name but while representing the rules one found that in three different cases this convention was hurt. As the rule base in due course increased, it was not possible to detect these errors manually. Following are the cases where this convention was hurt:

  Case I: When the c rules are defined as follows:

  ```
  c("$","c46",[32],[["C30","C32"]],[34],"")
  c("$","c46",[33],[["C30","C32"]],[35],"")
  c("$","c46",[36],[["C30","C32"]],[37],"")
  ```

  and the general interpretation of the rule is:

  If C30 and C32 exist Then replace 32 with 34.

  Case II: When the p rules are defined as follows:

  ```
  p("()","u2",["325"],["310"],[],"")
  p("()","u2",["326"],["310"],[],"")
  ```

  and the general interpretation of the rule is:

  325 includes 310

  Case III: When the p rules are defined as follows:

  ```
  cp("!","v1001",[],["E88"],[["E73"]],[],"")
  cp(">","v1001",[],["E11"],[["V10"]],[],"")
  ```

  and their respective interpretation is:

E88 contradicts with E73 &
If E11 exists Then exists V10.

- Circular Rules: In this section rules were checked for their circularity i.e. for loops and following are two such cases resulting in loops:

p("()","u1",["H48"],["H47"],[],"")
cp(">","V30",[],["H47"],[["H56","H57"]],[],"")

Here the first rule says H48 includes H47 but the second rule says that the existence of H56 & H57 presupposes H47.Thus resulting in a loop.

p("%","ex32",["V22","V24","V25","V26"],[],[],"")
cp(">","V41",[],["V22"],[["T25","T26",V24]],[],"")

The first rule says that only one of V22,V24,V25 or V26 exists but the second rule has V22 as the precondition for the existence of T25, T26,V24. Thus resulting in a loop.

Before we can say that the knowledge base of system ABC is consistent, we have to remove all the above-mentioned errors from the knowledge base and in the example we have seen how difficult it would be to detect and then correct these errors manually in a large knowledge base. For details see [Thakar'89, Gielisch'89 and Puls'91].

2.3.5 Inference Mechanism:

In the TURBO/PDC-PROLOG version of ABC, the rules are encoded as prolog facts and have a flat structure as mentioned above (s. section 2.3.4.3) and therefore show all the above-mentioned drawbacks. Here, following six types of rules:

1. ( ) → includes
2. ersetze → replaces
3. % → only one-of
4. > → pre condition
5. ! → contradicts
6. prisregel → price rule

are sequentially processed and the inferences provided by the prolog (i.e. backward chaining) is used. However, the rules are broadly categorized in following broad categories to facilitate a four-step inference mechanism. The four steps of the inference mechanisms are:

1. Universal (Domain) Facts: rules are fired automatically to make the decision without consulting the user.
2. Resolvable Constraints: independent but revisable decisions of the system during the technical conflicts.

3. Hard (not resolvable) Constraints: system queries the user in case of multiple choice solutions for the given technical conflict (choice of different alternatives, retraction of constraint)
4. Calculative Knowledge: Analysis of valid technical solution and calculative assessment of the codes.

2.4 Re-Knowledge Engineering (Methodology Adopted): New Approach:

In recognition of the tremendous burden presented by the aging XYZ system and escalating maintenance costs, a task in front of us was to find a suitable structure for the knowledge incorporated in the present XYZ. This is a task of re-structuring the knowledge already incorporated into a pre-existing system at a higher level of abstraction (physical, logical, conceptual) so that the resulting knowledge base is an „open system kernel“ which could be used not only for configuration but could also be extended for say consultation, multi-media product information system or production planning, design and for designing advanced future oriented marketing and sales strategies. In order to revitalize the existing system and relieving the maintenance burden, we call this process of restructuring “Re-knowledge-engineering” (Fig. 2.19).

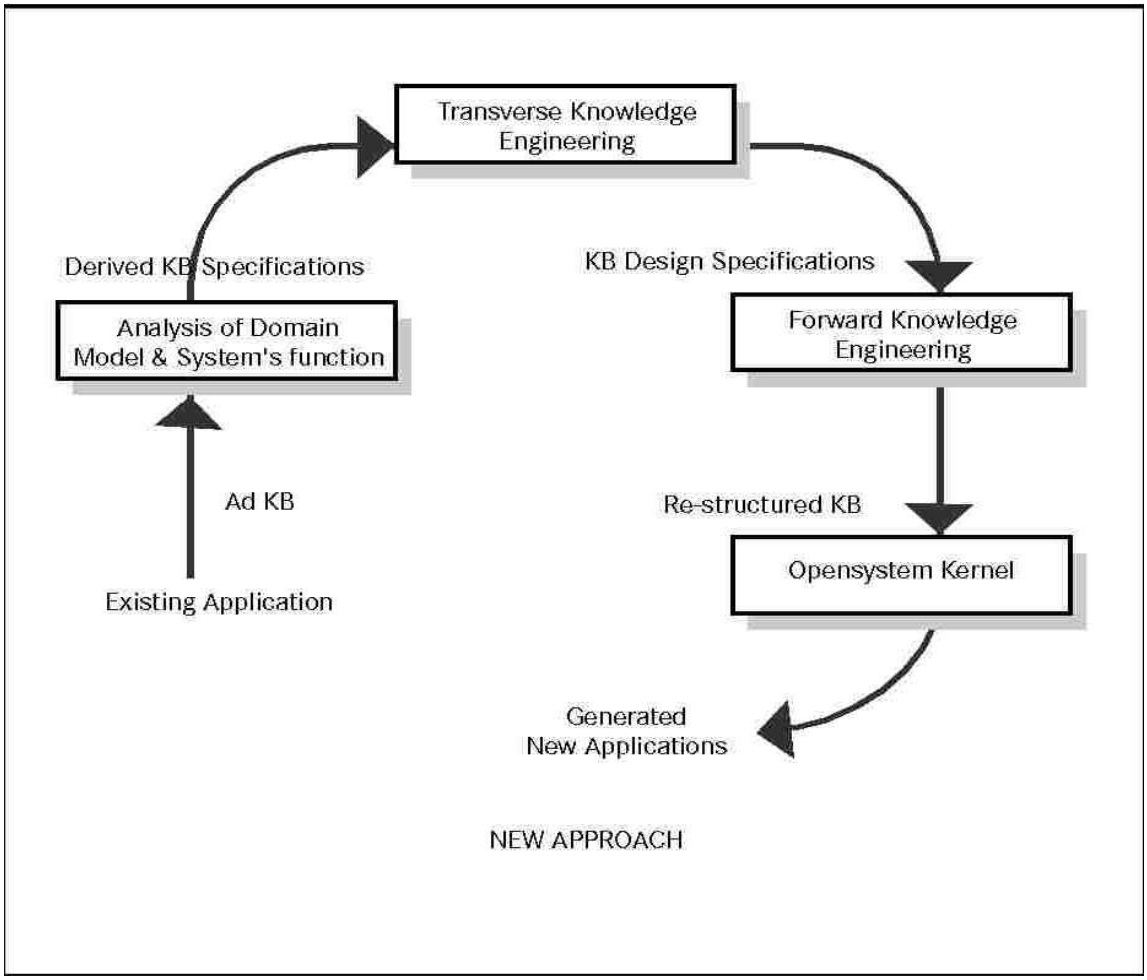


Fig. 2.19: Re-Knowledge Engineering (new) Approach

Re-knowledge-engineering is closely related to the notion of domain analysis [Neighbors'80] and reverse knowledge engineering [Brachman et al. 91]. There are some differences: the domain analyst looks at a variety of systems that service the same application domain and produces an external description of a set of reusable components, which can be used to build applications for that domain. The reverse knowledge engineer, on the other hand, looks at the architecture of a specific large system and produces an internal description of the components of the system and the architectural framework that unites them. Contrary to both, the re-knowledge engineer analyses the domain model and follows the cycle of Re-Knowledge engineering.

Despite the differences, the results of all three kinds of analyses include a description of a set of components. Thus a terminology used by domain analyst and reverse knowledge engineer to describe the results of their respective effort is pertinent to a re-knowledge engineering effort. All Neighbors [Neighbors 80 & 84], Diaz [Diaz'91] & Brachman [Brachman et all '91] suggest a model based on objects and actions.

3 Knowledge Representation & Reasoning

3.1 Introduction and Overview

This chapter presents an introduction into the field of knowledge representation & reasoning, which is also referred to as KR. Here, we not only present the fundamental ideas of this field but give a short overview of historical developments as well. Further, some important issues in knowledge representations are dealt with. In the next section state of the art KR systems (KRS) are presented, followed by the general-purpose hybrid logic-based KR systems. These systems are compared with other KR-systems and the pros and cons of such systems are given. This chapter closes with the deficits of KR systems.

3.1.1 Short Overview of Historical Developments

Knowledge representation is the field of Artificial Intelligence, which focuses on the design of formalisms that are both epistemologically and computationally adequate for expressing knowledge on a particular domain. The field of knowledge representation was originated in the late 1960's and has given rise to various paradigms of knowledge representation ranging from associational representation highlighting *Semantic Networks* [Quillian '68] to object representations exemplified by *Frame* representations [Minsky '75]. On the other hand *first order logic* [McCarthy et al. '69] has played a central role in knowledge representation and has given rise to formal logic-based representations. Another important trend in knowledge representation can be identified in *procedural representation* [Hewitt '72] and *production systems* [Shortliffe et al. '77]. Other approaches, which cannot be easily categorized, include *default reasoning* [Reiter '78], *hybrid* approach to representation (integrating two different representation schemes) [Brachman et al. '83], more *pictorial (Diagrammatic)*, less linguistic representations [Funt '80], and inexact, *evidential reasoning* [Garvey et al '81] to name a few. Specific KR systems may be better suited to specific problems, but much of KR has proceeded under the banner of "general purpose" KR languages and systems. KR has always been principally concerned with the design of forms for expressing information, ranging from informal memory models to complex formal languages. In the 80's, semantic networks, frames, predicate logic and production systems were the predominant representational paradigms. By the mid 80's, combining multiple types of representation was popular and hybrids of various sorts were developed, including several marrying logic and frames. Sorted logics grew in popularity, and commercial expert system shells generally offered several loosely integrated types of representation. However the following statement can easily express one problem, which attracts a great deal of attention from the scientific community:

"...But fundamental difficulties exist in representing and reasoning at the level of real-world complexity, and no surveys or textbooks exist that can guide us through those problems"

[Brachman et al. 85]

The research in KR&R in last 10 years is following in the direction of identifying these difficulties while applying KR systems in complex real-world domains and providing some answers to the following questions posed in the same context:

1. What issues arise in applying knowledge representation systems to real problems and how can they be addressed?
2. What are the theoretical principles in knowledge representation and reasoning?

3. How can these principles be embodied in knowledge representation systems?

In recent years, the quest to find reasonable answers to the above-mentioned questions is gaining lots of interest in the KR researcher's community. [see, KR96 and KR98 proceedings]

3.1.2 Fundamental Ideas of this Field

In the history of Artificial Intelligence (AI) up till now the two fundamental and most talked about issues of importance are the acquisition and representation of knowledge. The notion of representing knowledge is, at heart, easy to understand. It simply has to do with writing down, in some language or communicative medium, descriptions or pictures that correspond in some salient way to the world or a state of the world. In Artificial Intelligence (AI), we are concerned with writing down descriptions of the world in such a way that an intelligent machine can come to new conclusions about its environment by formally manipulating these descriptions.

Despite the apparent simplicity of this general goal, the research area of Knowledge Representation has a long, complex, and as yet non-convergent history. Despite the fact that just about every current AI program has what is called a "knowledge base" containing symbolic descriptions represented in some "representation scheme", there is still a vast amount to be understood before, for example, the knowledge in one system can be shared with another. There are tremendous subtleties in the notions of "representation", "knowledge", and their combination in AI.

3.1.3 Issues in the Representation of Knowledge

Before beginning to explore these subtleties, one must deal with the important issues surrounding knowledge representation.

The first issue focuses on the overall adequacy of a knowledge representation system and includes topics such as Expressive Adequacy and Reasoning Efficiency.

Another cluster of issues involves what might be called the basic epistemology of a knowledge representation system: where one considers the role of primitive knowledge structures, meta-representations and incompleteness (How incomplete the system's knowledge can be?).

Given the importance of logic in the field, several issues focus on aspects of knowledge representation not covered by standard logic, for example: Definitions vs. Facts, Universal vs. Defaults, Non-Deductive Reasoning and Non-Monotonic Reasoning.

Yet another group of issues is concerned with representing knowledge in ways that are quite different from simple yes-no sentences and deal with representational & reasoning schemes such as: Procedural, Analogical and Probabilistic.

The final grouping of issues is concerned with "what exactly the knowledge being represented is about?" this covers the problems occurring while representing substances (e.g. a pound of butter or a quart of milk), causality and Time (how to represent an event as something changing in time) and finally (knowledge to represent knowledge about other agents) it is often necessary to deal with propositional attitudes such as their (agents) beliefs, desires and intentions which take propositions as arguments.

3.2 Knowledge Representation Systems: State of the Art

KR has always been principally concerned with the design of forms for expressing information, ranging from informal memory models to complex formal languages. In the 80's, semantic networks, frames, predicate logic and production systems were the predominant representational paradigms. By the mid 80's, combining multiple types of representation was popular. Hybrids of various sorts were developed, including several marrying logic and frames. Sorted logics grew in popularity and commercial expert system shells generally offered several loosely integrated types of representation.

One important criterion for separation of hybrid components [Brachman & Levesque '82] distinguished between terminology (knowledge about the meanings of terms, independent of the existence of any objects exemplifying those terms) and assertion (knowledge of contingent facts). A large family of terminological and hybrid systems were developed inspired by KRYPTON [Brachman et al. '83] and the roots can ultimately be attributed to the work done on KL-ONE [Brachman & Schmolze '85] in the late 70's and early to mid 80's. In the following years several KR systems were developed incorporating different dialects but the underlying representation philosophy remained the same. The respective formalisms and systems were propagated under different banners like KL-One alike systems, term subsumption systems, concept logics, terminological logics and description logics.

3.2.1 General Purpose Hybrid Logic-Based KR Systems:

Research in knowledge representation led to the development of terminological representation languages, which mainly originated from Brachman's KL-ONE. Work on terminological languages further led to hybrid representation systems like BACK, CLASSIC, LOOM, KANDOR, KL-TWO, KRYPTON, MESON and SB-ONE [Heinson '91]. They have been developed mainly in response to two different demands: First, the need for representation tools equipped with clean and well-defined semantics at the level of „what is to be represented" rather than „how is it to be represented" and second the need to provide these representational facilities in a computationally efficient manner. Thus, they are an attempt to bridge the gap between general purpose theorem provers, which are semantically well-founded though extremely inefficient, and commercially available KR tools, which are more on the efficient side, but lack a semantic foundation [Schmiedel '88].

All systems based on the KL-ONE paradigm have embraced the ideas of frames as structured description, differentiation between terminological and assertional aspects of knowledge representation and the central nature of subsumption and classification inferences [Brachman 90].

Generally, the terminological knowledge base (TBox) stores terms and their definitions, while the assertional knowledge base (ABox) stores sentences constructed using these terms. In other words, the definitions stored in the TBox are taken as describing Platonic concepts and their analytic interrelations, while the assertions stored in the ABox state contingent facts about what is true of the world [Doyle & Patil '89].

Furthermore, storing definitions and assertions separately clarifies the meanings of representation and moreover, optimizing definitional and assertional inferences separately maximizes the efficiency of the representation system [Brachman et al. '83].

3.2.2 Classification:

Classification is based on the notion of subsumption of concepts [when C and C' are concepts (for discussion, unary predicates defined in logical language) such that in every model the extension of C is a superset of the extension C' i.e. every instance of C' is an instance of C, then one can say that C subsumes C'] and is closely related to the logical notion of entailment (i.e. if a concept both subsumes and is subsumed by another concept, then its definition is logically equivalent to the definition of the other).

To classify a new concept with respect to a hierarchy of concept definitions means to determine its place in the subsumption partial order. Thus as each new concept is defined, KL-ONE style representational languages automatically compare its definition with previously encountered definitions and place the concept in the taxonomic hierarchy, explicitly recording the derived subsumption relationship.

In addition, KL-ONE-like systems use classification for answering queries. Whenever a description is used in a query, it is classified as though it were a new definition. Once this is done, the query can be answered from the recorded subsumption relation, which indicates where to look for inheritable information.

Classification performs several important functions for knowledge-base maintenance, e.g.

1. It detects redundancies by recognizing definitions equivalent to ones already present in the hierarchy i.e. equivalent definitions are collapsed so that they are represented by the same concept.
2. It helps to keep the hierarchy coherent and nontrivial by recognizing inconsistent and vacuous definitions [i.e. Any concept subsumed by the empty concept is equivalent to the empty concept (hence its definition is inconsistent), and any concept subsuming the universal concept is equivalent to the universal concept (hence its definition is vacuous)].
3. Classification also helps to speed up query answering when the subsumption relation among concepts is explicitly and uniformly recorded whenever a new concept is added to the hierarchy.
4. Another important motivation for automatic classification is that it can be used to recognize concepts from their features. For example, if a human parent is defined to be a human who has a child, recognizing that some person is a parent allows one to apply reasoning appropriate to parents in addition to that appropriate merely to humans [Doyle & Patil '89].

3.2.3 Inferences:

There are numerous deductive inferences that KL-ONE style languages provide:

3.2.3.1 Completion:

Logical consequences of assertions about individuals and descriptions of concepts are computed; there are a number of „completion" inferences one can make:

Inheritance: restrictions that apply to instances of a concept must also apply to instances of specialization of that concept; in a sense, then, properties are „inherited" by more specific concepts from those that they specialize.

Combination: restrictions on concepts and individuals can be logically combined together to make narrower restrictions.

Propagation: when an assertion is made about an individual, it may yield logical consequences for some other related individual; KL-ONE style languages then „propagate“ this information forward when an assertion is made.

Contradiction detection: it is possible to accidentally assert two facts about an individual that are logically impossible to conjoin together; KL-ONE style languages detect this kind of contradiction.

Incoherent concept detection: it is possible to accidentally give a concept some restrictions that combine to make a logical impossibility thereby not allowing any instances of the concept to be possible; KL-ONE style languages detect this kind of inconsistent description.

3.2.3.2 Classification and Subsumption:

Concept classification: all concepts more general than a concept and all concepts more specific than a concept are found.

Individual classification: all concepts that an individual satisfies are determined.

Subsumption: questions about whether or not one concept is more general than another concept are answered (important during concept classification).

3.2.3.3 Rule application:

Simple forward-chaining rules have concepts as antecedents and consequent; when an individual is determined to satisfy the antecedent of a role, it is asserted to satisfy the consequent as well.

3.2.4 KL-ONE style languages versus:

Standard logic & Production systems: The most notable feature of KR systems based on KL-ONE is the „self-organization“ of concepts defined: because concepts have clear definitions, it is possible to have the system organize them into the subsumption hierarchy, rather than have the user specify their exact place. This is important because standard logic and production systems, for example, do not address the knowledge engineering issue of organizing large collections of knowledge. [Brachman et al. 90].

Standard Database management systems: are relatively poor at supporting schema changes, in comparison to straight updates to data. In contrast to standard repositories of data (traditional database), a knowledgebase based on KL-ONE style languages allows the user to maintain a partial, incomplete view of the domain of discourse, a view in which information is incrementally acquired.

Relational database systems: KL-ONE style languages are not just a passive repository for unconnected assertions, like a relational database; the system actively searches to find an

entire class of propositions entailed by the facts it has been explicitly told (KL-ONE style languages are deductive).
Partial, incomplete descriptions of individuals are acceptable (KL-ONE style languages are incremental).

The system tracks dependencies between facts and allows certain facts to be retracted (KL-ONE style languages support knowledge retraction).

Object-Oriented programming languages: usually have inheritance but not classification and neither they are truly incremental nor do they support knowledge retraction.

Hypertext or Hypermedia systems: are manually cross linked and generally use the simple indexing scheme based on keywords; whereas KL-ONE style languages support sophisticated indexing schemes based on classification.

	KR-systems			DB	Prog.Lan.	IS
Systems / Characteristics	KL-ONE Style Languages	Production Rule Based	Standard Logic Based	Relational Database	Object Oriented Programming Languages	Hypertext/ Hypermedia
Expressiveness	descriptive	limited	descriptive	limited	limited	limited
Supports Classification	Yes	no	no	no	no	no
Knowledge Retraction	Easy & Supported	Complex & Supported	--	Complex & Not Supported	Complex & Not Supported	Complex & Not Supported
Automatic Consistency Check	Yes	no	partial	no	no	no
Links	Automatic	manual	manual	manual	manual	manually cross linked
Organisation of Concepts	Automatic	--	--	manual	manual	manual
Indexing	Sophisticated Based on Classification	--	--	Simple Keyword Scheme	--	Simple Keyword Scheme
Partial & Incomplete Knowledge	Supported	--	--	Not Supported	Not Supported	Not Supported

Fig. 3.1: Comparison of KL-ONE Style Languages versus other KR systems

3.2.5 Applications for which KL-ONE style languages can be useful

Domains with a large set of objects: KL-ONE style languages can be exploited in any domain where it is useful to organize a large set of objects that can naturally be represented in terms of „features“ or „roles“. For example it has been argued that this kind of automatic classification is a useful way of organizing a large set of rules in some expert system [Yen et al.’89].

Another example of such a family of applications would be information retrieval, where every object (An object might be a text document, some software component, chemical compound

etc.) has a complex description, and a query may be phrased as a description of objects having a certain structure (e.g. find all meals with at least two courses, each of which has a sweet wine as its drink). In such cases, the description can be classified with respect to each other so that similar objects are grouped together. This can provide a much more sophisticated indexing scheme than simple keyword schemes, without increasing retrieval time significantly since everything is pre-classified. (Possible application: Multi-Media Corporate/Product Information retrieval; could be used by press & publicity department, strategic planning department or strategic decision makers)

Database-like Applications: because the hierarchy of concepts can change dynamically, KL-ONE style languages are more appropriate for database-like applications that have an evolving schema- for example, as this is the normal state of affairs in design and specification efforts. (Here as application domain one can think of various databases for manufacturing e.g. "Machining Database").

Incrementally evolving descriptions: another class of applications consists of those involving incrementally evolving descriptions. Where a knowledge base allows the user to maintain a partial, incomplete view of the domain of discourse, a view in which information is incrementally acquired. This ability to handle partial knowledge can be usefully exploited to support tasks such as design or configuration of artifacts (where something is being created, without having an exact idea of all its parts until it is complete) [Owsniciki-Klewe, 1988] (e.g. Automobile Configurator).

Yet another suitable application class would be where one wants to enforce constraints on collections of facts because inheritance is strict and „trigger"-like rules are also available (i.e. configuration) [Brachman et al.'90]; such systems have been used in the domain of configuration as an integrity checker (automobile manufacturing).

3.2.6 An Application Scenario:

A potential problem area in a large corporation could be the maintenance and upkeep of an updated view of the social structure of the corporation, which evolves in time due to company acquisitions, disposals and share transfers. Such a view is fundamental for top management who are in charge of defining and supervising the economic profile of the corporation as well as for the shareholders.

Generally the overall knowledge is maintained by people very close to top management, split among several offices with different tasks, often duplicated and stored on paper. In general access to this bulk of information is not simple; complex requests often need to be decomposed into sub queries to be turned over to the competent offices.

In context of an above-mentioned scenario, a possible application would be to develop a knowledge base as the central repository of the social structure of the corporate.

The social structure of a corporate could be quite complex, as it incorporates several companies in different countries, which are involved in several commercial activities, for example:

Electrical Equipment
Automobile
Aerospace
Insurance, Finance & Services.

All these corporate divisions are extremely heterogeneous. They may have interests in different countries, thus are subject to different rules & regulations or laws depending upon the legislation of the host nation. As they are engaged in diversified business in several economic areas and have different administrative organization, in part deriving from their country-dependent judicial status.

As a consequence, legal and financial terminology as well as jargon specific to the corporate or its divisions is extensively used both by experts in specifying and users in accessing the domain description. In some cases terminology is precise and stable, in others it could be ambiguous and not fixed.

In general, terminology acquisition is quite a complex and time-consuming activity, though essential for the soundness of a domain model. It leads to the identification of relevant conceptual entities in the domain and to the specification of their features including relationships with other entities, which results in the so-called terminological model.

3.2.7 Requirements for the Representation Services:

Terminology in highly structured domains like the one described in the above mentioned scenario, presents specific features that it would be desirable to capture in the terminological model and thus in the representational formalism:

Description Equivalence: Often the same conceptual entity can be referred to either by using the specific terminology or through properties, which unequivocally identify the object. For example the queries, as posed to the application domain:

1. Which of the German companies are listed on the New York Stock Exchange (NYSE)?

2. Which of the "AG's" (here, for illustration purposes it is assumed that only AG companies can be listed on the NYSE) are listed on the NYSE?

Have an identical answer, i.e. the set of companies which are formally declared as located in Germany, with capital subdivided in shares, with shares of the specific type that make them quotable on a Stock Exchange and that are listed in New York; Given Query 1, one can derive, reasoning on the terminology, that the answer consists only of companies with judicial status AG.

In fact, only those companies whose capital is subdivided in shares can be listed on the Stock Exchange and the German companies with such property are only AG companies. Vice versa, given Query 2 and reasoning on the definition of AG, one can infer that the answer consists of the companies that are located in Germany with their capital subdivided in shares of specific types. Some terms can be identified by both necessary and sufficient properties.

One can assert or retrieve a specific entity either through the proper domain term i.e. an AG company or through the distinguishing properties, i.e. A German company with...., in turn relying upon further terms. A knowledge base exhibiting the behavior described above must thus incorporate both an accurate definition of domain concepts and an inferential engine for reasoning on the properties specified at the structural level.

Complex Term Interdependencies: Domain terms can strictly be interrelated. For example, to remain in the legal area, the judicial status of a company determines the form of its administrative apparatus (i.e. Board of Auditors, Board of Directors, etc.) and constrains the

set of administrative/financial events the company is subjected to (i.e. capital addition, share participation/transfer etc), financial assets, the type of business activity of the company. Consequently an assertion such as the following:
The company X had a capital addition of $10mio. in ordinary shares

Is meaningful only if the company X has a judicial form for which the capital formally declared can be increased and such capital consists of shares that in turn can be ordinary. If, for instance, X is a non-profit company, then the assertion is wrongly formulated because by definition it makes no sense.

The inconsistency can be detected only if judicial terms are properly correlated to the concepts of capital addition and, consequently, capital and share. In that hypothesis, the resulting description is a complex net of term definitions, one strictly depending on the other. For the description to be sound, overall consistency has to be guaranteed.

Taxonomic Organization: In the domain, terminology is often inherently structured in taxonomy. It is quite common, both in legal clauses and in the jargon of the corporation or its divisions to find terms ordered in a hierarchy of abstractions. For example, with respect to the economic activities of companies, management presents these as grouped in increasingly specialized areas.

For instance the agro-industry business consists of activities of transformation of agricultural products, that can in turn be further specialized on the basis of products, trading etc. and that can further be specialized depending upon the forms of trading and so on.

Inference on Terms Definitions: For a sound representation, the positioning of a given object into the taxonomic description must respect the terminological definitions so far introduced. For example, given the economic activity of growing, defined as a type of production limited to the agricultural sector, if a company is engaged in soybean production, it will necessarily be involved in the growing business as well. In this case the positioning of soybean production is the result of an inference on term definitions.

Retrieval Facility and Model Scalability: The terminological model has to be accessed not only for updating but also above all for querying purposes. In general, articulated queries require a complex navigation of the model and consequently powerful mechanisms of query interpretation. In the example domain, the dimension of the Corporate and its heterogeneity entails a remarkable quantity of terminology to be modeled.

3.2.8 When not to use KL-ONE style languages:

To provide effective reasoning services, certain expressive features have been deliberately left out of the languages belonging to KL-ONE family. Thus there are situations where these languages are not appropriate as representation tools. For example, it would be cumbersome to use such languages in cases where mathematical entities such as tupels, sequences, geometric entities, etc. are the center of attention. Defaults and exceptions cannot be easily encoded in such languages due to strict inheritance. Applications requiring complex conditions in the antecedent are much more difficult to handle because of the limited form of rules, where both antecedent and the consequent refer to membership of a single individual in

a concept. If an application will constantly need to refer to a concept that includes everything that is not an instance of some other concept, then the application is not well suited for such languages, as they do not support full-negation. Problems can also arise in situations where „ontology“ of the domain is not self-evident. Moreover, as these languages are designed as general-purpose reasoners, and do not support the direct and efficient addition of specialized kinds of inference. For example, applications needing to make intensive use of temporal or spatial reasoning would find it difficult to have these languages deduce the desired relationship.

3.3 Description Logics

Just about every current AI program has what is called a “knowledge base” containing symbolic descriptions represented in some representation scheme. For implementing the knowledge, an operational description of rationality is needed and logic has always been considered an important aspect of rationality. Logic can be used to generate new knowledge based on given knowledge. Because of the importance of Logic this chapter provides an insight into the description logics and discusses the reasoning techniques. The exposition starts off with basics and proceeds to a rather detailed exposition of technicalities.

3.3.1 Origin of Description Logics (Roots/History)

The field of knowledge representation found its first paradigm in late 1960s as *semantic networks* [Quillian 68]. Here, Quillian assumed semantic memory to be general memory. In this early stage Knowledge Representation was assigned the task of modeling the human ability to use information for intelligent activities like perception, planning and natural language understanding.

The second paradigm of KR is due to Minsky´s work on *Frames* [Minsky 75]; where the main goal was to account for the effectiveness of common-sense reasoning in real world tasks – to a large extent motivated by “human “ models borrowed from psychology or text linguistics.

The notion of *Semantic Networks* and *Frames* is quite similar but do have some important differences: semantic nets adequately convey the general structure of the represented information, especially interconnections and dependencies; a frame representation, on the other hand, focuses not so much on the overall structure but on the basic units and the information locally associated with each frame [Quantz 93].

In most of the second half of the 70s the KR community was faced with the problem of the missing entailment relation for semantic nets or frames: how do we know what exactly a semantic net or a frame means and which contains more information than the other. Hayes [Hayes 80] answers this question by mapping frame definitions into first order logic (FOL) formulae and shows that Frames are not superior to FOL with respect to expressive adequacy.

In late 70s and early 80s Brachman [Brachman 77 & Brachman 79] endorsed the logic-oriented view on knowledge representation. Brachmann examined in detail what the constructs used in semantic nets were supposed to represent. In his work Brachmann distinguishes between different levels of KR systems: the implementational, the logical, the epistemological, the conceptual and the linguistic level. In addition he proposes KL-ONE as formalism on the epistemological level [Brachman, Schmolze 85].

From then onwards several similar systems (e.g., CLASSIC [3], LOOM [8], BACK [12], & KRIS [1] were developed based on the knowledge representation philosophy proposed by Brachmann. The respective formalisms and systems were called KL-ONE alike systems, term subsumption systems, concept logics, terminological logics and description logics.

3.3.2 Description Logics: An Introduction

Description logics (DLs) are descendents of the KLONE semantic network/frame system and formalize the reasoning about concepts of frame-based KR systems in a variable-free notation of terms. They have a variety of "concept constructors" motivated by empirical experience with object-centered KR in general, "world modeling". In other words DLs are languages tailored for expressing knowledge about concepts and concept hierarchies. They are usually given Tarski style declarative semantics, which allows them to be seen as sub-languages of predicate logic. One starts with primitive concepts and roles, and can use the language constructs (such as intersection, union, role quantification, etc.) to define new concepts and roles. Concepts can be considered as unary predicates, which are interpreted as sets of individuals whereas roles are binary predicates, which are interpreted as binary relations between individuals. Concepts and roles can be either primitive or complex. Complex concepts are defined via descriptions built from a set of constructors. Description logics vary in the set of constructors they allow, and consequently in the complexity of determining class subsumption. The main reasoning tasks are classification and subsumption checking. Subsumption represents the "is-a" relation. A whole family of knowledge representation systems has been built using these languages and for most of them complexity results for the subsumption algorithm is known. Description logic systems have been used for building a variety of applications including software management systems, planning systems, configuration systems and natural language understanding.

3.3.3 Theoretical Foundations

The fundamental observation underlying DLs is that there is a benefit to be gained if languages talking about classes of individuals yield structured objects that can be reasoned with [Borgida 1992, DLs are not for the Flightless-Birds...]. Following is an example of a typical compositional description in CLASSIC.

(**and**

COURSE

(**at-most** 10 participants) ,

(**all** participants GRADS))

Its intended reading would be "*Courses with at most 10 participants, all participants being instances of GRADS*". In this description, COURSE and GRADS are identifiers for concepts introduced elsewhere, while 'participants' is the name of the binary relation, intended to relate courses to students taking them. One can to do several things with such a description, like:

- *Recognizing* those individuals that satisfy the description, based on what is currently known about them. For example, say EXSYS00 is an individual object in the knowledge base, and it is known to be an instance of the concept COURSE; further, the fillers for the participants role for EXSYS00 are individuals Jason and Mini, both of whom are instances of GRADS. Then EXSYS00 is inferred to be an instance of the concept defined above, since all of *necessary and sufficient* conditions of that concept are satisfied.
- *Reasoning* about the relationship of one description to others, treating them as "intentional" objects. For example the description given in above example is subsumed by (entails) the following description:

(**and**

COURSE ,

(**at-most** 15 participants))

Since everything with at most 10 fillers for some role, also have at most 15 fillers for it. Similarly, the description, the description (at-least 4 participants) can be inferred to be inconsistent with (all participants (one-of Lorenz, Christian, Jan)) since the number of fillers for anything satisfying the latter description is at most three.

DLs are considered to be the descendants of the influential KL-ONE system [Brachman '77, Brachmann & Schmolze '85] and have been extensively studied under the name of "terminological logics". The features and history of these logics have been surveyed in [Woods and Schmolze 92, MacGregor 91]. Here our aim is to give the reader a sense of syntactic variations in use and many different possible semantics.

3.3.4 The Syntax of DLs

We begin by considering syntactic aspects of the DLs. Though the original KL-ONE system supported a graphical notation for representing definitions of concepts, all DLs since the KRYPTON system [Brachmann et al. '83] provide a formal linear syntax for writing descriptions. Following is the above-mentioned description in two other DLs, LOOM [MacGregor '87] and BACK [von Luck et al. '87], using an infix notation used in many theoretical papers:

COURSE and **atmost**(10, participants) and **all**(participants, GRADS)

(**:and** COURSE (**:at-most** 10 participants) (**:all** participants GRADS))

COURSE $\cap$ $\leq$10 participants $\cap$ $\forall$ participants : GRADS

In [ait-Kaci '84] described that it is useful to view DLs as special languages obtained by *term composition*. All DLs have (at least) two sorts of terms: *concepts* (intuitively, denoting collections of individuals), and *roles* (intuitively denoting relationships between individuals). Thus, the syntax of DLs consists of rules for creating composite terms from atomic symbols — identifiers of various sorts — and *term constructors*.

Following is syntax of a very simple DL:

< conceptDesc > :=

thing ;; the universal concept, resembling True

| **nothing** ;; the empty concept, resembling False

| **prim**(< atomic-concept-id>) ;; concepts without necessary and sufficient conditions

| **and**(< conceptDesc> +) ;; concept intersection

| **all**(< roleDesc>, <conceptDesc>) ;; restricting the range of roles

| **at-least** (< pos-int> , <roleDesc>) ;; lower bound on number of fillers

| **at-most** (<non-neg-int> , <multiroleDesc>);; upper bound on the number of fillers

< roleDesc> :=

<Role-id>

| **compose**(< Role-id> , <roleDesc>) ;; relation composition

The following would be a syntactically valid description in this term language

and(**prim**(COURSE),

at-most(20, participants),

all(compose(taughtBy, rank), **prim**(TENURED-RANKS)))

denoting those instances of the (primitive) concept COURSE, which have at most 20 students taking them (at most 20 fillers for the participants role), and are only taught by tenured persons (the range of *taugtBy* ° *rank* binary relation is a subset of the denotation of TENURED-RANKS)

Descriptions can usually be expressed in standard first order predicate calculus, treating concepts as unary predicates and roles as binary ones. For the preceding example, the corresponding formula would be

$$\text{COURSE}(this) \wedge$$

$$(\exists x_1, \exists x_2, \ldots \exists x_{20})\ \text{participants}(this, x_1) \wedge \text{participants}(this, x_2) \wedge \ldots \wedge \text{participants}(this, x_{20})$$

$$\wedge$$

$$(x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge \ldots\ldots \wedge x_{19} \neq x_{20})$$

$$\wedge$$

$$\forall t \forall r\ \text{taught-by}(this, t) \wedge \text{rank}(t, r) \supset \text{TENURED-RANKS}(r)$$

This example shows that the encoding in predicate logic is much more verbose and complex, mostly because of the proliferation of variables and quantifiers. As a result it is more difficult to represent information in this notation and it is less readable for humans.

3.3.5 Semantics of DLs

It has been suggested that descriptions are terms and they are organized by the logic of subsumption and this logic is used in many different ways. If B and C are descriptions of sort "concept", then we want B to subsume C (written C $\Rightarrow$ B) when, in each possible world, every individual in the denotation of C is in the denotation of B; similarly, for descriptions denoting relationships, r $\Rightarrow_{role}$ s if every pair of objects related by r is also related by s (e.g., sons $\Rightarrow_{role}$ children). Given a description language, we thus always start by looking for the subsumption relation between concepts and between roles.

For any specific language, there are many different possible semantics, which can be presented in following different ways:

- Denotational Semantics: Define a function ***D*** that associates with every description B a set of values, ***D*** (B), in some domain. Then define C $\Rightarrow$ B iff ***D*** © is a subset obtained from the denotation of its subterms. For example,

$$\boldsymbol{D}\ (\textbf{and}\ (\alpha, \beta)) \equiv \boldsymbol{D}(\alpha) \cap \boldsymbol{D}(\beta)$$

Such a formal specification for DLs was introduced in [Brachmann and Levesque '84], and has been applied to most DLs that have been developed since then.

- Axiomatic semantics: Specify the $\Rightarrow$ relation by syntactic rules of inference. Natural deduction systems (especially as used in programming languages, e.g. [Khan '87]) seem particularly appropriate for this task, as illustrated by the following subsumption rule for **all** restrictions

$$\frac{\vdash C \Rightarrow B \qquad \vdash s \Rightarrow_{role} r}{\vdash \textbf{all}(r, C) \Rightarrow \textbf{all}(s, B)}$$

Which states that if the restriction on the role is stronger, and/or the role is more general, then the whole term is more specialized.

- Operational semantics: develop a program for computing the function subsumes? (B, C), and then declare that (C $\Rightarrow$ B) iff this program returns the value true. This is not an uncommon approach, and even systems such as Prolog have been described in this way. For DL-based systems, this approach is relatively unappealing because their implementation is quite complex when there are more than a few description constructors interacting in many, sometimes subtle ways [Borgida '92].

Given its origins in set containment, the *sine qua non* of the subsumption relation is that it be a pre-order, i.e. it should be reflexive and transitive.

The following requirements are to be kept in mind when defining subsumption (and hence the logical semantics of constructors):

The need to explain to other users of the language how things should be represented with it and how the system will reason with terms;

The desire to produce an implementation of the system. There should be reasons to use DLs as pure specification languages, but if one wants to build systems that perform reasoning with them, then one must have some ideas on how to implement them *in the way they were described to the user*.

3.3.6 Reasoning with Descriptions

There are two fundamental aspects to the logic of descriptions: the recognition of individuals that satisfy a description and detecting various judgments, such as subsumption that relate pairs of descriptions.

Since descriptions denote concepts or relationships it is natural to take their analogues in logic to be ordinary unary or binary predicates. Consider the following knowledge base of horn clauses:

ParentOf(mini, abu). Male(abu).

ParentOf(mini, julian). Male (julian).

ParentOf(mini, tulika). Female (tulika).

Child (_x): - ParentOf (_z, _x).

Son(_y) : - Male(_y), ParentOf(_w, _y.

Normally, such a system is used to deduce new properties of individuals, e.g., whether the son predicate "recognizes" the individual Abu. On the other hand one might want to reason entirely from intentional information — the rules — ignoring ground facts. For example, one might be interested in whether Child (_x) is implied by ("subsumes") Son (_x). Note that although one cannot express this question in Prolog, theoretically the answer would be "yes", because the last two clauses are in fact treated as the following definitions

$$\text{Child}(x) \Leftrightarrow (\exists z)\text{ParentOf}(z,x)$$

$$\text{Son}(y) \Leftrightarrow (\exists w)\text{Male}(y) \wedge \text{ParentOf}(w, y)$$

by the semantics of "predicate completion" in Prolog. However, if one takes seriously the rule for Son as its definition, then asserting Son(arjun) ought to allow us to deduce that Child(arjun) — a deduction not made in current logic programming systems. It is such reasoning with definitions that is the trademark of description logics.

3.3.6.1 Lattices: a foundation for reasoning with description

Since concept and role descriptions can easily be thought of as unary and binary predicates, it is natural to consider their ordering by the implication (subsumption) relationship, which can be denoted by the symbol $\Rightarrow$. Most existing DLs provide (nullary) term constructors' **thing** (the universal concept) and **nothing** (the inconsistent concept), as well as some constructors like **and**, whose semantics is that of set intersection. In addition to having been found useful in all applications, these constructors provide the important benefit of turning the (infinite) space of descriptions into a mathematical structure called a "meet semi-lattice", where every pair of descriptions B and C has a greatest lower bound — a description that subsumes any other description that is subsumed by both B and C — namely and (B, C). The DLs one encounters in practice are in fact lattices: it is always possible to find a unique least common subsumer for every pair of descriptions. For example, figure 2 presents the two descriptions and their least common subsumer in the language introduced in the previous section. For conditions under which the semi-lattice is guaranteed to be a lattice, one may consult [Cohen et al.'92]. The first investigations concerning theoretic-theoretic aspects of a special DL can be found in [Ait-kaci '84].

and(**prim**(Course),
  at-most(20, participants),
  all(taughtBy, one-of(gauss,Euclid)))
and(**prim**(COURSE), **prim**(FUNNY-EVENT),
  at-most(25, participants),
  all(taughtBy, **one-of**(gaus, Marx)))
and(**prim**(COURSE),
  at-most(25, participants),
  all(taughtBy, **one-of**(Gauss, Euclid,Marx)))

Fig. 3.2: Two descriptions and their least common subsumer.

In [Quantz '93] these theoretical issues have been summarized as follows "DLs are characterized by a particular stance towards the essentials of KR-formalisms — in order to call something a KR formalism:

1. it has to be a formal language in the sense that there is a formal specification of its syntax;
2. it has to have formal semantics which defines an entailment relation on formulae;
3. there have to be (efficient) algorithms computing entailment between formulae".

3.3.7 Basics of DL (fundamental expressions in DL)

In DL one differentiates between *terms* and *objects* as basic language entities from which three kinds of formulae can be formed: *definitions*, *descriptions* and *rules*. A definition has the form $t_n ::= t$ and expresses the fact that the name t_n is used as an abbreviation for the term t. A list of such definitions is often called terminology (hence the name Terminological Logics). All DL dialects provide two types of terms, namely *concepts* (unary predicates) and *roles* (binary predicates), but they differ with respect to the term-forming operators they contain. Common concept-forming operators are: conjunction (c_1 **and** c_2), disjunction (c_1 **or** c_2) and negation (**not**(c)) and as well as quantified restrictions [Quantz '92a] such as value restrictions ($\forall$r:c), which stipulate that all fillers for a role r must be of type c, or number restrictions ($\geq n$ r:c or $\leq n$ r:c), stating that there are at least or at most *n* role-fillers of type c for r. Role-forming operators besides conjunction, disjunction and negations are role composition ($r_1.r_2$), transitive closure(r^+), inverse roles(r^-) and domain or range restrictions ($_c|r$ or $r|_c$). In a description, an object is described as being an instance of concept (o :: c), or as being related to another object by a role(o_1 :: r:o_2). Rules have the form $c_1 \Rightarrow c_2$ and states that each instance of the concept c_1 is also an instance of the concept c_2.

3.3.8 Definition of Concepts and Roles

KL-ONE/DL is more than just a representational language, as it includes facilities for building, storing, querying etc. of the network. The main interest in KL-One/DLs is in its capabilities to explicitly represent conceptual information as a *structured inheritance network.*

KL-ONE/DL is primarily an epistemological level network, which provides the necessary primitives with which to describe and handle knowledge. The primitives are knowledge independent, in that they can be used to describe the internal structure of a broad spectrum of concepts. In all DLs, a distinction is made between concepts (intuitively denoting relationships between individuals/unary predicates) and roles (intuitively denoting relationships between individuals/binary predicates).

3.3.8.1 Concepts

basically, are used to describe objects, i.e. concepts *structure* a domain of individuals. There are several ways in which the structuring of a domain is desirable. The most basic one concerns the storing and retrieval of objects—if one defines a concept "book" one can use this concept to:

1. inform the system that a particular object is an instance of the concept "book";
2. let the system retrieve all known instances of the concept "book".

Storing and retrieving instances of a concept in this simple manner may not be interesting but things become interesting if the concepts are not only used as atomic labels but are rather semantically related to other concepts. The most basic way of relating concepts in a DL system is by building a conceptual hierarchy. Further, concepts correspond to conceptually primitive pieces of domain knowledge and are either *primitive concepts* or *defined concepts*. The easiest way of building such a hierarchy is to "define" a concept by specifying its super concepts (see example).

```
flexinit.
product    :< ctop.
vehicle          :< product.
car      :< vehicle.
coach:< vehicle
              truck    :< vehicle.
```

Primitive concepts:

are used for domain concepts that are atomic, i.e. have no internal structure, or that cannot be defined in terms of necessary and sufficient properties. However, primitive concepts can still specify necessary properties, though they may not be able to define all of them. For example, the generic concept for a natural kind such as "elephant" cannot be defined by necessary and sufficient properties, so it is primitive. Most generic concepts fall into the primitive category.

Defined concepts:

are built up from primitive concepts and other defined concepts and have their necessary and sufficient properties defined. For example consider the following definitions:

```
book  :<   publication
novel[1] :<    book and fiction.
scientific_book  :<  book  and scientific_publication.
```

Generic Concepts:

this is an intentional description of class domain objects e.g. person message, date etc., and are the most important type of concepts. Generic concepts are either primitive or are defined, using super concept links, in terms of other generic concepts. This creates a basic taxonomy formed of those concepts that subsume, or are subsumed by, other concepts. Such conceptual hierarchies form the basis of *inheritance*, the most fundamental inference capability of DL systems.

[1] Note that 'novel' is introduced as a primitive concept, whereas 'scientific_book' is a defined concept. The standard explanation of this distinction is that for primitive terms only necessary conditions are specified, whereas for defined terms necessary and sufficient conditions are specified.

3.3.8.2 Roles

The primitives used to represent the internal structure of a concept are *Roles*, which represent the attributes associated with the concept. Roles not only hold the information about the function of the attribute, i.e. the intention of the attribute, but also act as a description of the potential fillers, i.e. the extension, or instances, of the attribute. Whereas, concepts correspond to unary predicates, roles correspond to binary predicates/relations. Consequently, roles can be described by specifying the type of their first argument (DOMAIN) or their second argument (RANGE). Roles are often seen as properties of concepts or objects and not as independently existing entities. For example, consider the concept "publication". The meaningful properties of publications may be 'author', 'title' 'publication_year', etc.. One can thus introduce a role

has_author :< domain(publication).

This definition does not contain any restriction with respect to its range, i.e. the objects which are allowed as fillers for the role 'has_author'. Moreover, it is possible that a publication has more than one author:

chinese_room :: has_author:searle.

principia :: has_author:[russel, whitehead).

Thus, the domain information specified for the role 'has_author' is sufficient for the system to infer that 'principia' is a publication:

principia ?: publication.

Like concepts, roles also distinguish between primitive and defined. The roles, which have been inserted as defined, represent the necessary and sufficient conditions for the classification process. These roles can be filled with objects, thus representing the relationship between two objects. Let us assume that we have an object *apple*, which is an instance of a concept *foodstuff* and we have another object *jane* , which has been instantiated as *person*, in this way the object *apple* can fill the role *eats* for *jane*. We would have thus represented the following term: *"jane eats apple"*.

Further, there are operators to create the concept terms, which describe the restrictions of the roles. These are value and number restrictions. The later is also known as the cardinality restrictions**.**

Value Restriction (all (r, c)):

describes those objects whose fillers for role r are all instances of c. The following operator illustrates the all operator, where we say that "all the authors of the principia are famous, thus 'whitehead' immediately becomes famous too:

principia :: all(has_author, famous).

Whitehead :: famous.

Number Restrictions:

Another type of role restriction supported by the DL systems are number restrictions. The following example contains two definitions which use a minimum and a maximum restriction, respectively:

individual_publication := atmost(1,has_author).

Group_publication := atleast(3,has_author).

3.3.8.3 Descriptions

Specifying the *roles* that define a *concept* as well as the relationships that must exist between the role fillers have also to be specified. For example, the sent-date must be before the received-date, the recipient of a message needs to be the senders supervisor etc. This is a function of the structural descriptions. These relate to two or more roles in terms of another concept.

In descriptions, it is fixed that an object is either connected to an instance of a concept or with another concept. In BACK '::' is used for the descriptions and ':' is used to express a specific role has been assigned with a specific concept:

''Oliver'' :: father

has_Child : ''Jhon''.

the description can be interpreted as ''Oliver is a father and has one child with the name John''. From our definition of concept father one can deduce that Oliver as well as John have to be persons and Oliver is a male.

Note that terminological systems do not follow the closed world assumption. Thus, let us assume that we have assigned the concept of Oliver only a child John, thus a terminolgical system at this stage is in no position to infer that all the children of Oliver are males. To achieve this we have to tell the system explicitly that Oliver does not have any other children apart from John. This can be achieved by the following definition:

''Oliver'' :: atmost(1, has_Child).

In order to once again stress upon this expression and in addition to formulate a little complex description, we would like to describe a person, for whom it is known that all his male children are doctors, without saying it explicitly. This description shows the feature of DLs to deal with incomplete descriptions.

''Oliver'' :: father
and all(has_child and range(all(sex, male)),
all(profession, doctor)).

3.3.9 Rules

Terminological systems offer as other expressions a possibility in forward directed rules ($K_1 \Rightarrow K_2$). With the help of this, one can determine that every instance of a concept term in the the premises part of the rule (K_1) is also the instance of a concept on the conclusion side of the rule (K_2). The feature of such rules is that they fire only then, if an object has been found, which is an instance of a concept on the premises side of the rule.

This procedure has an effect on the tests of used rules. If one wants to check that they are working in a given manner, then one has to first create objects for all the premises. See example below:

conference :< ctop.
ai_conference :< conference with filter = topic_conf_type.
ai_conference :< domain(conference_article) and
range(conference) and feature.
the(at_conference, ai_conference) $\Rightarrow$ has_topic:artificial_intelligence.

In other words, whenever an object is subsumed by the left-hand side of a rule, its right-hand side is added to its description.

ijcai91 :: ai_conference.

tractable_dl :: at_conference:ijcai91.
tractable_dl ?: has_topic:artificial_intelligence.

3.3.10 Query Formation

In this section, we will describe what queries can be answered by a terminological system itself. Basically one differentiates here between 6 different types of queries. Here alphabet 't' is used for terms, 'c' for concepts, 'r' for roles and 'o' for objects. The different queries can be segmented according to the knowledge types they belong to. First of all, queries at concept level are described which explicitly deliver a specific truth value.

- t_1 ?: t_2
  Is term t_1 special to Term t_2, i.e. is t_1 subsumed by t_2?

Example:
>> father ?: person.
>> yes

- t_1 and t_2 ?< nothing

are two terms t_1 and t_2 disjunct?

Example:
<<father and atmost(0, has_child) ?< **nothing**.
>>Warning: incoherent concept
>>yes.
There are four different types of queries on the assertional side:

- o ?: c

is the object o an instance of concept c?

Example:
<< ''Oliver'' ?: father
>> yes.

- o1 ?: r: o2

are the two objects in relation to one another through a role, i.e. is the object o2 a rolefiller for role r in object o1.

Example:
<< ''Oliver'' ?: has_child : ''John''
>> yes

- o1 :: r : o2

if the above mentioned description is not accepted by the system, then it means that the above mentioned description is inconsistent with the presently existing terminology

Example:
<< ‘‘Marc’’ :: vegetarian and eats : ‘‘Meat’’
>> no

- getall(c)

which objects are instances of c?

Example:
<< getall(person)
>>[‘‘Oliver’’, ‘‘Jhon’’, ‘‘Sunil’’, ‘‘Marc’’]

3.3.11 Inferences in DL

There are numerous deductive inferences that DL provides**:**

3.3.11.1 Classification:

all concepts more general than a concept and all concepts more specific than a concept are found.

- **The task of classification:** one of the main capabilities of the description logics is to automatically determine the position of a new concept within the given concept hierarchy. The portion of the knowledge representation system which fulfills this task is known as the classifier. Classification is based on the notion of the subsumption of concepts. Classification is an operation which is performed within a taxonomic system in order to support the computation of subsumption.

when C and C’ are concepts (for discussion, unary predicates defined in logical language) such that in every model the extension of C is a superset of the extension C’ i.e. every instance of C’ is an instance of C, then one can say that C subsumes C’.

Example:

```
Husband  := male_person
  and exactly(1, hasWife).

Parent := male_person
  and exactly(1, hasWife)
  and some(hasChildren).
```

From the above example, it is clear that every *Parent* has to be a *Husband* as well . All objects that are instances of the class (of concepts) Father inevitabely fulfill the necessary and sufficient conditions which have been laid down for the concept *Husband*.

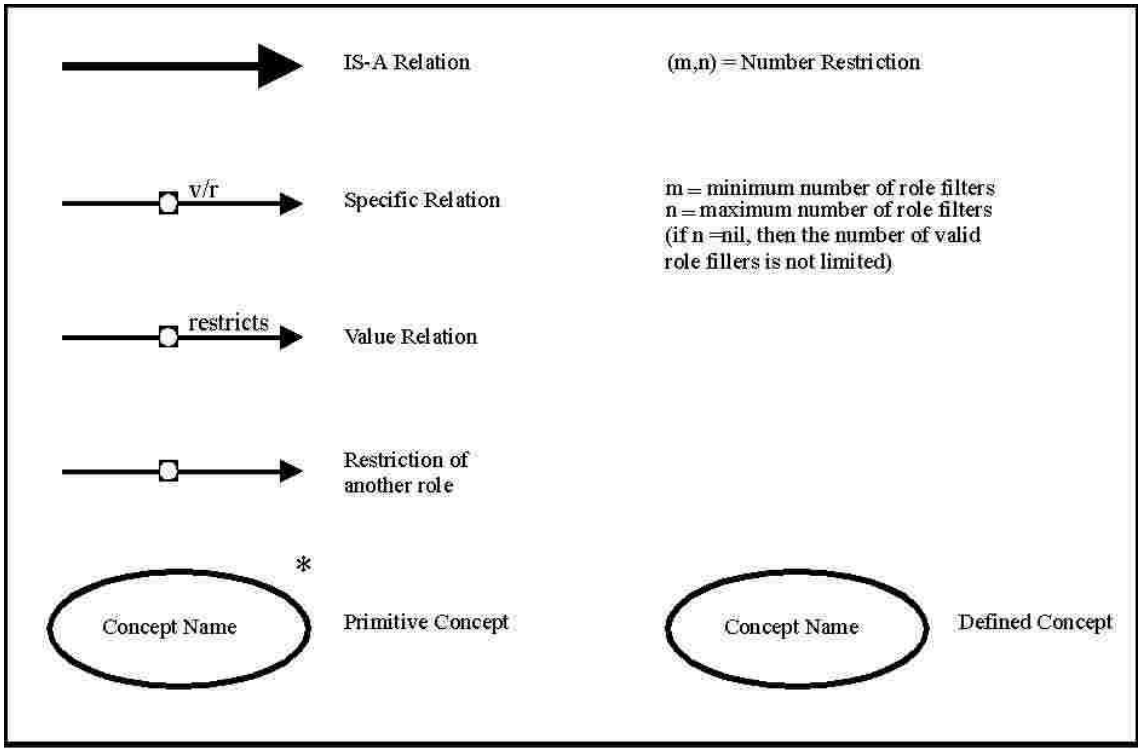


Fig. 3.3: meaning of the used symbols in the graphical representation

Indeed, the above mentioned example shows the efficiency of classification. The classification establishes the implicit relation between a concept and a given terminology. Due to this process a system is in a position to determine independently the order of the concepts, without ever being explicitly formulated, that a parent is a *husband* (Fig. 3.4). It concerns here a semantically well founded super-subclass relation.

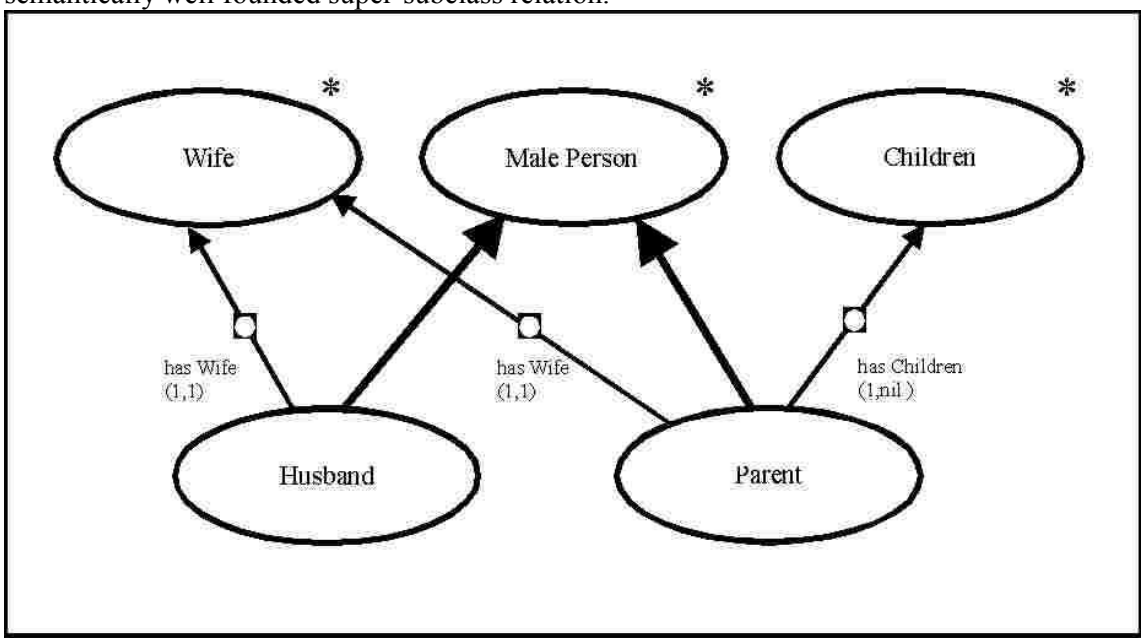


Fig. 3.4: concept hierarchy before the classification

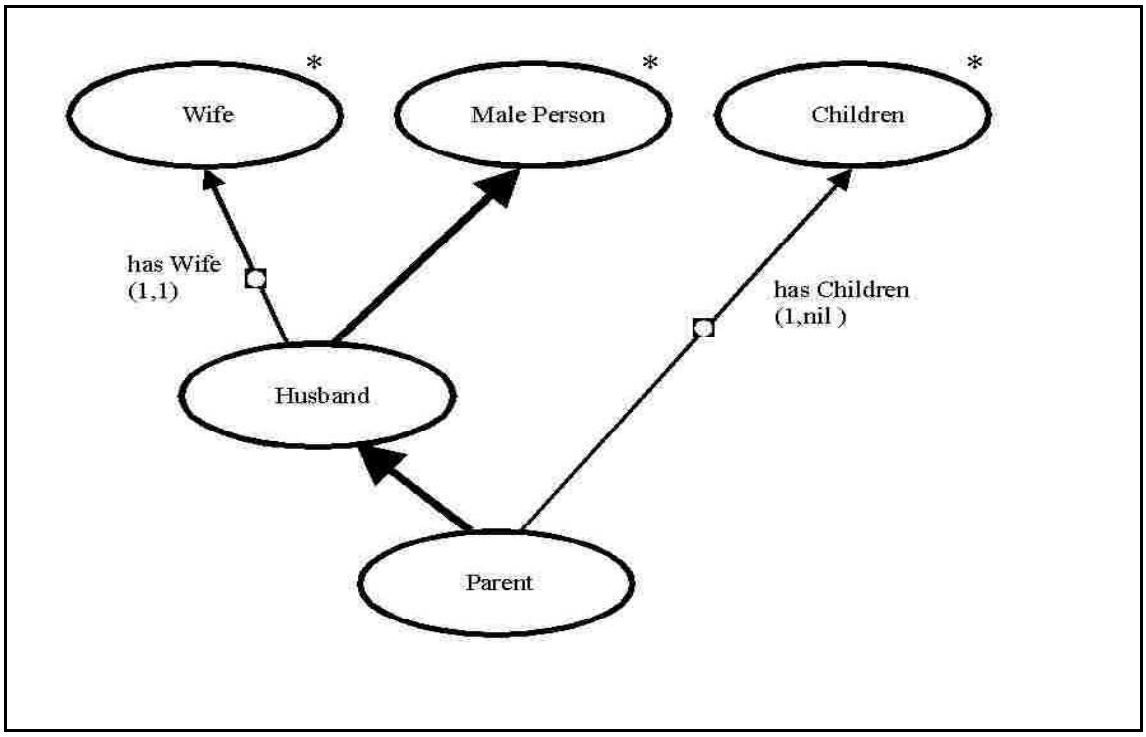


Fig. 3.5: concept hierarcy after the classification

During the classification subsumption relations (at least in BACK) are also determined, which can be formulated by the rules (I-Links). Thus, the following description will also lead to the same partial subsumption order as in Fig. 3.5

Example:

Parent := male_person
and exactly(1, hasWife).

Parent => some (hasChildren).

The classification performs several important functions. For example, it detects redundancies, keeps the hierarchy coherent and recognizes inconsistent and vacuous definitions, speed up the query answering and finally it recognizes concepts by their features.

3.3.11.2 Subsumption

Questions about whether or not one concept is more general than another concept are answered.

- **The task of subsumption:** determining subsumption is the fundamental service of a taxonomic reasoner (DL): given two descriptions δ1 and δ2 , asking whether δ2 subsumes δ1 means to determine the set of individuals represented by δ1 is a subset of those represented by δ2. On the basis of subsumption, several other kinds of questions can be answered, by turning them into subsumption question:

δ1 **is** δ2 Subsumption
(δ1 **and** δ2) **is Nothing** Disjointness
δ **is Nothing** Existence
(**Any** =i **such_that** δ) Retrieval
=x **such_that** δ **is** =x Describe

Here, we have used the notation of Omega [Attardi & Simi '81], where the operator **and** joins the two descriptions and **nothing** corresponds to the empty description. Thus, asking whether the union of two descriptions is empty corresponds to asking whether they are disjointed. Similarly, for retrieval, we exploit the Omega construct Any, whose semantics requires that the variable = i ranges over the individuals of the interpretation domain. The describe operation means to find out whatever is known about a certain description.

3.3.11.3 Inheritance

restrictions that apply to instances of a concept must also apply to instances of specialization of that concept; in a sense, then, properties are "inherited" by more specific concepts from those that they specialize in

- **The task of inheritance:** In many systems inheritance basically means the possibility of migrating and transferring slots or attributes from one class to another. Terminological logics generalize inheritance formalisms in that inheritance queries can be translated into subsumption queries in the following way: to determine if an object or concept inherits a property, e.g., to see if Clyde has three legs, determine if the object or concept is subsumed by a concept expressing the property, e.g., see if Clyde is subsumed by the concept of three-legged objects. However, subsumption is more powerful than inheritance in that combination of different properties may be used to answer subsumption queries.

A second issue for terminological logics is computational tractability. One reason for limiting expressive power was an attempt to create useful formalisms that were computationally tractable in the worst case. Unfortunately, reasonably expressive terminological logics have undecidable subsumptions [Patel-Schneider '89], so this goal can't be realized. Given this negative result, effective use of terminological logics in KR systems will require the development of useful partial subsumption algorithms, perhaps derived from non-standard semantics.

The most important role for inheritance in TL is to provide a minimal partial subsumption algorithm - any subsumption algorithm for TLs should include those subsumptions that are the result of inheritance. These sorts of subsumptions, where the pieces of the general concept must show up in the more specific, is called structural subsumption [Patel-Schneider '89].

4 Sales Advisory

4.1 Introduction

The process of sales advisory is concerned with the design and operation of interactions that take place among the vendor (expert) and the customer while selling a technically complex end product. Roughly speaking , sales advisory process comprises of different stages according to the sales cycle and usually occurs through a series of meetings, conversations and interventions. In effect the process is a loop, but one that need not to be completed before it begins again.

On the other hand, the days when a salesperson could carry the company catalog around in his or her head have disappeared. From high-tech to low-tech industries, today's salesperson often represents thousands of products available in countless permutations. To help overcome this situation many companies are rushing to market with information technology to aid millions of salespeople worldwide. But according to analysts, these systems are destined to fail. Why? Because they focus only on improving efficiency rather than on increasing effectiveness of the selling process.

In this chapter we provide the reader with the insights of the domain of sales advisory and try to answer certain questions regarding the process of sales advisory and the need to provide computer aid to make this process more effective. Here, the aim is to help the sales person by providing him an electronic assistant (computer) during the sales advisory task (i.e. while he/she is involved in a consultation dialog). The sales person's job is to understand the requirements/specification of the customer and then accordingly suggest a product, its alternatives and a comparative analysis of suggested product with its alternatives. The first section of this chapter deals with the rationale. Here, we try to answer such questions like, why the presently deployed sales and marketing management systems are not sufficient to fulfill the present day's demands; Why the computer aid is unavoidable in today's ever changing product development and financial market situations; Finally, why there is a need to develop the knowledge based sales advisory (consultation) systems.

Section 4.2 provides us with the *state of the art* consultation systems and in section 4.3 (*Aspects of Consultation*), we try to define Consultation in general as a communicative situation and its implications to a Computer Aided Sales Advisory. Furthermore, we also try to define what all - e.g. agents, their functions their roles, task distribution among them, cooperation & communication among them and its type, and finally the medium of communication among them - is involved in the process of sales consultation. Thereafter, we will try to define a sensible division of labor between the human and the machine involved in this process (of Sales Advisory/Consultation).

The main thesis of Section 4.4 is that the task of sales advisory in the new (experience) economy has to be a memorable buying experience and how can one turn a sales advisory task into a memorable experience. In an answer to that we argue that the task of sales advisory is a process based on interactions and the interactions can be designed in such a way that they finally turn into an experience.

The final section of this chapter provide the design criteria, general principles of the design and special requirements for representing consultation knowledge.

4.1.1 Rationale

Now, we aim to help the sales person by providing him an electronic assistant (computer) during the sales advisory task (i.e. while he/she is involved in a consultation dialog). The sales person's job is to understand the requirements/specification of the customer and then accordingly suggest a product, its alternatives and a comparative analysis of suggested product with its alternatives.

Traditionally, the process of sales advisory takes place orally and with the help of many documents and manuals. The efficiency of the advisory process is influenced due to manual search of the documents for a necessary product relevant information needed to advice the customer. State of the art technology in this field offers lots of solutions, either by providing the sales person with an access to product info base or simply a data base where all the product catalogues are stored. Now, may be the product relevant information is digitized but the sales people seldom make use of such facilities or totally reject such aids.

Why is it so? The answer to this question is two fold. Firstly, all the conventional aids provided to sales force do not take their end users and the environment in which these systems are to be installed into account (engineering viewpoint). Secondly, the systems which claim to have considered the end user (social viewpoint) in their design make compromises at the engineering level i.e. the quality of information content (representation of knowledge and its retrieval thereafter) in their systems is poor, which obviously leads to low performance. An ideal situation would be to develop an aid to support the sales force which is based upon the blend of the two above mentioned approaches. That means an application which is strong on the engineering side as well as the cognitive side and also considers the social environment of the complete system at the same time (socio-cognitive engineering viewpoint).

Now, The task is to develop an electronic aid/assistant - for the sales person - which improves upon not only the quality of advice but also makes this process more efficient.

Why have researchers not done this till today? Of course, that has already been done before but with less success and why were they not so successful? Because the approach adopted to tackle this problem was always the same, namely to automate the complete process of sales advisory using the over simplified model of consultation and developing the systems solely based upon either of the two above mentioned views (Engineering- or Social-) and neglecting completely the existence of other model.

Another reason why the sales advisory systems developed today are less successful is that the technology used to develop these systems is not sufficient to achieve the desired results. For example, an automobile sales person - during the sales dialog with a customer - is confronted with such questions:

Q1. What all BasicTruckType(BTT) have 5000kg Load Capacity?
Q2. Give a BTT which has Strong Motor (LDM) & 5000kg Load Capacity?
Q3. Can Fuel System = FSxxx be fitted with BTT = XXX?

Q4. Get Total Price of BTT = Dxxxx with ACC = FIS3000T & ESU7000 & RS5T-XT & LDM5000

Q5. List Accessories available after Date = 3.5.92 and only for BTT = D5000 ?

Q6. Is Configured BTT = XXXX With ACC = xxxx & yyyy & zzzz Manufacturable?

Q7. Is there any Discount Available on BTT = D5000?

Q8. Is there any Discount if a BTT is bought on credit?

Q9. Calculate VAT on BTT = D5000 with ACC = xxx & yyy & zzz?

A close look at above mentioned queries shows that as the queries become more complex, they begin to exceed the capabilities of present technologies such as databases (relational or object oriented) and rule based expert systems. Why can't these queries be answered? Because conventional databases do not possess the necessary deductive power and rule base systems although possess the deductive power but pose problems in keeping them consistent all the time and further lack the descriptive power required to represent the necessary knowledge.

Now, what is required is a hybrid knowledge representation mechanism which not only overcomes the above mentioned problems but also fulfills the following two demands: First, the need for representation tools equipped with a clean and well-defined semantics at the level of "what is to be represented" rather than "how it is to be represented". Second, providing these representational facilities in a computationally efficient manner.

In response to these demands, research in KR led to the development of terminological representation languages as an attempt to fill the gap between general purpose theorem provers which are semantically well-founded though extremely inefficient, and commercially available KR tools, which are more on the efficient side, but lack a semantic foundation [Schmiedel'88]. For these reasons we decided for a hybrid knowledge representation system (say BACK/FLEX/CLASSIC) based on the terminological representation language.

4.1.2 Domain/Scenario

A customer presently owns a small transport/travel company and is close to exceeding the load/volume capacity of his present fleet. He can increase his load/volume capacity by either:

1. Buying/Adding new trucks /coaches or
2. Leasing some more trucks/coaches or
3. A sensible combination of two or
4. Reorganizing his present activities + Buying/Leasing new trucks/coaches

Considering the above four situations, if he decides to buy trucks/coaches then the cost of buying will probably be more than the benefits and if he leases them, in the long run it will probably be an expensive solution; a combination of the two is also not so optimal. So what should he do? Choose the fourth option; here again he would not be able to produce an optimal mix on his own. In this situation what he requires is an able consultant.

If he walked into a local sales office of a leading Truck/Coach manufacturer, he would be received by a friendly sales representative who is ready to advise him on his current problem. Once, they have settled themselves down in a comfortable corner then the sales representative proceeds systematically; asks the customer for his/her requirements and provide him/her the solutions.

4.1.3 Situation

Currently the system - used for supporting a sales person in an above mentioned scenario - is completely integrated to the sales transactions. Along Order-Form generation and verification one can determine the delivery date, calculate the price and Order confirmation & Order Slips can also be generated. Further interfaces are provided to production planning and material disposition. In short the functionality of the system is comparable to the classical example of configuration system XSEL & R1/XCON [McDermott'82a,b].

4.1.4 What does our KB consists of?

Before we give the actual representation of the knowledge in the chosen knowledge representation system BACK, we would like to answer the question: what exactly does our KB consists of?

Our KB is concerned with storing knowledge about a domain which represents a product, different product versions and models, all the required accessories, availability & prices of accessories, accessories compatibility with each other and restrictions or constraints on the accessories which can be ordered with one particular type or model or version of the product.

All this information is available to a sales person in the form of a sales manual (in the case of LTV, 3 sales manuals each approximately 1000 pages long). All these manuals are organized in such a way that a sales person first selects a type and model or a version of the specific product along with its factory-fitted accessories. Thereafter, the sales person's job is to include accessories according to customer specifications from a list of all the available accessories, which are recommended for this particular/specific product version/model/type. While doing this job the sales person is required to consider the following types of restrictions/constraints:

- A particular type of accessory can only be ordered for a particular type/model/version of the product.
- A particular type of accessory cannot be ordered for a particular type/model/version of the product.
- A particular type of accessory can only be ordered if another specific accessory (Pre-requisite accessory) has already been ordered.
- A particular type of accessory cannot be ordered because it is not compatible with the specific accessory which has already been ordered.(violation of pre-condition)
- A particular type of accessory in combination with other accessories may have either different prices or are offered free of cost.
- Exclusive OR accessories mean only one would be sufficient even though there are several possibilities
- A particular accessory may include/consist of certain other separately available accessories

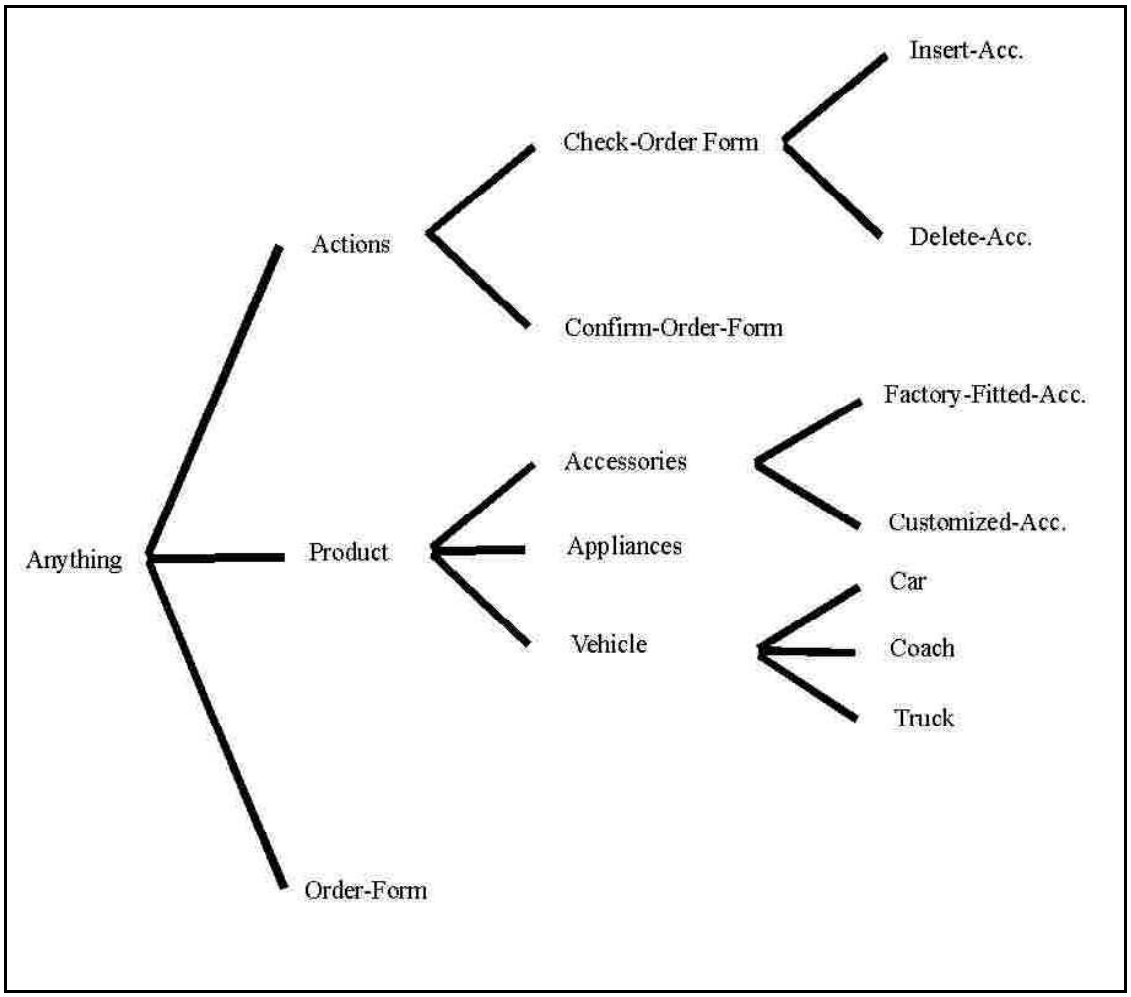


Fig. 4.1: Top three levels of SAS-BACK KB

Fig. 4.1 shows the root of the SAS-BACK (Sales Advisory System in BACK) taxonomy (ANYTHING) and two levels immediately below it. The two principle object types of concern in our domain are PRODUCT and ACCESSORIES. The edges of the taxonomy have their usual "IS-A" interpretation. The PRODUCT describes ANYTHING manufactured by a company. Nodes below PRODUCT and ACCESSORIES represent a particular product and the accessories which come along with this specific product. The relationship between the two is captured by various slot-filler relationships.

The above mentioned restrictions or constraints are represented through the I-links (Implication-links) between the models and the accessories or within themselves. Here is an example of a typical constraint:

If model Z has been selected only Then Accessory X can be ordered

or

If model Z has been selected Then Accessory X cannot be ordered

or

If Accessory Z has been selected Then Accessory X can be ordered

or

If Accessory Z has been selected Then Accessory X can't be ordered

This can be represented in our representational scheme BACK as follows:

all(hasAcc, Z) -> model Z.

and a Concept with its Roles in BACK is defined as follows:

Bm673 :< LKW and atleast(1, hatAusfuehrung)
and atleast(1, hatMotor)
and atmost(1, hatMotor)
.
.
.
and atleast(1, PreisFgMitaufbau).

hatAusfuehrung :< domain(lkw) and range(ausfuehrung)

whereas Ausfuehrung is further defined as

Ausfuehrung := attribute_domain([k,s,l,o,ol,ls]).

The natural language interpretation of the above mentioned concept is:

"Production pattern type Bm673 is a product which is of a particular version where the version is from the list of versions k, s, l, etc. and has at least and at most one motor and so on".

4.2 Computer Systems used in Sales Organization

This section deals with state of the art consultation systems. Here, we try to answer such questions as, why the presently deployed sales and marketing management systems are not sufficient to fulfill present day demands, why the computer aid is unavoidable in today's ever changing product development and financial market situations and finally, why there is a need to develop the knowledge based sales advisory (consultation) systems. In this section we provide a list of systems which are being used in sales organizations to support various functions and the reasons why are they not sufficient to fulfill present day demands.

- *Databases* are being used in sales organization to store customer, product, production and inventory data required to support different kind of sales functions.

- *Information Systems* today are being used for the improvement of delivery service, after sales service, acceleration of contract transactions etc.

- *Know-How Databases* could be understood as the ordered collection of past solutions prepared for customer specific problems.

- *Technical Drawings and Document Management Systems* serve to store and retrieve the graphics, sketches and texts which are either created by either a CAD system or manually.

- *Electronic Product Catalogue* systems are in their core selection systems based upon the end products and different variants are represented explicitly.

- *Technical Configurators* can be termed as systems which support the selection process, where, after considering the specifications and wishes of the customer, a complex end product is finally put together.

An intensive and sufficiently lengthy survey of presently available marketing and sales management systems provides us with enough material to conclude that due to today's ever changing product development and financial market situations, computer aids are unavoidable & the presently deployed sales and marketing management systems are not sufficient to fulfill present day demands. In addition, our analysis came to following conclusions:

All the conventional EDP systems used in present day marketing are passive support systems which are designed to support one particular function of the whole marketing department e.g. calendar system to manage the appointments of sales people and text processing systems to write letters etc.

The systems are passive because they just store the information and process the information in a given fixed way and do not posses any knowledge about when and how to make use of this stored knowledge. This function of knowing exactly what information/knowledge is required in what situation, where to find this knowledge and how to access it is left to the knowledge. That means only an experienced sales person can find out the right information at right time and in right place.

None of the systems used supports a specific process such as sales consultation. In other words no system directly supports the sales person during the actual sales consultation session with a customer.

During the sales consultation sessions it is not always the technical features of the product - which one can represent with the electronic product catalogues or configurators – which are in lime-light but questions such as how this specific product is going to behave in a specific application area also arise.

4.2.1 State of the Art: Consultation/Advisory Systems

Having consulted literature in German and in English, from business management to computer science and from marketing and sales management to artificial intelligence, I have not come across a single system (used in the marketing field) which considers its environment (i.e. the place where it is ultimately going to be installed and its surroundings). What is more,

most of the systems installed did not even consider their end users while designing the systems. Furthermore, there are very few systems in the literature which are termed as "sales advisory" or "sales consultation" systems and after a careful study of these systems, one comes to the conclusion that they were not essentially designed for the task of "consultation" but for something else, i.e. mostly for helping the technical departments in checking the order-forms for technical configuration and verifying the manufacturability of the technically complex end product. A classic example of this is R1/XCON [Mcdermott ’82b] and XSEL [McDermott ’82a].

Another point which one notices after a detailed analysis of these systems is that all these consultation systems have an over simplified model of "consultation" (i.e. Technical Configuration) as the basis of their design. One reason for such an over simplified model is that until today, designers and/or developers of such systems neither considered the potential end users of such systems nor did they take the environment (i.e. the place or circumstances under which such systems are to be used) into their design considerations. Such vital questions as who exactly is going to be the end user of the systems and what their requirements under the specific environments are, were never considered. Thus, such systems had their failure programmed into them from their conception and their consequent development. In the recent past, alarmed by failure ridden "ZUSTAND/State", scientists started looking for explanations and with their controversial book Winograd and Flores [Winograd and Flores ’86] came up with not only explanations but also some suggestions for new design principles based upon their philosophy of language & action. They refuted the present design principles and challenged the underlying philosophy of artificial Intelligence.

”The task we have undertaken in this book is to challenge the rationalistic tradition, introducing an alternative orientation that can lead to asking new questions. In developing this new orientation, we were lead to a critique of the current mythology of artificial intelligence and it’s related cognitive theories, drawing conclusions that contradict the naive optimism..... Our ultimate goal, however, is not a debunking but a redirection. The alternative we pose is not a position in a debate about whether or not computers will be intelligent, but an attempt to create a new understanding of how to design computer tools suited to human use and human purposes.”

[Winograd and Flores ’86]

This direction influenced our work greatly. At this point, rather than going into the details of these new design principles or the philosophy, we would very much like to give the reader a short and crisp excursus of the state of the art in the field of sales advisory and sales consultation systems. Thus the following section takes us through the series of sales advisory/consultation systems starting from Xcon/Xsel to presently available or mentioned in the literature.

4.2.2 Advisory/Consultation Systems Aavailable

This section surveys the application of knowledge-based systems to the area of sales advisory/consultation. This survey is by no means comprehensive. One of the major difficulties in surveying sales advisory application/consultation is the secrecy surrounding them. All the major sales organizations are reluctant to speak about their technological activities in enhancing any of their capabilities as they consider it to be a competitive tool, thus they want to conserve this advantage for as long as possible. This, however, is changing, since companies are now using this technology to build an image of a progressive company in their marketing efforts.

The main sources of this survey are technical reports and articles published in different magazines, conference proceedings and trade journals. These sources naturally narrow down the application to a few well-known references but give a fair picture of activities taking place in this particular field.

In systematizing the information, it was extremely difficult to categorize/classify presently available sales advisory/consultations systems in a particular scheme. Thus, we decided to give the readers, a glimpse of a wide horizon of these systems in next sub-section, and allow them to decide for how to classify/categorize them.

Financial Advisor (Palladian Inc. USA) helps managers to make capital investment decisions using a wide range of financial techniques, it explains how it arrives at each step in an analysis. Like an expert consultant, the system tests every input assumption and conclusion against an extensive knowledge of general business practices such as the users company policies, its accounting and management practices, its historical performance, and its competition [Turban '89].

Portfolio Management Advisor (Athena Group USA) is an expert system which assists the portfolio managers in construction and maintenance of investment portfolios. The system performs portfolio analysis by making use of specific investment information provided by the portfolio manager, and internal investment analysis and portfolio construction heuristics, along with rules for implementing modern portfolio theory both quantitatively and qualitatively. The system evaluates all potential equity investments and their combinations with regard to value, risk and the investment goals and constraints. This process is facilitated by analytical models. The expert system performs asset selection, asset allocation and portfolio construction.

Le Courtier (developed by Cognitive Systems Inc. for Belgium's largest bank, the Generale de Banque), another security portfolio advisory system, provides high-quality investment advice to customers of the bank. The system, which accepts both conversational French and English input, offers specific recommendations about stock purchases and portfolio distribution and also answers factual questions about the Belgian stock market. Le Courtier can give detailed investment advice only if it has a detailed financial profile of the customer. It gives the customer the option of completing a brief or an in-depth financial statement questionnaire, including current investments, assets and liabilities, and cash available for investment. From this information, Le Courtier may make stock portfolio recommendations or it may determine that the customer's assets are not large enough to warrant investments in the stock market. the customer may, however, override the system's recommendations.
The Athena group's system was designed for fund managers, Le Courtier is designed to be used directly by the banks customers. Le Courtier differs from most other knowledge-based systems in the way it puts emphasis on natural language conversations and combination of this with videotext.[Turban '89].

The **IF-FX** (Intelligent Forecaster-Foreign Exchange) from Data Logic in Great Britain provides advice on sterling/dollar market in the form of buy/sell recommendations based on chart forecasting techniques.

The **Foreign Exchange Advisory System** from the Athena group provides advice for both the initial development of a strategy in foreign currency option trading, as well as suggestions for

future modifications to the option strategy as a result of currency price movements in the market-place. The system provides recommendations for a foreign currency option trading strategy based upon market outlook, expected price movements, price volatility and the investors risk profile.

APEX (Applied Expert Systems, Cambridge, MA) is a client profiling system. It provides salesmen in the field with the expertise and analytical skills needed to sell an expanding mix of financial products effectively and uses expert systems technology to develop financial strategies for the clients. With the system, financial services firms can:

- understand client needs and match specific product offerings to those needs;
- provide tangible value to clients by offering advice that addresses their expressed needs;
- gather client information to improve customer targeting and product development.

The APEX client profiling system analyses information regarding the customer's current financial profile - income, assets, and liabilities, in the context of his or her expressed priorities and attitudes. The system includes both a knowledge base and a product base.

Knowledge-Based Sales Support to Timber Products Brokers was developed as a knowledge-based system to support the work of sales representatives for a lumber broker. The system provides the traders the necessary fright routings and applicable rates interactively to serve the needs of their customers. PacTim's (Pacific Timber Products Brokers Inc.) sales staff buy large lots of lumber and other wood products from mills in the Northwest and sell smaller lots of these products to nationwide network of customers and clients. PacTim buys from hundreds of mills and sells to thousands of customers. It is an information intensive business. brokers must locate available products, assess the relative scarcity or overabundance of these products compared to market demand, and find buyers. Product prices are volatile and the large number of producing mills makes information about product availability both difficult to obtain and valuable [Tamura et al. '87/88].

Sales Edge provides a method of comparing the psychological attributes of sales person with those of particular prospect to determine the best way to approach the sale and conduct the close. This software can be used prior to or after the interview [Collins '84].

MasterSELLER, a program developed by Rubash and associates, is designed to assimilate product knowledge and master salesperson knowledge into the knowledge base to create a "best fit" between the customer and the product. The MasterSELLER is designed to be used before, during and after the interview [Rubash et al. '87].

Xcon/Xsel: One of the most successful commercial expert system, configures VAX computer systems for Digital Equipment Corporation (Digital/DEC). No survey is complete without mentioning Xcon/Xsel. XCON, eXpert CONfiguration program that checks the accuarcy of complex computer systems configurations before the configuration information is fed into manufacturing and shiping schedules, to ensure that the system can be assembled and will function as designed — that no vital parts are omitted, no incompatible components paired , etc.. During the development of XCON, Digital recognized that many potentially disruptive configuration errors originate early in the order process. Thus conceived of an interactive sales-force aid (XSEL) that would guide and check designs as sales representatives originated them in response to customers particular needs. The configuration that emerged from this process could then be submitted in batch form to XCON for a final, detailed check before submission to manufacturing [Leonard-Barton'87]. Both these systems are written in VAX

OPS5, hence both are forward-chaining rule-based systems and use pattern-matching approach to search rules[Harmon'89].

4.2.2.1 UNIX[1] Consultant (UC)

UC is a natural language help facility which advises users in using the UNIX operating system. UC allows the user to engage in natural language dialogues with the operating system. The user can: query UC about how to do things in UNIX; ask about command names and formats; receive online definitions of UNIX or general operating systems terminology; and get help debugging problems in using UNIX commands [Wilensky et al. 84). UC is comprised of the following components: a language analyzer and generator, a context and memory model, an experimental common-sense planner, highly extensible knowledge bases on both the UNIX domain and English language, a goal analysis component, and a system for acquisition of new knowledge through instruction in English. The language interface of UC is based on a "phrasal analysis" approach which integrates semantic, grammatical and other types of information. In addition it includes capabilities for ellipsis resolution and reference disambiguation.

Knowledge Representation

UC uses a frame like representation where some of the contents are based on Schank's conceptual dependencies. The knowledge structures are stored in PEARL (Package for Efficient Access to Representation in LISP) databases [Deering et al. '81 & '82]. PEARL incorporates such standard features as automatic inheritance and various demon facilities. In addition, PEARL has a flexible indexing structure which allows the user to advise it about how to store facts to take advantage of how they are likely to be used.

Inference Mechanism

Although computational methods of inference are undoubtedly still needed and desired, the premise taken in UC is that most common everyday inferences made by humans are not computational but rather fall out of the structure of memory [Chin '83]. Thus, these can be modeled, as in UC, through the appropriate design of knowledge representation.

UC uses the hash indexing provided by PEARL databases to simulate associative access as it lacks the true associative memory. Frames stored in PEARL databases are indexed by combinations of the frame type and/or the contents of selected slots. For example, planfors provide immediate links between plans and their effects, out-planfors connect common queries to their generator ready outputs, and memory associations provide a general mechanism for associative links of almost any kind.

Interactions/User interface

The primary natural language processing in UC is done by an extended version of PHRAN (PHRasal ANalyzer) and PHRED (PHRasal English Diction) [Wilensky '87]. PHRAN reads sentences in English and produces representations that denote their meanings; PHRED takes representations for ideas to be expressed and produces natural language utterances that convey these ideas. These programs represent the very front and back ends of the interface respectively.

Knowledge Acquisition

A UC Teacher, which enables UC to learn new vocabulary and new facts about UNIX by being instructed in natural language. This was made possible due to the highly declarative knowledge representation of UC.

4.2.2.2 FAME

The project for Financial Marketing Expertise (FAME) was to produce a system that addresses the area usually referred to as financial marketing. This term characterizes the financial services of such large scale that they can significantly impact a company's financial status. for instance, a customer interested in buying computing technology on a large scale is usually concerned that the financing plan being used to acquire the technology is safe, sound, and attractive from a financial investment point of view. Therefore, in making very large sales, financial considerations often become as important as the computing considerations [Kastner et al. '86].

Human financial marketing experts, rather than exhaustively generating an optimal plan (which might not even exist), use their experimental heuristics and domain knowledge to prune the generate-and-test space for efficiently designing a plan that is attractive from the customer's viewpoint. Another imortant distinction is that a typical financial marketing problem frequently has no one solution. the importance lies not only in the answer you provide but also in the explanation and justification that you use to back the answer.

The system mainly consists of a set of relatively independent OPS5 rule groups with supporting Lisp functions for efficient database manipulation, graphics handling, and some numerical calculations. All of the system's control and inference (heuristically guided generate-and-test problem solving) process are encoded in rules. This makes the system quite flexible

Each rule group contains its own computational expertise and communicates with other groups through a set of protocols. In the simplest case, communication between rule groups is accomplished by simply sharing internal memory. For other communication, a rule group generates the data expected by the next group and then creates the group's task, thereby transferring control.

Finally, For an automated financial marketing problem solver to generate convincing arguments, it is crucial to accurately determine what concerns a potential customer and then to use these concerns in the plan and justification generation.

4.2.2.3 ENS

The Expert Network Selector (ENS) is a knowledge-based system designed to assist sales representatives to select, configure and cost customer data network configurations in an integrated manner [Ferguson et al. '87]. In particular, ENS gives advice on design and sales of customer networking and communications facilities. It uses a multi-paradigm knowledge representation scheme to define the various types of knowledge and information typically handled by communications networks sales staff.

The ENS has been provided with multiple entry points, which gives its users the flexibility to use ENS for different purposes at different times; e.g. to access rate information directly, to analyze the effect of rate changes on an existing customer configuration etc.. Further, for efficiency purposes ENS has been divided into following 4 discrete stages:

1. Information gathering by the system to define or edit customer network descriptions (e.g. the system asks for network topology, response time requirements, usage characteristics, data traffic, etc.).

2. Selection (with explanation), using both technical and non-technical selection criteria, of all viable services meeting the customer's requirements, and exclusion (with explanation) of all non-viable services.
3. Configuration at a customer access level of the selected solutions.
4. Production of a detailed quotation for these solutions.

Knowledge Representation
Knowledge in ENS is represented as both frames and rules, with Prolog as the chosen Implementation. The representation scheme chosen for ENS has following main categories [Ferguson et al. '87 b]:

- a hierarchy of frames to represent the customer's network requirements,
- a rule base of design/selection criteria to generate partial designs from these requirements, and
- frame-like design plans which are used to generate and represent fully-configured solutions.

The first category of knowledge represents the customer's network requirements. Here, ENS views a network as consisting of a hierarchy of sub-network design components. Networks are thus represented as a collection of customer applications (e.g. process control, remote job entry, file transfer) running on a set of network links. Further, each link has been described by its two node locations, where each node is further described by the type of termination equipment it represents. Finally, each of these design components is represented as a frame whose slot values are obtained either from the user or from stored defaults.

The second category of knowledge consists of service-specific design and selection criteria and is represented in the form of rules . Each technical, qualitative, or heuristic criterion is composed of two parts: a Design-Task and a Constraint-Set. The Design-Task generates a minimal set of partial design data which is further used to determine which network carrier services are viable and which are not. The rejections or recommendations are stored in special slots of the customer application frames, which are later used by the interface module to explain the reasoning behind the decisions made. The Constraint-Set is a set of design constraints which are provided by the domain expert and are ordered according to the domain expert's heuristics. These constraints are activated each time a Design-Task has finished executing. These constraints are viewed as having two sub-parts: a Boolean operation which is performed on a specific set of design parameters appearing in the associated Design-Task, and an instruction on what to do should this Boolean operation fail.

The third category of knowledge consists of a library of frame-like design plans. These design plans have two basic parts to them: a props slot that indicates which sub-link components originating from the partial design produced thus far will take part in the configured solution, and an Event-Sequence method that will dictate the sequence in which these components actually get configured.

Inference Mechanism
ENS's problem solving strategy is based on partial design plan generation with constraint propagation. Here, the total design space for a given application is divided into partial design stages, with either technical or heuristic constraints being applied to the appropriate design parameters at each of these stages. These constraints can be regarded as heuristics whose role is to direct the search for the associated design parameter which, upon re-calculating, will

most likely ensure the constraint is not violated a second time. This approach is somewhat similar to the advice mechanism described by Mittal and Araya in [Mittal and Arya ’86].

User Interface
Due to its highly interactive nature and consistent with the familiar computing environment of its users, ENS has been provided with a form-based interface. Implemented as frames, ENS treats forms and menus as states in a Finite State Machine (FSM). This FSM consists of a set of states, together with appropriate inter-state transition arc (these correspond to the function key settings which are made available to the user in each form), and a state-transition table consisting of a set of NextState (CurrentState, PFKey, NextState) prolog facts.

4.2.2.4 WISBER

is a (German) natural language consultation system. It covers the whole spectrum of natural language processing including analysis, response determination, and generation. The chosen domain of application is financial investment [Horacek et al. ’88]. Goal of this system, like many other consultation systems, is to fully automate the consultation process. Due to the highly interactive nature of the task, the dialogs play a very important role and therefore, while designing WISBER lot of care has been taken not only for dialog control but to keep the communication behavior of the system flexible. In the process of consultation, both the partners may initiate new sub-dialogs, e.g. by asking for additional facts or some explanation. In order to master such a task a system is supposed to infer the relevant intentions behind each user utterance [Grosz & Sidner ’86]. In WISBER those intentions are the basis for determining how the dialog will continue.

Knowledge Representation
The Knowledge used by the semantic-pragmatic components is split into a terminological and assertional part (each manipulated by a dedicated component), and a formalism dedicated to represent the meaning of utterances:

- QUIRK (QUIck Re-implementation of KL-ONE) provides means to construct a terminological knowledge base (TBox) of concept and role definitions with limited reasoning facilities [Bergmann and Gerlach ’86].
- QUARK [Poesio ’88a] is used to store and manipulate the assertional knowledge base(ABox) including facts about the user and the domain as well as the wants and beliefs of the participants in the dialog.
- IRS(Interne Repräsentations Sprache) [Bergmann et al. ’87], which is used by the components of WISBER at all levels of semantic-pragmatic analysis and generation.

QUIRK provides WISBER with a terminological representational language based on the syntax and semantics of NIKL [Schmolze ’85], reasoning capabilities (e.g. classification) as well as program and user interfaces. Furthermore, QUIRK is used for:

- resolving references [Frederking and Gehrke ’87],
- disambiguating scope relations and distributivity of reference [Fliegner ’88],
- performing terminological transformation on IRS formulas, e.g., for paraphrasing,
- detecting inconsistencies in assertions and queries [Bergmann and Gerlach ’87]

QUARK makes it possible to build and consult a knowledge source of assertions whose organization reflects the epistemology embedded in QUARK, and which is consistent with definitions of concepts and roles. To comply with the needs of a discourse understanding system, Quark provides means to store temporal information [Poesio ’87], to record the propositional attitudes of the participants in the dialog (e.g., (WANT USER (KNOW USER

P)), *´the user wants to know P`*), and to maintain alternative theories in a separate ABox includes the models of the user and of the system, the dialog memory, and knowledge about investment forms.

IRS is based on predicate calculus, but contains a rich collection of additional features required by different levels of processing between the initial representation of the user's utterance (containing some partially implicit ambiguities) and the 'deep' representation which can be stored in the ABox (the deep representation includes propositional attitudes and quantifiers rearranged according to their scope).

User Interface
WISBER does not support a fancy graphical user interface but has a reasonably good text based input output facility which supports the dialogue between the system and the user. Following Fig.4.2 shows a sample dialogue :

WISBER: GutenTag, hier ist WISBER
Hello, this WISBER
USER: Ich habe 40000 DM geerbt und möchte diesen Betrag anlegen.
I have inherited 40000 DM and want to invest this amount.
WISBER: Welche Laufzeit soll die Anlage haben?
What term should the investment have?
USER: Die Laufzeit der Anlage soll acht jahre betragen.
The term of the investment should be eight years.
.

Fig. 4.2: Sample dialogue of WISBER

4.2.2.5 ExTel

is an expert system to assist the sales person of DBP-TELEKOM (German PTT) during the sales consultation at various points of sales. This system was designed for two types of tasks, first to give detailed information regarding the products and services available and second to configure the potential telephone installation and check the technical and legal realization of these installations. The system was developed on a Siemens Computer of MX family which provide the teletext facility and Prolog chosen as the software platform.

ExTel's knowledge base contains more than 2000 rules and more than 20000 lines of Prolog code. The knowledge base contains text and pictorial information to all the products and services including respective prices and extra information regarding services like cable-connection etc.. The configuration system is not designed to give absolutely perfect technical solutions but to show how a proposed solution can be best adapted to the customer's wishes. Knowledge acquisition or revision of the knowledge base is done via the teletext facility which allows the post to maintain and sustain the system centrally. Due to interactive nature of the task and its users non familiarity with computers, the developers of ExTel provided the system with a graphic-oriented user interface which uses COLLAGE as a window-manager.

ExTel is the first nation-wide publicly installed expert system which provides a glimpse of the expert system technology and its usefulness to the public where everyone can form his/her own opinion about it [for details see Werner Kohl '90].

4.2.2.6 Consultant

is a knowledge-based system which advises on suitable Machine Learning tools available within the Machine Learning Toolbox. It is designed to assist a domain expert who is inexperienced in Machine learning. Consultant questions the user about features of his/her Machine Learning application and builds up a description of the task. From this, it provides an analysis which recommends suitable algorithms for the task.

In addition, Consultant provides a Help system containing information about the algorithms in the toolbox, their requirements and marriages between existing applications and Machine Learning algorithms. Consultant also allows access to a database of references to Machine Learning literature.

Consultant is a classification expert system where the user is interrogated about features of the learning task, and the consultant generally classifies this task as requiring one of the MLT algorithms [Kodratoff et al. '92]. It evolved through the following three versions:

- Consultant-0 [WP5, 1990], the initial version was a rule-based system implemented in Nextpert. it was designed very much like a black box, where the user answered the questions and at the end the recommendation was given by the system. In this version the user was unable to change responses to earlier questions, as he/she gained better insights into the task.
- Consultant-1 [Sleeman et al. '91] was more transparent to its user: it offers a flexible control of questioning and an extensive help system. To overcome the restrictions of expert system shells, Consultant-1 was implemented in Sun Common LISP; HyperNeWS 1.4 [HyperNeWS 1.4, 1990] provides the User Interface.
- Consultant-2 [Craw et al. '92] extends the functionality of Consultant-1 by giving advice on improving the performance of the recommended algorithm for the application. The pre- run advice of the Consultant-1 has been supplemented by post-run advice, recommending refinements to the parameter settings. In addition, it advises on changes of representation and suitable data sets.

Consultant comprises two main components: an advising module and a help system. Both subsystems can be accessed by the user in parallel. The user can thus be using the advising module, but consult the help system to allow him/her to understand his/her interaction with the advising module; e.g., he/she does not know some terminology, he/she wants to find out about currently recommended algorithm, he/she wants to know if his/her data is suitable for this algorithm, etc.

Advising Module
Here the system asks the users questions about their learning tasks and assumes that users have a clear picture of the task to be learnt and the type of knowledge to be acquired. From the answers provided, a Description Set representing the Task (TDS) is built by the system and a knowledge base links features of the TDS with pieces of advice. TDS triggers the evidence in the knowledge base for and against algorithms and "knowledge functions" integrate these pieces of evidence before presenting them to the user in a range of displays. The Advising module provides following three different interaction modes according to the user:

- Fixed Path Mode is suitable for the beginner because here all related questions are grouped together and presented in a focused logical manner.

- User Browsing is suitable for the expert user as it allows the user to answer those question which he/she may wishes to. It assumes that all the expert users know well which questions are relevant for their application.
- Intelligent Mode asks the fewest questions thus providing the most efficient questioning - here Consultant chooses the next question not in a logical manner but the question which has the most impact on the advice. In other words that question which discriminates most between the recommended algorithms.

Help Module
This module consists of a glossary and a set of help files together with simple means of accessing relevant topics from them. Consultant's help facility consists of the following three mechanisms:

- *Glossary:* short entries for basic terminology in machine learning
- *Information Browser:* a comprehensive source of help files on ML and MLT topics, descriptions of MLT applications and successful "marriages" with algorithms.
- *ML Database:* a collection of current ML abstracts [Morales '90], more relevant for ML experts.

Knowledge Base
Consultant's knowledge base is a network of properties which link features of the learning task with the suitability of various algorithms. Those properties which contain algorithm recommendations indicate the level of suitability using a certainty factor, a number between -1 and 1. A positive certainty indicates support for the algorithm, thus a certainty of 1 represents conclusive evidence for the algorithm. Negative certainties act against the recommendation, therefore -1 excludes the algorithm from the recommendation. Zero certainty neither supports nor detracts from this algorithm's recommendation.

To calculate overall algorithm suitability, Consultant also uses MYCIN's integration function for certainty factors [Davis '84], as this is still a standard method for handling uncertain evidence in classification systems. As Consultant only need to rank the various algorithms and describe the evidence using an eleven point scale, the certainties attached to Consultant's properties do not need to be very accurate.

Knowledge Acquisition
The knowledge acquisition process of Consultant was difficult because the consensus of the knowledge of several experts was required. In order to do this, a common terminology was defined which was suitable for all the algorithm developers. As Consultant's KB does not represent sufficient ML knowledge to choose the suitable ML approach, distinguishing features of MLT's algorithms must be represented, and detailed knowledge is only required when several algorithms with similar approaches must be compared for suitability [Kodratoff et al. '92].

4.2.2.7 PROSE

A Knowledge-Based Configurator that supports Sales, Engineering, and Manufacturing at AT&T Network Systems [Wright et al. '93]. Its knowledge base is written in C-Classic, a frame-based knowledge representation system in the KL-ONE family of languages. Thus providing its developers with the ability to add knowledge quickly and consistently in a purely declarative form. PROSE not only generates configurations from just a few high level

parameters, but it can also verify configurations produced manually by customers, engineers or sales people. The same product knowledge, encoded in C-Classic, supports both generation and verification of product configurations (for details see section 6.3.2).

NOTE:- At this point to support my observations and later to strengthen my line of argument I would like to call the attention of the reader to some more sales advisory/consultation systems (actually product configuration systems, no active sales support) which represent the automobile industry and are used by the industry as an aid to their respective sales person or directly the end customer (online) (see Appendix 1 and 3).

4.2.3 Summary

Here, we summarize our observations about the present sales advisory/consultations systems. All the above mentioned systems attempt to automate the process of consultation; whilst they claim that they help the sales person during the consultation but in fact they are automating the product information (online product catalogues). In other words, they use an oversimplified model of consultation (i.e. for them technical configuration is equal to consultation). Furthermore, none of the above mentioned systems have explicitly considered either the environment or the end user while designing their systems.

4.3 Aspects of Consultation

In this section, we try to define consultation in general as a communicative & cooperative situation and its implications to computer aided sales advisory. Furthermore, we also try to define what all is involved in the process of sales consultation. But prior to that, we would like to give the reader a small glimpse of how this subject of human-human advisory has been tackled in the past.

4.3.1 What is good Advice?

In the past, human-human advisory encounters have been a subject of investigation in a number of different contexts including face-to-face advice on using a local computer [Alty & Coombs ’80], [Coombs & Alty ’80], [McKendree & Carroll ’86] and radio talk show “call in” advice on finance management [Pollack et al. ’82]. The results uniformly indicate that good advice is more than recommending a solution. Good advisory interactions involve cooperative problem solving. The advisor does not merely respond to the immediate request of the user, rather he/she aids the user in problem formulation and plan generation (especially with regard to obstacles, side effects, interactions and tradeoffs). The Advisory encounter aids the user to determine the right questions to ask; how to look for or evaluate possible answers; and how to develop or debug a plan of action to achieve his goal [Jackson & Lefrere ’84].

Analyses of human-human advisory interaction clearly reveal the elements of good advice and point to the features required for effective joint cognitive systems. Machine decision support must be more than a solution plus justification; it must be structured around the problem solving process and involve close cooperation with the user in formulating the problem and identifying & evaluating the solution paths [Perkins & Martin’86]. The function of an “advisor” (man or machine) is to broaden the user’s horizons; to raise and help answer questions like [Woods et al. ’88, Woods & Hollnagel’87]: What would happen if? Are there side effects to this response? How do x and y interact? What are the pre-conditions (requirements) and post-conditions for a particular response(given the response, what consequences must be handled...etc.

Studies of human-human advisory interactions reveal another important characteristic of joint cognitive systems: the relationship between the kinds of skills and knowledge represented in the human and in the machine. The human user of decision tools is rarely incompetent in the problem domain. Even when the design intent is to reduce human skill and knowledge requirements, significant amounts of domain knowledge are required for minimal capabilities to understand and execute machine instructions [Roth et al.'87]. The relation of human to machine is not one of novice to expert; instead, the human and machine elements contain partial and overlapping expertise that, if integrated, can result in better joint system performance than is possible by either element alone.

Too often and in contrast to the above results, designers of consultant systems implicitly assume that "advice" is synonymous with outputting an answer, that is, solving the problem for the practitioner. For example, [Zisner & Henneman'88][Resnick et al.'87] contain examples of laboratory based machine advisors which simply recommended a solution to the human problem solver, which were ignored.

There are several challenges in formulating and delivering advice. What is the appropriate level of context-sensitivity and specificity to aim for in generating advice? Should the system attempt to generate advice under all conditions or only for those situations where the appropriateness of the advice can be assured? Should the system generate highly specific micro-responses or should it generate more global information leaving the operator degrees of freedom in deciding how to instantiate the advice? For answers & discussion about these question please refer to [Rassmussen'86] [Roth et al.'87] [Woods & Roth'88] [Suchman'87].

4.3.2 Consultation: What is it?

In order to understand the communicative situation - Consultation/Advisory - we conducted interviews and surveyed the literature. In the following we present an analysis of interviews based upon the transcripts of some of these interviews followed by an analysis of the literature as well. For the interview the following set of questions was designed:

Q1. How can one define a communicative situation called Consultation/Advisory on the basis of: Agents Involved; Their Functions; Their Roles; Task distribution among them; Cooperation among them and its Type; Communication among them and its Type and finally the medium of communication among them.

Q2. Is it possible to automate consultation ?

Q3. What classes/types of knowledge are relevant for consultative or advisory situation?

Q4. How does a consultative/advisory situation effect the configuration process?

Before we start defining the process called consultation, let us consider an example or two for the illustration purposes:

Scenario A:

C. Hi! I'm looking for a TV for myself!

S. Hi! Have you looked around here?

C. No!

S. Then have a look around, first. Then we'll talk about the rest.

C. [Annoyed] Good bye!.... leaves the showroom

Scenario B:

C1. Hi! I'm looking for a TV!

S1. Hi! What do you have in mind?

C2. Well nothing fancy but it should have:

- Color system
- app. 51 cm's. screen
- Stereo sound
- Teletext &
- naturally a cable tuner

S2. How much would you like to spend?

C3. Well it is immaterial at the moment; but what do you say about "TV-X" and "TV-Y" ?

S3. Certainly, "TV-X" because "TV-X" has the best Picture Tube among all the available makes.

C4. Well then! How much would a "TV-X" cost?

S4. xxxx DM

C5. and a "TV-Y"

S5. XXXX DM

C6a. O.K. I'll go in for Brand "TV-X"

S6a. Well, let me get the contract form so that you can sign it

or

C6b. Well! Thanks a lot but I need sometime to think

S6b. Thanks, for showing interest. Here is my card if you need some more information or decide for one of our products please give me a call

or

C6c. O.K., If I go for Brand "TV-Y" but what about the discount and the delivery date?

S6c. Normally, if you pay cash we give 3% discount on all the brands but on TV-Y we pay 5% and its immediately available

C7. Do you deliver it to home?

S7. Yes, but it costs DM 50 extra and its delivered after 3 working days

Scenario C:

C1. Hi! I'm looking for a TV!

S1. Hi! What do you have in mind?

C2. No I don't no, may be you can suggest one.

S2. Well, Do you prefer a black and white or color TV?

C3. Color TV!

S3. What size of TV would you like ? (small, medium, large)

C4. Well, medium should be alright.

S4. Would you prefer a stereo or mono TV?

C5. Of course, Stereo!

S6. Would you like to have teletext and cable tuner as well?

C7. Cable Tuner yes. But what's the benefit of teletext?

S7. With the help of teletext system you can retrieve lots of information regarding everyday TV programs, news, weather, stock exchange and so on....and that free of cost.

C8. Well! it seems interesting but I can do without it.

S8. How much would you like to spend?

C9. Well, My budget is around xxxx DM plus minus yyy DM

S9. Now, you are looking for a TV set which has:

- color system
- app. 51 cms. screen
- stereo sound
- cable tuner
- teletext (optional)

C10. Yes, certainly.

S11. TV-X which costs XXXX and TV-Y which costs YYYY would fulfill your requirements.

C12. Why is TV-X more expensive than TV-Y?

S12. TV-X has also a teletext system

C13. Is there any other difference?

S13. Yes! The picture tube of TV-X is the best among all the available brands

C14a. O.K. I'll go for "TV-X"

S14a. Well, Let me get the contract form, so that you can sign it

or

C14b. Well! thanks a lot; I need sometime to think

S14b. Thanks, for showing interest, here is my card if you need some more information or decide for one; please give me a call.

or

C14c. O.K. I'll go in for "TV-Y" but what about the discount and the delivery date?

S14c. Normally, if you pay cash we give 3% discount on all the brands but on TV-Y we pay 5% and its immediately available

C15. Would you deliver it to home?

S15. Yes, but it costs DM 50 extra and it's delivered after 3 working days

Now, when we try to analyze these three scenarios, we find that even though they all started with the same notion which we call an initiation of consultation but result in three different situations: (a) no consultation occurred (b) consultation for confirmation of tentative decision (c) consultation leading to a decision.

Scenario A, being a "no consultation occurred situation" is of no interest to us and will not be considered for further analysis. In scenario B and C in the initiation process the domain/context (e.g. product TV) for the consultation was fixed. Thereafter, the sales person tries (with the help of question S1) to find out the customer's requirements. In scenario B the customer seems to have well defined requirements whereas in scenario C (S2-C8) the sales person has to identify them and while doing so he tries to explain certain features of the product and their benefits as well. Before the sales person begins to map the customer requirements to product features he tries to identify the restrictions (S2 and S8 respectively). From now on the sales consultation described in scenario B deviates from the consultation described in scenario A.

Let us first consider scenario A, as soon as the sales person put question S2 to find out the restrictions; the customer instead of telling of the restrictions, asks a question C3 in which he is seeking confirmation of his tentative decision. In response to this, the sales person has to leave his original plan to map the customer requirements to product features and initiate a process which does a comparative analysis of features of the different products as suggested by the customer. The sales person not only makes a decision for the customer but also supplies an additional piece of information as an explanation supporting his decision, which helps him convince his customer. To be sure the customer requests additional information, which are then supplied by the sales person. After evaluation of this additional information the customer is able to make a decision. Depending upon the nature of the decision, the future course of sales dialog is determined, e.g. in case of C6a the sales person tries to close the sales dialogue by making the customer commit to the purchase of TV-X by signing the contract. In case C6b the sales person politely provides the opportunity for the customer to continue the sales dialogue either immediately or in near future. In case C6c, the customer seems to be not fully satisfied by his decision and requires more information either to confirm the decision or revise the decision in the light of new and additional information.

Now, let us consider scenario C, here as an answer to S8 the sales person gets the restriction which helps him shorten the list of the possible solutions which he had tentatively prepared.

The different solutions which satisfy the restriction are then provided to the customer so that he can compare the suggested solutions and make his own decisions. The only difference in both the scenario's is that in scenario B the sales person is actually making a decision on behalf of the customer whereas in scenario C he is steering the customer to make his own decision. Another difference is that in scenario B the sales person is forced to change to different plans as the situation changes whereas in scenario C he can more or less stick to his original plan.

The above analysis of different sales scenario's may lead to the following tentative definition of consultation in general:

4.3.3 Tentative Definition:

Consultation is a multi-step/layered (communicative) interactive cooperative-process, which involves intentions, where the intentions of the person being counseled are to be judged by the counselor and brought to a consensus with his/her own intentions to finally set a long term counseling strategy/plan. In its simplest form it takes place orally and involves a minimum of two agents, where one is a novice (an agent who lacks & seeks information) and the other an expert (an agent who posses & provides required/seeked information) in a specific field. This process can further be described as change of states, where the initial state is described by the ill defined (diffuse/fuzzy) solution space and the final state is described by the well defined solution space which ultimately leads to a decision making process.

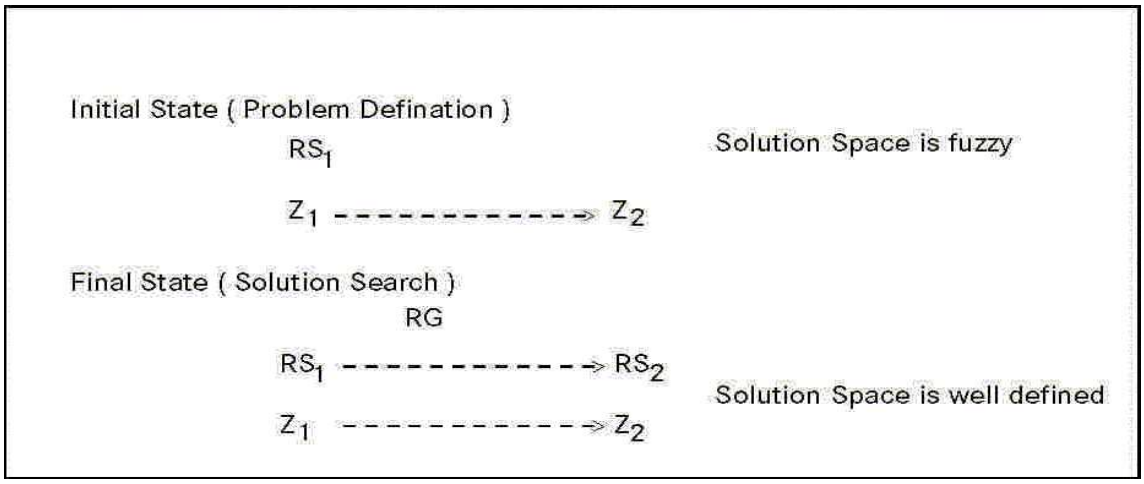


Fig. 4.3: Consultation in terms of solution spaces

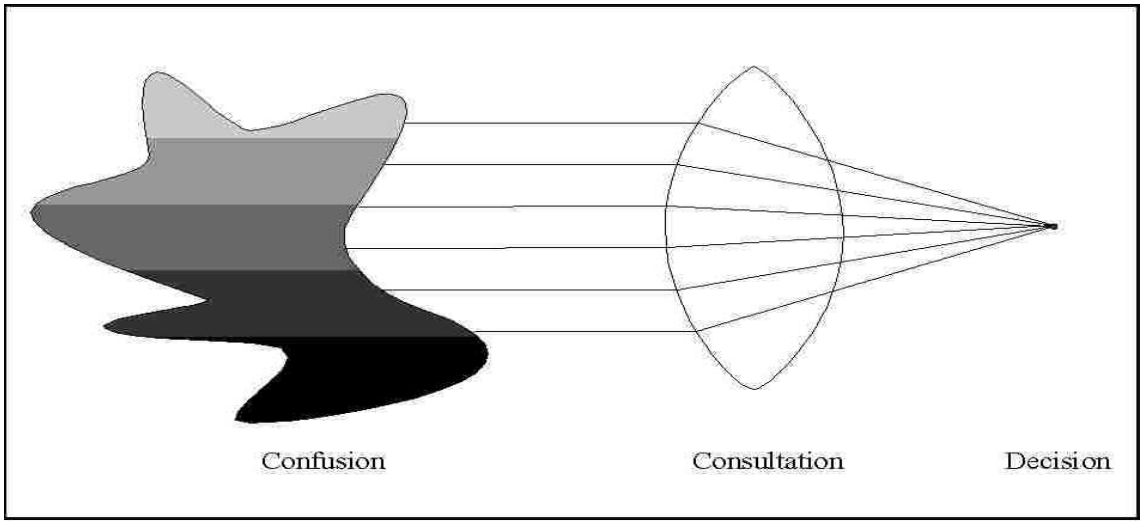


Fig. 4.4: Consultation as focusing process

4.3.4 Classification:

The process of consultation can be classified according to different criteria and views, as shown in Fig 4.5:

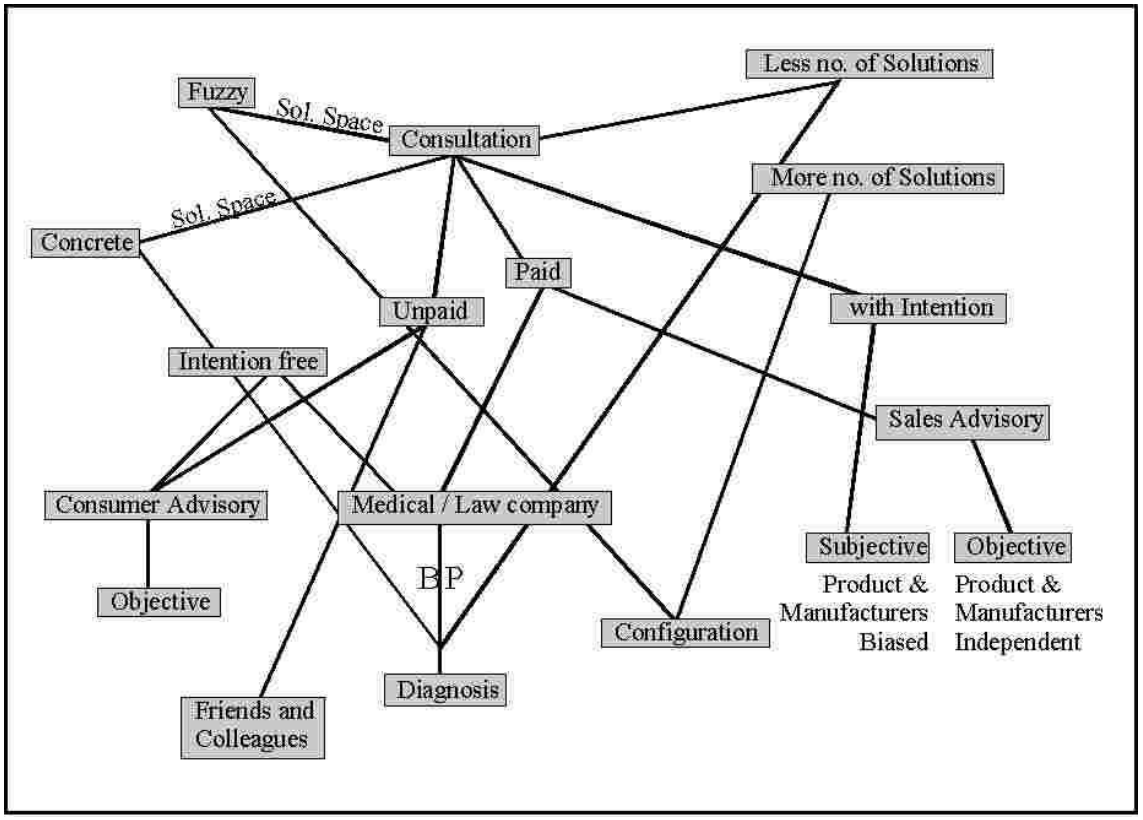


Fig. 4.5: Classification of Consultation

Here, we try to classify Consultation/Advisory into two parts one is Intentions Free and another (could be) with Intentions or Purpose bound. The instances of the two kinds could be consumer and medical advisory. Medical advisory for the former (i.e. intentions free), and

sales advisory for the latter i.e. with intentions or purpose bound; Whereas, the latter subsumes the former.

In another attempt, consultation can be classified into different classes depending upon the goals and intentions of the involved agents. For example:

Objective/Product Independent Counseling
Subjective/Product or Manufacturer biased or
Paid Services (lawyer, doctor etc.)
Unpaid Services (tradeunion, colleagues or friends etc.)

Yet another attempt to classify counseling may lead us to following different classifications:

According to Decision Space: Fuzzy and Concrete
According to Number of Solutions: More or Less
Background Process: Configuration or Diagnosis

Finally, consultation can be classified according to styles of cooperation, which we will consider at some later stage.[1]

Most of the researchers, to date, have studied the process of consultation without considering the intentions and have produced results in this direction with different degrees of success. In the present research we are trying to study the process of consultation in the context of sales advisory where intentions are involved and where its a paid activity (transaction of money is involved) and has configuration as a background process, the solution space is ill defined (fuzzy) and has a large number of solutions too. Furthermore, we consider two different aspects of this process: objective (product and manufacturer independent) and subjective (product and manufacturer biased), Fig. 4.6

[1] Note:- Generally, the consultation/advisory process is neutral but in a sales situation (with intentions/purpose bound) it is a part of the whole. In other words consultation/advisory with intentions or purpose bound (C/AwI/P) does always contain a neutral part which consists of product independent knowledge. From a technical point of view counseling may include: non monotonic reasoning, constraints satisfaction etc..

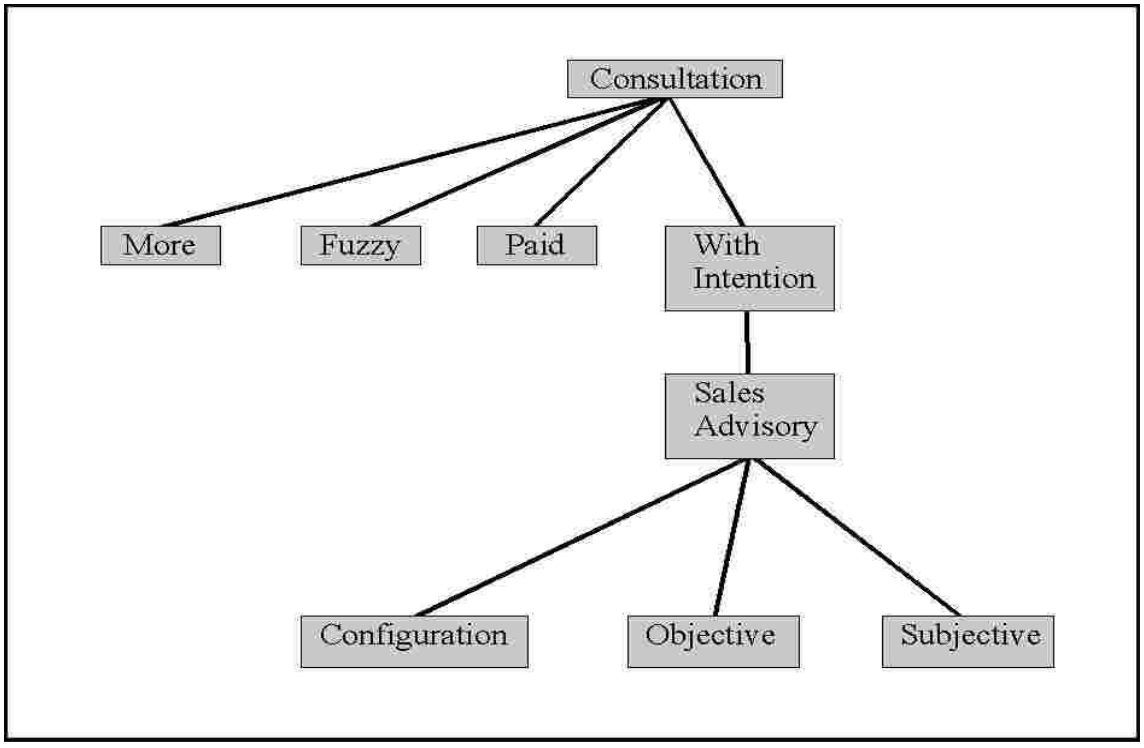


Fig. 4.6: yet another classification of consultation process

4.3.5 Consultation as Communicative Process:

According to many theorists the simplest form of interaction between organisms occur when the behavior of one organism has a causal effect on the behavior of another. If the behavior of the first organism has no function other than to bring about the behavior of the second, then one speak of a communication as having occurred. This may be true for the animal world but the human communication is more complex than any other because it can use language. Although human beings can communicate in other ways - by gesture, by facial expression, by emotional cries - and although language can be used for purposes other than communication - for the externalization of thought, for self-expression - no system of communication is as rich as natural language [Johnson-Laird'88].

As we gather from all the three above mentioned scenarios that they are highly interactive among the participants which are humans and use language as the means, thus the process of consultation can be termed as communicative.

4.3.5.1 Communication:

once, we have defined and classified consultation, as a communicative process let us now analyze this phenomena of communication which is involved in consultation/advisory.

The communication among the agents involved in counseling/advisory should/could be based on every other Human Communication which could be divided into three parts: Syntax, Semantics and Pragmatics.

The first (Syntax) deals with the problems of information transmission e.g. coding, channels, capacity, noise, redundant and other statistical features of the language etc.; all these problems

are of a syntactical nature and have nothing to do with the meaning of the used symbols. (role of syntax in consultation)

Meaning is the main concern of semantics; even though it is possible to transmit series of symbols with syntactic precision, they would remain senseless unless and until the receiver and sender have (in advance) come to an agreement about their meaning. In this sense, to every information a semantical agreement is a prerequisite (role of semantics in consultation).

Further, every type of communication influences the behavior of all the agents involved and that is the pragmatic aspect of it (role of pragmatics in consultation)[1].

Finally, according to Meferet [Mefert'98] *"The result of the sales processes can be determined by all the elements of a communication process: characteristics of an organization as "Sender", characteristics and behavior of a salesperson as "Medium", characteristics of a customer as "Receiver" and finally the quality of the sales arguments as "Messages".*

4.3.5.2 Communication seen through the Perspective of Speech Act Theory:

The examples given above (scenario a, b, c) clearly show that during the consultation communication naturally takes place and could be implicit (to perceive) and/or explicit (language etc.).

To analyze this in detail we make use of all the levels of communication as defined in terms of speech acts [Austin '62][Searle'69]:

1. **Locutionary:** the act of saying something, which comprises the phonetic, phatic and rehtic acts, is termed by Austin a Locutionary act. Where the phonetic act is the production of phonetic sounds, the phatic act indicates the production of lexically/syntactically constrained elements and the rhetic act is the production of a meaningful utterance, raising questions of sense and reference. The Locutionary act could thus be either performative or a constative (i.e. it is not the carrier of this distinction). According to Austin with the constative utterance, we are really concentrating upon the Locutionary aspects, and in particular upon the rehtic aspect (i.e. the sense/reference) of the speech act. By contrast, with the performative utterance, we are mainly concerned with its illocutionary force, abstracting from questions of correspondence with the facts.
2. **Illocutionary:** Austin proposes to call the performance of a speech act in saying something, as distinct from the Locutionary act of saying something, an illocutionary act. Thus an illocutionary act could be a request, a warning, a command, a verdict(or a statement).
3. **Perlocutionary:** the further act (collection of consequences) which is consequent upon the illocution is called by Austin the perlocutionary act: typically it is beyond the control of the utterer, unlike the Locutionary and illocutionary acts.

An example is that ordering someone to do something is illocutionary, but getting him to do it is perlocutionary. The same sentence may serve as both, but the conditions for successful

[1] Note:- Theoretically, there is a clear cut demarcation among all the three but practically they all are interchangeably dependent on each other. "*it is right -in many respect- to say that syntax corresponds to mathematical logic, semantics to philosophy or science and the pragmatics corresponds to psychology but still these fields are difficult to separate from each other*" [Watzlawick et al.'67].

perlocutionary acts don't just involve what the speaker says; they involve its effect on the hearer. Though it is difficult to make the distinction precise but for the sake of consultation it's easy. Thus making our view of speech act some what corresponding to the view presented by McCarthy *„My view of speech acts is that they are necessary in the common sense informatic situation in which people interact with each other to achieve their goals"* [McCarthy'98].

This lead us to the observation that speech acts are valuable in desigining computer systems to interact with humans (consultation/advisory systems), and with each other. Another point which is of important to us is that not all the aspects ascribed to speech acts are valuable to our purpose of sales advisory systems. Which ones they are will depend on the purpose(e.g. different type of sales advisory) itself. For further details on discussion about speech acts and their applications (e.g. converasational recommendation systems[Göker & Thompson'00]), we recommend the interested reader to the literature as found in multi agent systems discipline and some rescent publications [Elio & Haddadi'98& 99][Langley et al.'99].

4.3.5.3 Medium of Communication

is heavily dependent on the situation/context and maybe on the domain, too. Therefore, it could be any combination of language, paper, pencil, catalogue, stills, audio-video aids etc. In our case, in the domain of selling a truck, the salesperson of the truck manufacturing company uses, at present paper, pencil and a catalogue during the sales dialogue. Looking into a catalogue for prices and technical restrictions and explanations like why a particular accessory can and/or cannot be ordered with a specific product model or with already chosen some specific accessory. This leads to lengthy and boring sales dialogues which in turn are mostly very inefficient. To improve upon this situation a salesperson requires - apart from language - a medium not to communicate but one which facilitates all the necessary information required promptly and in the required form (i.e. in the form of numbers, texts, stills, video or audio) while he/she is communicating with the customer. Thus, an assistant which is a combination of paper, pencil, catalogue, stills and audio-video. Here, one can only think of one thing and that is a computer with all the capabilities.

4.3.6 Consultation as Cooperative Process

Another important aspect of consultation, which we gather from the example scenarios, is that the interaction taking place during the consultation among the involved agents is of a special form, where the involved agents have some intentions and each one of them tries to behave in such a manner that both could achieve an optimum while satisfying their individual intentions.

Now, if we consider a definition of cooperation which says that cooperation is a special form of goal-directed behavior, where the involved agents share (may not share) goals and act in concert with them over an extended period of time.

And if we now substitute goals with intentions and the goal-directed behavior as a special form of interaction, then we get precisely the same description of that particular aspect of consultation (as mentioned above) which makes consultation termed as a cooperative process.

Now, if we consider consultation in the context of sales advisory, then we find out that it is a process of cooperating without sharing goals e.g.

> I need to buy a truck and therefore I go to the sales organization of a leading truck manufacturer and once there; ask the sales person to suggest an appropriate truck.

In this example, both parties though have different intentions (buying and selling) but both are still engaged in a cooperative process while the intentions involved here are positive ones. By positive intentions we mean those intentions which help both the parties involved - in sales consultation - to achieve a certain optimum in satisfying their individual intentions i.e. they result in a win-win situation e.g.

A win-win situation, in the above example, would be if I could buy a truck which suits my requirements and is reasonably priced. For the sales person it would be good if he/she could sell me a truck where he/she makes maximum profit for the manufacturer.

Now, how would the sales person, in the above example, know what I exactly want? Here, one could consider three different possibilities for the communication of buyers (agent-1) goals to the sales person (agent-2): First, agent-1 could have told agent-2 what he or she wanted e.g.

I could have asked the sales person of a truck manufacturer straight forward;

I need a Truck of type XYZ with A, B, C, and D as extras (accessories).

One can refer to this as request-based cooperation between the agents. In the request-based systems, agents need not know anything about the behavior of other agents. They require only a communication network that allows requests to be passed efficiently. (Least interesting)

Second, agent-2 could infer agent-1's goal from the behavior of agent-1. This is also an act of communication, except that agent-1 does not intend (or is unable) to communicate the goal. This type of inference would require that agent-2 know the kinds of behavior that count as evidence for another agent's goals. One can term this as inference-based cooperation. In the inference-based cooperative system, the agents must have access to information about the behavior of other agents in order to make inferences about their states and capabilities. Agents who initiate actions on behalf of other agents must know enough about the way other agents are structured to be able to undertake useful activity at appropriate times. Also, agents who have suspended problem solving to await the results of other agent's actions must be informed about the outcomes and effects of these actions. Questions concerning such systems might involve plan recognition problems, communication among agents about intermediate results, availability of information about the behavior of agents to other agents, or differing roles and capabilities of specialized agents.

The third possibility is that agent-2 helps agent-1 because the situation is structured in a particular way—as in above example agent-2 is a sales person and agent-1 is a buyer and is engaged in the process of buying, then agent-2 automatically helps agent-1 in case of any difficulty. This can be termed as structurally-based cooperation. Agents in a structure-based cooperative system have predefined roles that can include initiating actions on behalf of the goals of other agents[1].

4.3.7 Summary:

Much of the human-machine interaction literature stress the importance of structuring computational environments so that they help humans to achieve their goals. There is a recognition that cooperation can be passively engineered into the design of a human-machine system. This forms the basis for our argument that all the advisory systems are to be designed

[1] **Note:-** one can conduct the counseling in a standardized fashion if the roles (of cooperation) are clearly defined. That means, more clear roles result into more standardized counseling [Werner'89].

as communicative and cooperative systems and where cooperation among the agents (buyers & sellers) is acheived through the mutual communication (sales dialogue/consultation) which is based upon some form of speech acts.

4.4 Sales Advisory: A Memorable Buying Experience

"Sales is the lifeblood of any business. It is the only activity in business that contributes to profits"

(Scott Clark, business consultant & columnist, Rapis, Iowa).

In today's business world *commoditization* means the differentiation of product (goods or services) disappears, margins fall to rock bottom and customer buy solely on the basis of price. No business wants that word applied to its offerings, so what does one do?

Historically (till now) business has been evolved around the three (commodity, good, or service) major economic offerings (depending upon what business does with it) with three distinct ranges of value customers attach to the offering. In today's fast paced business environment, where many companies compete for the same dollar, are trying to make their offerings distinct from each other in form of a new economic offering termed as "***experience***".

Experiences are a fourth economic offering [Pine and Gilmore'99]. They are distinct from services as services are from goods and bring in many fold increase in the price of a commodity (e.g. coffee beans) if sold as a distinctive experience (cup of coffee at Eiffel tower). Does this lead manufacturers to jump into the experience business itself ? No, according to economist Stanley Lebergott [Lebergott'93], Manufacturers must focus on the experience customers have while using their goods. For example, Recreational Equipment, Inc. (REI) erected a fifty-five-foot mountain that customers climb to test out its gear. In other words "manufacturers must explicitly design their goods to enhance the user experience as well —essentially experientializing the goods—even when customers pursue less adventurous activities." [Pine & Gilmore'99]. For example automakers do this while focusing on enhancing the *driving experience*.

To cap their full-fledged entrance into the experience business, companies will have to concentrate on more than one experiential aspect which their good encompass. Not only that, more and more companies would need to wrap experiences around their existing goods and services to differentiate their offerings. In case of automobile industry, apart from the *driving experience*, one would enhance the task of ***sales advisory*** as a ***memorable buying experience***.

In the following sections I'll try to consolidate that how the task of sales advisory can be turned into a memorable buying experience. Next subsection provides different ways of defining experience followed by different types and theories of experience. Further defining sales advisory as a designed interaction and finally turing this interactions to an experience.

4.4.1 Ways to Define Experience

It is difficult to define experience as the word "experience" is general and ubiquitous. Thus in literature one finds lots of different approaches to define experience.

The simplest way to talk about experience is as the constant stream of thoughts and sensations that happens during conscious moments. This definition is inspired by cognitive scientist

Richard Carlson's theory of consciousness known as experienced cognition [Carlson'97]. Self-talk or self-narration are the typical examples of this type of experience and the designers refer to this type of experience when talking about stimulus such as visual design elements or about task flows and transactions.

Another type of experience, known as an experience -is described by philosopher John Dewey[Dewey'63]. This type of experience has a beginning and an end, and changes the user, and often, the context of the experience, as a result. A simple example of an experience is a fine meal in a restaurant which comes to define or epitomize a fine dining experience. People often acknowledge this type of experience by telling stories about the experience, or by using the experience to compare and judge new experiences. When designers wish to provide a particularly memorable or category-defining experience, they may be referring to "an experience".

The connection between storytelling, memory and experience is an interesting one, and has been discussed at length by Roger Schank[Schank'90], a cognitive scientist. Stories are the ways that people condense and remember experiences, how they choose important threads to communicate in certain situations, and how experiences are relayed from person to person. Experience as story plays an important role in events as diverse as courtroom proceedings and television soap operas. These ways of talking about experiences are related. For example an experience can be made up of an infinite amount of smaller experiences, relating to other people, settings and products.

In addition, experiences have different amplitude -- some play heavily on our emotions, and some are simply about accomplishing a task with minimal effort. Some experiences we want to collect (for example, seeing a movie or visiting a painting in an art museum) and some we don't want to repeat (travel delays or difficult interactions with nasty clerks). One point which one wants to make here is that not all intended experiences had to be positive; for example, aversion training to quit smoking works on the principle of creating a bad experience.

4.4.2 Types of Experience:

To understand what kind of experience is appropriate for a particular audience in a particular context, one needs to make distinctions about the types of experience based on particular product audience and business goals. In addition we need to understand better the principles of how people interact with various types of artifacts (products, services and environments), and how those interactions affect the experience people have or in other words what kinds of experiences we can facilitate and how interaction styles might affect these experiences:

- *The Experience of Knowledge:*

Knowledge is a phenomenon that one can build for others just as one can build information for others from data. This is done through interaction design and the creation of experience. With every experience, one acquires knowledge; it is the understanding gained through experiences — good or bad. Knowledge is communicated by building compelling interactions with others or with tools so that the patterns and meanings in their information can be learned by others. There are many types of experiences that confer different types of knowledge. Some knowledge is personal, having meaning unique to one person's experiences, thoughts, or point of view. Local knowledge is knowledge shared by a few people because of their shared experiences. Global knowledge is more general, limited and process-based, since it relies on such heavy levels of shared understandings and agreements about communication. Effective communication must take into account the audience's level of knowledge.

- *Productivity and Creative Experiences:*

Some experiences can be used more productively than others (such as entertainment) and productivity is traditionally of more concern in business products than entertainment products— but being creative and producing something are typically more interesting, entertaining and fulfilling activities. Creative experiences allow a user, creator or participant to make, do, or share something themselves. Thus creation tools are important components for creating meaningful, compelling and useful experiences. People are naturally creative and are almost always more interested in experiences that allow them to create instead of merely participate. While many situations can create anxiety if people are not accustomed to performing with tools or techniques, if this anxiety can be lessened (either through the careful design of the experience or offered assistance), people express their creativity. This can take the form of recommendations, guidelines, ***advice*** or actually performing operations for users.

- *Adaptive Experiences:*

Adaptive technologies are those that change the experience based on the behavior of the user, reader, consumer or actor. These can include "agents" modifying behaviors and "pseudo-intelligence". Agents are processes that can be set to run autonomously, performing specific, unsupervised (or lightly supervised) activities and reporting back when finished. Modifying behaviors are those that change the tools and/or content involved based on the actions of techniques of users (e.g. Games becoming difficult as the level of proficiency of the player increases). Other possibilities include content changing to reflect point of view, level of proficiency required or amount of detail desired. Both of these techniques might have the effect of making the device or person in an experience appear intelligent, as might other techniques. Without going into a deep discussion on intelligence, life etc. It is sufficient to say that certain kinds of choices in changing behavior based on the actions of others (whether random, instinctive or algorithmic) can create the appearance of a more sophisticated system or process and imply a kind of intelligence.

- *Communicative Experiences:*

Like productive and creative experiences, opportunities to meet others, talk with them and share their personal stories and opinions are always viewed as valuable and interesting. Because these experiences involve two or more people, they also inherently involve high levels of control, feedback and adaptivity. The telephone is an excellent example of a communicative experience, as are chat lines, discussion boards, cocktail parties and sales advisory. Some of these are so valuable and enjoyable for some people, that they have become virtually indispensable.

4.4.3 Theories of Experience:

Experience has long been a topic of interest to philosophers but has only recently come into vogue among designers and business people. One of the first article published on the explicit position on the possibility of designing experiences is by Alben [Alben'96] and the second to mention experiences in the business context was by Pine and Gilmore [Pine and Gilmore'98], published in Harvard Business Review, where they argued that the world is transitioning from a service economy to an experience economy. Here instead of going into details of various topics/questions involved in the study/understanding of experiences, e.g. what kind of experiences can be facilitated and how interaction styles might affect these experiences, we will briefly mention the work of [Forlizzi'97]; [Pine and Gilnmore'98]; [Rhea'97] and refer to the following different theories of experience for further details: Quality of Experience [Alben'96]; Experience-Based Design [Cain'98]; Experienced Cognition [Carlson'97];

Experience and Education & Art as Experience [Dewey'63 Dewey'80]; Utility, Ceremony and Appeal [Hudspith'97]; Getting to know the User [Margolin'97].

- *Design for Experience[Forlizzi'97]:*

Forlizzi attempts to understand experience as it is relevant to interaction design. Two types of experience, satisfying and rich, are identified in user-product interactions. A satisfying experience is a process-driven act that is performed in a successful manner. A rich experience has a sense of immersive continuity and interaction, which may be made up of a series of satisfying experiences. Designers are given a set of principles for designing rich, immersive experiences.

- *Focusing on customer Experience [Rhea'97]:*

In the article [Rhea'97], Rhea argues that the real opportunities for design come from assessing all the ways in which products and services might influence and benefit customers -- physically, emotionally, intellectually and culturally. But taking advantage of these opportunities demands a shift in focus from products themselves to customer experiences.

Further, Rhea presents a model to conceptualize how a customer's everyday experience with artifacts moves through a cycle (Fig. 4.7). The cycle is comprised of four stages: Life Context, Engagement, Experience and Resolution. Briefly, Life Context refers to the background of consumer's lives including everything the consumer thinks, feels and does (behaviors, needs, attitudes, perceptions etc.). This context is mostly unarticulated and constantly changing.

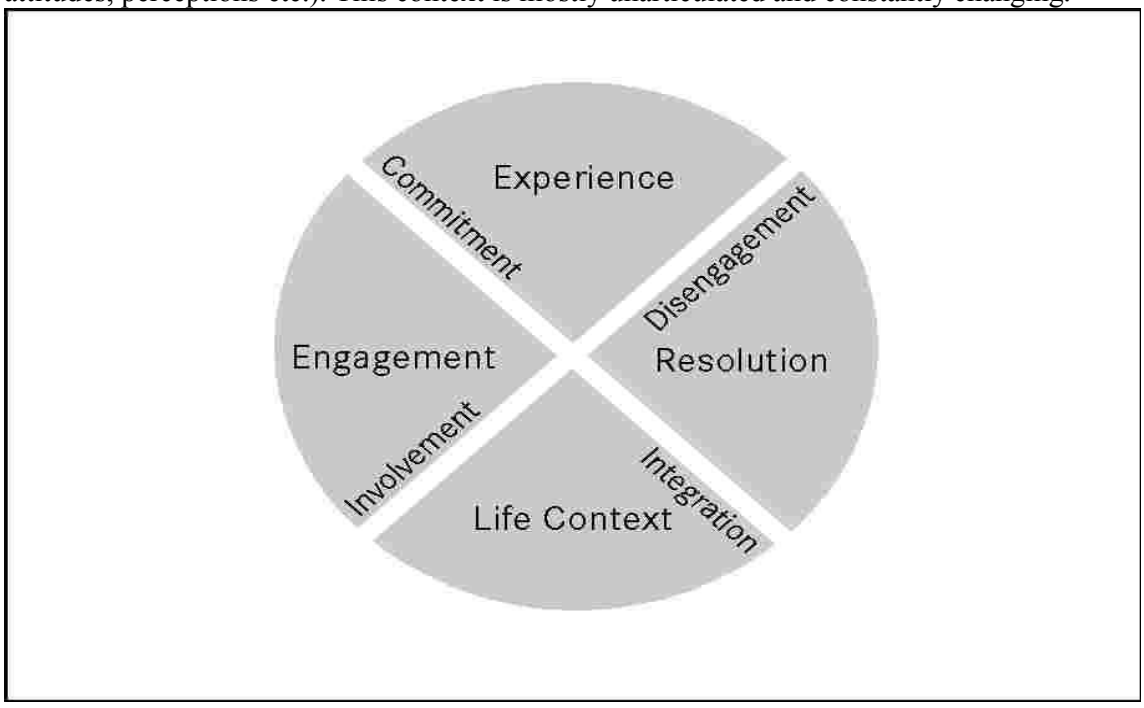


Fig. 4.7: Focusing on customer experience (source: Rhea'97)

Engagement refers to the initial interaction a customer has with an artifact. The importance of this initial involvement id "profound" and may be influenced by a number of factors, including prior experience with the product, advertising, word-of-mouth, the product's "cognitive presence", attraction and communication.

Experience refers to the period of ownership and use. During the use, customers continually assess the quality of their experiences with the product. In this stage, the product must be reliable, creating a pleasing experience that meets expectations, addresses concerns, solves problems and fits into customers lives and gives customers something extra.

Resolution refers to both the experience of disposing of the product and how customers resolve (or think about) their overall experience with the product to form a lasting impression. This impression then feeds back into Life Context and forms a basis for expectations and desires for the next cycle.

- *Welcome to the Experience Economy [Pine and Gilnmore'98]:*

Pine & Gilmore claim that "consumers unquestionably desire experiences". The mark of success, for them, is the ability to wrap products and services with deliberately designed, engaging experiences that command a fee. "An experience occurs when a company intentionally uses services as a stage and goods as props, to engage individual customers in a way that creates a memorable event". They explore examples of the intentional design of experiences primarily from the "shoppertainment" and "entertailing" world - such as Niketown & rainforest Cafe - and from new media applications such as virtual reality and chat rooms. According to them experiences are not exclusively about entertainment but their text is biased towards having "an experience" in Dewey's terms. They also offers a model for characterizing experiences and principles for designing memorable experiences: theme the experience; harmonize impressions with positive cues; eliminate negative cues; mix in memorabilia; engage the five senses (Fig. 4.8)

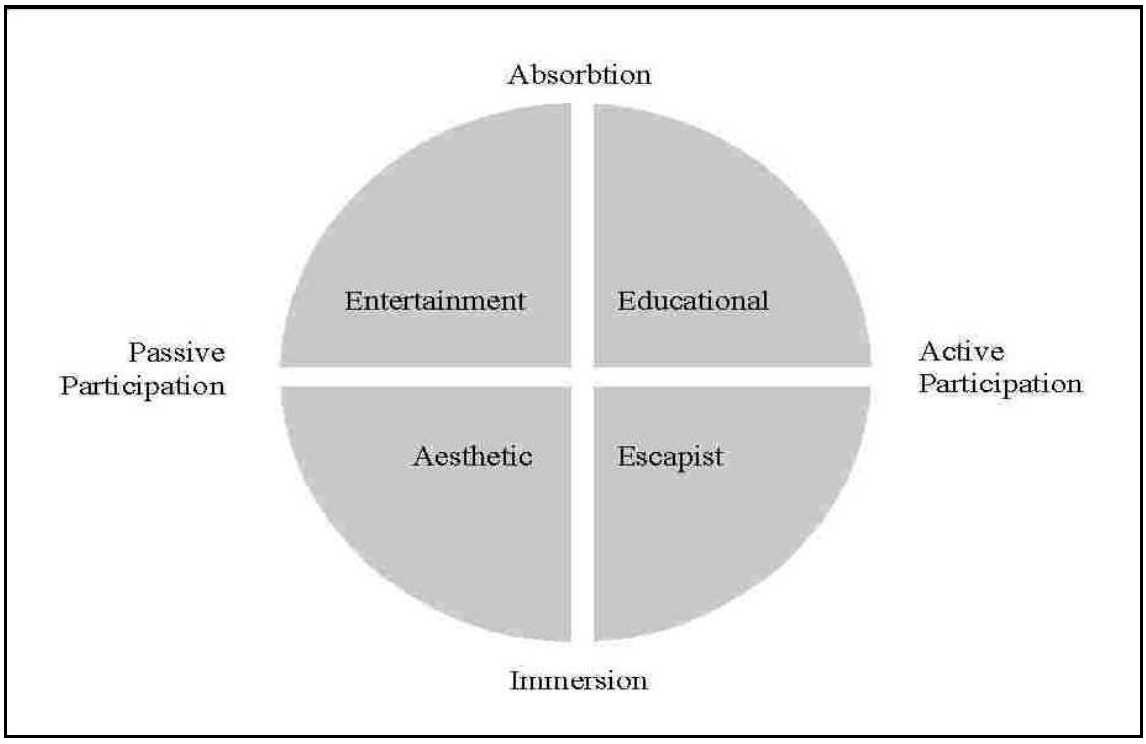


Fig. 4.8 Experience in all it's realms

They do caution, however, that following these principles is no guarantee of success, and hint that constant updating of the experience is necessary for long term success.

4.4.4 Experience:

"*Experiences are events that engage individuals in a personal way*"

[Pine & Gilmore'99]

In their recent book, *The Experience Economy: Work is Theater & Every Business a Stage*, Pine and Gilmore provide a detailed insights about how the experiences are going to help build an emerging economy, called the *Experience Economy*. According to them *"experiences have always been around but until now gone largely unrecognized because the consumers, business and economists have lumped them into the service sector"*. Further transition to this new economy, in which experiences fuel the engine of growth, begins when companies give away experiences in order to sell existing offerings better. Thus they suggest that to fully ripe the benefits of this economy the business would have to separate experiences from services and not only that, experiences are not going to be considered as an economic offering unless one starts charging explicitly for that. In their own words

"you're not truly selling an experience unless you charge admission"

[Pine & Gilmore'99]

In their book, they define experience as a new form of progression of economic value and believe that the same effect -customizing a good automatically turns it into a service-holds for experiences as well, i.e. customizing goods and services turn it into an experience. Based on this finding they suggest that as customizing a good turns it into a service, similarly individualizing the service can give the consumer sensation of experience.

To stage a positive experience they recommend principles of mass customization, where mass customizing means *"efficiently serving customers uniquely, combining the coequal imperatives for both low cost and individual customization present in today's highly turbulent*, *competitive environment"* and it doesn't mean *being everything to everybody; rather, it is doing only and exactly what each customer wants, when he wants it and at a price he is willing to pay.*

Depending upon the type of customer sacrifice (as defined by Dave Power III of J. D. Power & associates: *"when we measure satisfaction what we are really measuring is the difference between what a customer expects and what the customer perceives he gets"*) the following four types of customization approaches: *collaborative customization; adaptive customization; cosmetic customization and transparent customization*. Each of these approaches act as the basis for a distinct kind of experience: *the exploring experience, the experimenting experience, the gratifying experience and the elusive experience* respectively [see for details, Pine and Gilmore'99].

To choose the right approach is not a simple task as each type of customization addresses a different kind of sacrifice. For example, *collaborative customization* deals with customer sacrifice when forced to make difficult and multidimensional *either-or* choices, such as length versus width, complexity versus functionality etc. *Adaptive customization* deals with the

customer sacrifice when customers are presented with too many finished goods or too many component parts and must engage in an unwieldy *sort-through* process; for example, this is exactly the case while selecting an appropriate automobile for oneself. The companies can exploit *cosmetic customization* when customers sacrifice not on the basis of a product's functionality but its form —in terms of how it is packaged or presented, where and when it is delivered. The fourth kind of customer sacrifice, called *repeat-again* sacrifice occurs when customers must perform the same task or provide the same information repeatedly. In such cases, *transparent customization* does the trick. In short,

"manufacturers and service providers must discern the uniqueness of their offerings, ascertain the sacrifices their customers current experience and then identify which form of customization will yield the best results. Often a combination of approaches is needed to address complex sacrifice issues"

[Pine and Gilmore'99].

Businesses entering the Experience Economy must recognize that each and every action contributes to the total experience being staged. Where for them, theater is not a metaphor but a model. In their book they seek to focus attention on the quintessentially dramatic nature of an enterprise. Thus literally meaning: Work is theatre.

The following two cues *"In the emerging Experience Economy, any work observed directly by a customer must be recognized as an act of theatre"* and *"with theatre as the model, even mundane tasks engage customers in a memorable way"* form the basis of our work, where we try to argue that **in the emerging experience economy the mundane task of sales advisory has to be considered as an act of theatre in order to engage customers in a memorable way.**

Yet another *cue "technology-mediated interaction also presents a bare stage for business theatre",* which influenced this present work came from Brenda Laurel. In her book *Computers as Theatre* [Brenda Laurel'93] provides a detailed application of Aristotle's philosophy to computer-based performances, believing human-computer interaction should be a "designed experience". There she also defines principles and techniques using computers as a *medium* rather than an *interface*. She describes this technological stage as follows *"Thinking about interfaces is thinking too small. Designing human computer experience isn't about building a better desktop. It is about creating an imaginary worlds that have a special relationship to reality —worlds in which we can extend, amplify, and enrich our own capabilities to think, feel and act"*. This clearly depicts that she truly believes that working with computers is —or at least should be—theatre. Thus motivating us to define **sales advisory** in the context of Experience Economy as **"business locus of vendor customer (designed experience) interaction"**.

4.4.5 Sales Advisory: A Designed Interaction

Interaction *is two or more people having an exchange of ideas, of emotions, of physical objects, of words etc. On the computer, interaction is still two or more people having an exchange, except that the interaction is mediated by technology.*

[Flemming'98]

This forms the basis for our effort of designing the interaction in the automobile sales advisory context. But before we actually do so, let's have a look at the **sales process** or to be precise the **automobile sales cycle**. From time to time, one has seen varied sales cycle/process definitions. It is nothing new, and while the words may differ, most often, these definitions come down to the same thing. For our purposes, we will keep it simple: **A sales process is a sequence of sales activities** which constitute the following five basic phases: *Prospecting, Qualification, Discovery/Needs Analysis, Solution/Proposal and Implementation/Close.* One more type of basic phase which is typical for automobile industry or in general for industrial goods could be the *After-Sales/Reinforce* (services, complaints, repair/workshops) phase [Fig. 4.9].

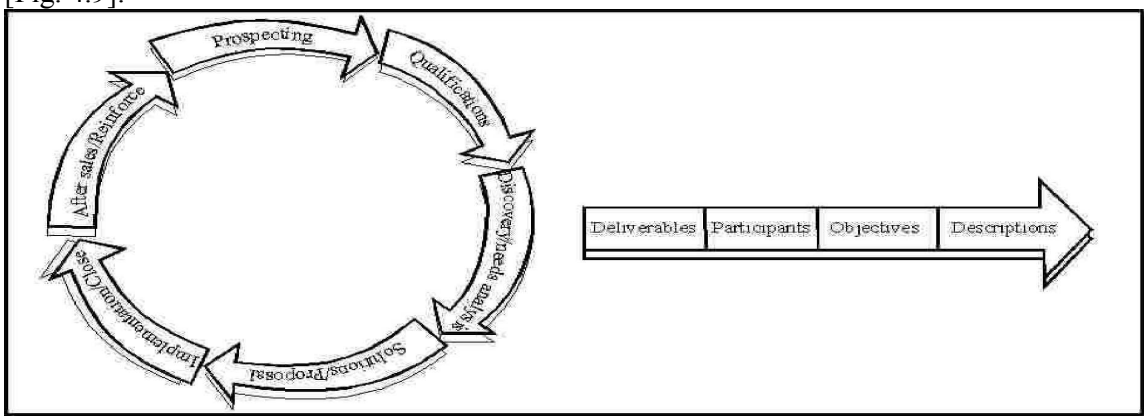


Fig. 4.9: Sales Process with its different phases

Within each phase there may be several steps. The number of steps will depend upon what is unique to the business, the complexity of sale and what is considered standard selling practice. Usually following four elements are considered in each step: *Description, Objectives, Participants and Deliverables*. An example of a step from the automobile sales would be:

- *Description:* fill out a qualification questionnaire for the (specific car model buying) opportunity
- *Objective:* is to determine early cycle qualification in regard to the need and ability to pay.
- *Participants:* include customer, sales representative and credit sources.
- *Deliverables:* would be the qualification questionnaire and credit reports.

The successful sales process can be analyzed and explained from different perspectives (Schoch'69; Weiz'81; Wotruba'81; Churchill et al.'85; Kramer'93). Here central to these different perspectives are the many success factors which can be differentiated in four fundamental directions. The first direction (*seller oriented*) explains the success of the sales process on the basis of the personal traits of a sales person. The second approach (*behavior oriented) explains it from the perspective of the activities of the sales person. The third approach has the* customer central to its observations (*customer oriented*). Finally, an extended approach represents the sales process as a social interaction process (*interaction oriented*). In this approach the sales process is defined as **a sequence of sales activities which not only depend and orient on each other but also influence each other.** This means that the seller and the buyer involved in the sales process are not to be considered as isolated individuals but as the members of a group in the larger context i. e. in their respective environments.

4.4.5.1 Situated Component of Sales

In addition to the above mentioned approaches to explain the successful sales process, in recent past many approaches have cropped up which focus on the situated component of the sales in their observations [Weiz'81, Kramer'93]. Such a context dependent observation of the sales process has following three goals [Kramer'93]:

- The classification of alternative sales behavior
- The classification of context factors, which influence the successful sales, such as
- an attempt to put both the classifications in relation to each other

The comparison of sales behavior and context variables has a goal to derive sensible suggestions for the sales advisory under different situated general conditions. According to this representation one interprets the *sales process as a social interaction process bounded to a situated context*. Apart from the specific sales situation, which can be characterized by the sales object, the organizational environment plus the social and cultural environment of the sellers play an important role [Mefert'98] [Fig. 4.10].

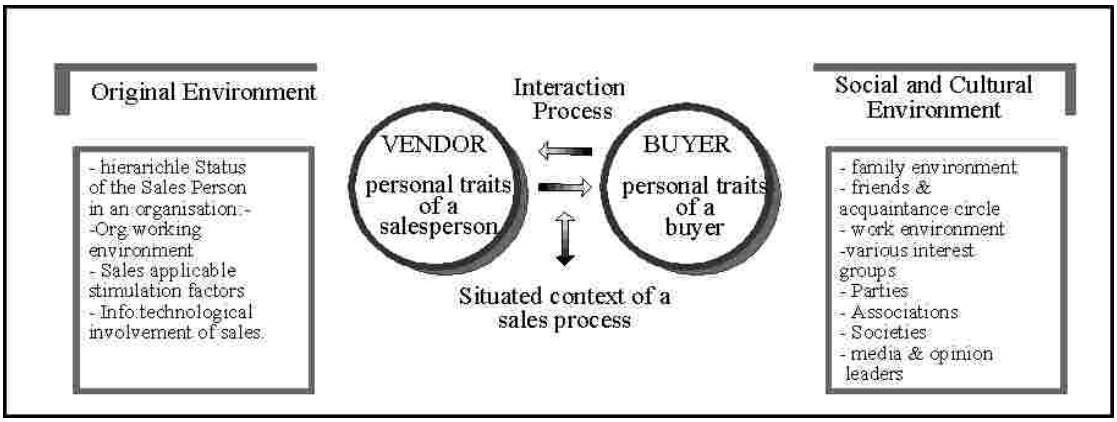


Fig. 4.10: Sales Process as a social interaction process bounded to a situated context (adopted from Mefert'98)

Further according to Meferet [Mefert'98] "The result of the sales processes can be determined by all the elements of a communication process: characteristics of an organization as "Sender", characteristics and behavior of a salesperson as "Medium", characteristics of a customer as "Receiver" and finally the quality of the sales arguments as "Messages".

4.4.6 A Process Based on Interactions

Sales Advisory is a complex multidimensional process, where the task is to deliver expertise - regarding the product features, utilization, services wrapped around the product & benefits - to the customer in such a manner that it impacts the customer positively i. e. it adds more value to the customers needs/ wants. This process is based upon a series of interactions among the sales person (expert) and the customer and can have five/six different stages corresponding to the stages of the buying cycle. In effect this process also becomes a loop -- but one that need not be completed before it begins again [Fig. 4.11].

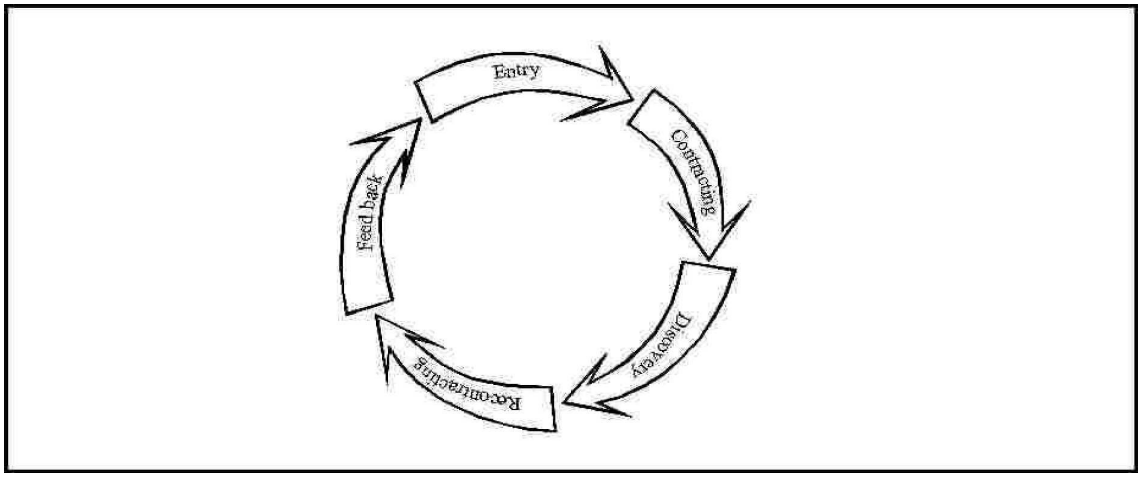


Fig. 4.11: Different stages of the Sales Advisory Process

To successfully define the interactions in this advisory process two basic questions are helpful; where in loop is one at present? and second where does one wants to go from here? Further, at each stage to add more value *(as selling added value and not just selling value added products and services will be the key success factor to remain competitive in the coming era.....Sellution White Paper)* it is important to know the tasks involved and the type of relationship one has with the customer during the advisory process. Here, tasks depend more or less on different phases of the sales process and the customer-advisor relationship can take any of the following three roles:

- Helping Role: where problems and solutions are identified by the customer with the help of the advisor
- Expert Role: where both are identified by the consultant
- Cooperative Role: In this situation both are identified by the customer and the advisor

In this arrangement two extremes would be that where the customer identifies the problems and wants the consultant to identify the solution here the customer is asking the consultant to take both a helping and an expert role simultaneously. This leads to blame the advisor when outcomes are less than desired. The ideal relationship to impact the customers values and the advisors expertise to be well used would be to have a *cooperative relationship*. Thus the exact nature of the relationship can be established by having a shared understanding of the wants/needs of the customer. This leads further to model the sales consultation process interactions as REQUEST-REPLY DISCIPLINE, which is analogous to the ideas presented to characterize the interactions between programs which support some service at various network sites[RFC 722]. The interactions here are fairly simple and follow the pattern as mentioned below.

”customer and vendor(expert) establish a channel by agreeing upon some external means. A single interaction among them begins with the customer issuing a REQUEST. The vendor(expert) receiving that request, issues a REPLY. The customer interprets the reply sequence to determine whether the request was successful, failed or partially failed and takes appropriate action. Such a sequence of events is termed as SALES-DIALOGUE (Fig. 4.12)”.

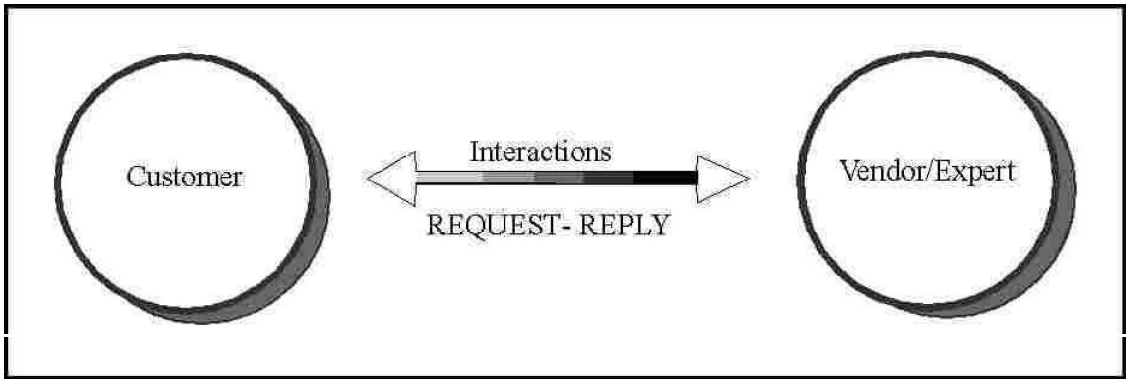


Fig. 4.12: Sales Dialogue

4.4.7 Turning Interactions into Experience:

When do simple interactions turn into an Experience? A simple answer to this question may be that by appropriately designing the interactions one can turn them into experience. This alone does not suffice to provide the consumer a rich and satisfying experience in a technology (computer) mediated environment. Here apart from function (interaction design), two other factors like structure (information design) and style (sensorial/visual design) play an important role. Thus leading us to yet another definition of the experience, which states that "these three elements come together in an elusive space and where they overlap is the "experience" (fig. 4.13).

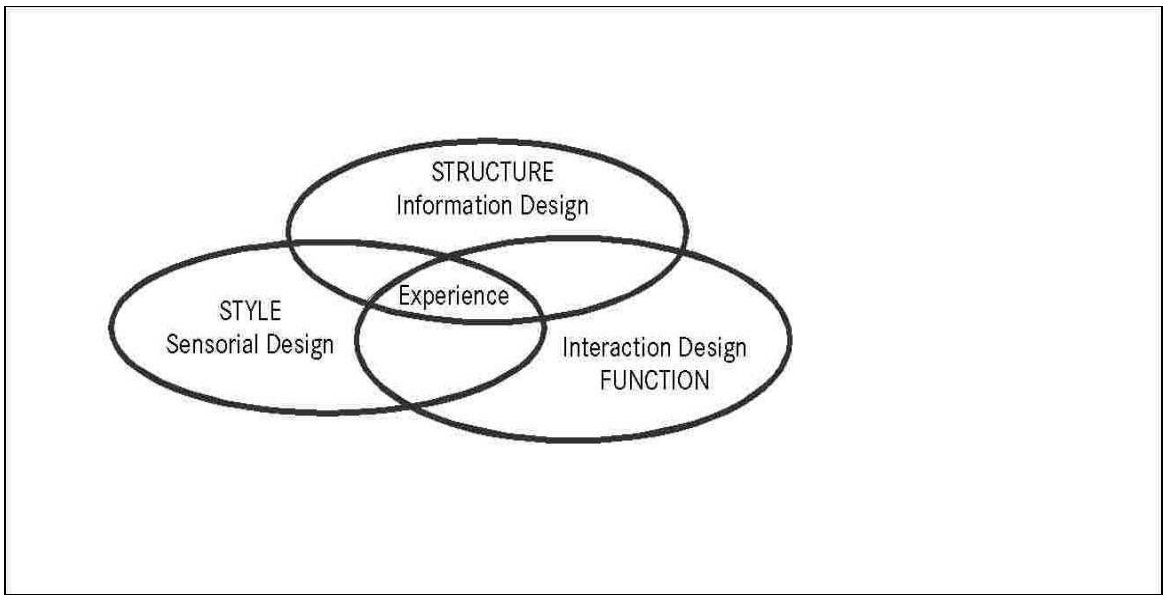


Fig. 4.13: Experience as an Intersection of STRUCTURE, FUNCTION & STYLE

In Fig. 4.13 interaction design depicts the FUNCTION and provides an answer to how the user engages and responds. Well designed interactions give the user what they need clearly and naturally. Here one has to remember that this is a new medium and though we have been interacting for thousands of years but we still bring the same toolset to problems i.e.
what can we see and hear

what can we imagine and reason
what can we absorb and use.

The information design element refers to information architecture or STRUCTURE. It provides the organization of information and the blueprint (the framework) for the application (advisory system). Further it provides the reduction and simplification of the complex or in other words it is the skeleton.

The sensorial/visual design implies the STYLE which consists of the look and feel elements i.e. it includes all the visual factors like, color palette, typography, imagery, video & audio. Finally,

A successful design of experience, one that will invigorate(enliven/stimulate) or encourage and add value to the user's life, balances cognition, emotion, perception & knowledge.

4.4.8 Personalization

"personalization enables the closest computerized approximation of face to face buyer-seller interactions"

personalization.com[1]

The advent of the internet brought dramatic changes in the way we get information and interact with others, especially in competitive business environments. The internet is turning the interactive elements of the sales process in a totally new direction i.e. personalized shopping experience.

One way for businesses that want to take advantage of these opportunities would be to individualize the interactions between the (potential) customer and the seller (web site). The single most important way to provide value to customers is to know them (i.e. know their needs) and serve them as individuals by providing tailored products and services. Thus providing a customer a unique personal feeling.

Currently, individual commerce sites provide some measure of personalization in the form of recommendation systems based on the buying history of the customer and are limited to only one product. But in future, personalization won't be limited to just one product category (e.g. books, CDs, computers). Instead, consumers will be able to find an online seller who sells a particular lifestyle, defined as a mix of products (books, clothes, cars, computers, home furnishings, personal care products etc.)and services (finance & leasing, mobility & telematics services).

The seller will be able to deploy a wide variety of technologies in order to reach the target customer (i.e. text, graphics, audio, video, push, chat, discussion etc.) and can create an online experience that correlates with the customer's personal aesthetics, sense of taste and desired level of interactivity.
In this context, three major tasks involved in the personalization process that creates an application with personalization features are:

[1] http://www.personalization.com/basics/faq/faq5.asp

Acquisition is the task to identify information that is available about environment and users (i.e. their characteristics, preferences etc) and to make this information accessible to the personalization component of the application and finally to construct an initial user model or usage model.

Representation and Inferencing are the tasks to represent the user model or usage model in appropriate ways to allow access and inferences and to infer further (secondary) assumptions about user and/or user groups, their behavior, and their environment, whereby integrating information from various sources.

Production is the task to generate the personalization of content, presentation and modality or structure based on a given user model or usage model.

As the details regarding these tasks do not fall under the scope of this work, however, interesting examples of all these tasks can be found in the literature[Phol'98], [Maes'94], [Balabanovic'97], [Pazzani &Billsus'97].

Apart from that, one interesting fact which has a direct relevance to our work is the work reported in [Phol'98 & Kosba et al.'94], where the use of knowledge-based (particularly, logic-based) methods in user modeling systems to maintain assumptions about user beliefs (i.e. the user's familiarity with the concepts of the topical domain) is based upon the same principles as ours i.e. representing both assumptions and domain knowledge using a concept formalism (a, KL-One derivative). Thus stressing upon characteristics: explicit & re-usable representation.

4.4.9 Summary

In this fast changing world and due to the technological advancement the technical/qualitative differentiation of the product is no longer feasible. Thus future product differentiation will not only be based upon extended products and services but on the experiential aspect which their good encompass. Thus, more and more companies would need to wrap experiences around their existing goods and services to differentiate their offerings. In addition to that companies will have to concentrate on more than one aspect of the experience. In case of automobile industry, apart from the *driving experience*, one would enhance the task of ***sales advisory*** as a ***memorable buying experience***.

In this section, starting from the different definitions, thoeries of experience and its role in the future economy we deduce the task of sales advisory/consultation to be of designing interactions and designing in such a way that finally these interactions turn into a memorable buying experience.

4.5 Specifications for the Consultation Systems:

in this section we provide some specifications to improve upon the consultation systems. The first set of specifications deals with the improvements in general and the second set provides the improvements regarding the knowledge representation.

4.5.1 In general:

while designing the consultation systems which support the sales of a complex end product, the following points must be considered:

- On one side, they should offer an appropriate support to the user (sales person/consultant) of such systems. That means, they should enable the sales person to carry out the consultation, better and faster. Thus, the problems of the sales person, which can arise during the sales consultation with the customer, are to be considered. From this, some of the suggestions for the consultation systems can be deduced.
- The consultation should be fluent, i.e. the dialog between the sales person and the customer should be interrupted as little as possible. The product vehicle is rather complex, there are lots of model variants, lots of accessories and quite a large number of manufacturing constraints which are to be considered while configuring the vehicle. Thus, the sales person has to continuously look manually for the appropriate information in the sales documents /manuals.
- The customer expects a competent partner in a sales person, which posses the necessary expert knowledge about the complete product range and helps him to translate his wishes to the available product components. The consultant should not only be informed about their own product range but should have knowledge about the offers and developments of the competitors products as well. Here also there lies a problem of a large number of models and accessories and the recency of this information.
- The sales person must continuously check whether the chosen configuration of the vehicle is technically possible (manufacturable) or not . The individual accessories are connected to each other with lots of relationships. Different accessories can be fitted only in connection with certain other accessories. The installation of some accessories can cause the replacement of another accessory or there occurs a price change . As a rule during the sales consultation the technical feasibility is checked once again.
- During the consultation the accessories are being continuously changed and thus the price of the vehicle also changes. In order to give the customer the actual price the sales person has to continuously extend or change his/her price calculations. This is rather difficult, if the price of such an accessory changes which is dependent upon the presence of a specific accessory.
- The sales person should always be in a condition to advise the customer that what accessory is going to fulfill which of the customer's wishes.

On the other hand, developers of such systems face certain problems, whose solution is important to their actual realization:

- The acquisition of product knowledge.

- There are often fast and unexpected changes in the product knowledge. The prices for accessories often change. New accessories are included in the product range or certain old ones are taken out (Actuality, maintenance of the knowledge base), sometimes they even change the structure of the knowledge.

- The complexity of system extension and maintenance: normally, it is difficult to add new knowledge areas into an existing system. For example, in our case, it would be sensible to add a financing component and the integration of appropriate knowledge for such systems.

- The knowledge should be represented in such a way that a product expert should be in a condition to understand the represented knowledge. This kind of high level

specification can only be achieved through an appropriate user interface. The goal here is that later an expert is in a position to make the changes and extensions on his own.

- The queries for the represented knowledge should be implemented efficiently.

The major portion of the above mentioned problems are being solved by the proposed consultation system. Here, the employed knowledge representation system plays an important role.

4.5.2 Regarding Knowledge Representation :

the knowledge represented in a consultation system should have the following characteristics:

- Object centered: all individuals should have a unique, intrinsic and immutable identity obtained at time of creation.
- Terminological: the system should support the definitions of complex ”Noun phrases” in the form of concepts (and the discovery of their inter-relationships); these concepts should then be used to make assertions about objects.
- Deductive: the consultation system should not be a passive repository for unconnected assertions, like a relational database; the system should actively search to find an entire class of propositions entailed by the facts it has been explicitly told.
- Incremental: partial, incomplete descriptions of individuals should be acceptable.
- Knowledge retraction: the system should track dependencies between facts and allow certain facts to be retracted.
- Rules: the system should be able to support rules (forward or backward chaining), whenever appropriate individuals are found.
- Procedural tests: complex concepts, not otherwise expressible in KR systems, should be described procedurally in the host language so that individuals in these classes can be recognized.

4.6 Conclusion

In conclusion to this chapter, I would like to state that whatever realm, paradigm, philosophy and methodology one chooses to conceptualize the advisory systems for future, the ultimate success of these systems will very much depend upon the knowledge they contain, its organization, its structure (representing) and the capability to manipulate/infer(reasoning) upon this knowledge. Thus, following chapters of this work/case study are dedicated to investigate the relationship between the principles of Knowledge Representation and Reasoning systems and their embodiment in working applications.

5 Domain Modeling

After presenting the basic concepts of the knowledge representation with the help of terminological formalism (Description Logics) in the previous chapter; in this chapter we would like to describe how the knowledge of a Coach-world was represented. First we would deal with some basic questions and then explain in detail the modeling of the domain. While modeling a knowledge area in the terminological system lots of questions arise, which can only be answered in the context of the developing application. Next section deals with some such questions. To this counts such principle decisions like at what level we represent the information as objects and as concepts. Furthermore, different points concern the choice of appropriate language expressions and the granularity of representation, i.e. how "deep" should the modeling be and the syntactic point of view.

5.1 Design Decisions

In this section certain distinct design decisions are presented which support the realistic knowledge-intensive applications in DLs such as ours (sales advisory). First, the practical consequences of the distinction between concepts and instances, second the difference between primitive and defined concepts and finally the perpetual issue of when to make something a concept or a role, are discussed. These topics, according to the KL-ONE philosophy, are basic organizational issues and from their discussion, one can derive the guidelines for organizing and structuring the knowledge to be represented in DLs.

In the recent past the literature has started mentioning about these issues with respect to applications. One example of such an approach is an extensive survey of the use of CLASSIC in several applications [Brachman et al. 1992]. Another example [Speel 1994]. The issues discussed in this section do not give us a complete methodology for knowledge acquisition and the application of DLs. However, the practical approach taken here helped us in the difficult task of collecting, defining, structuring and representing knowledge about a complex domain of sales advisory in the automobile sector.

5.1.1 Concept vs. Instances

one of the fundamental problems while modeling a domain in description logics is the distinction between the terminological and assertional knowledge. This distinction was proposed by Woods [Woods 1975] and Brachman & Levesque [Brachman & Levesque 1982]. Through this division, modularity of domain knowledge is increased. This leads to a knowledge base which is organized transparently and therefore is easier to understand and to maintain [Swartout and Neches 1986]. It is not clear what knowledge belongs to the technical vocabulary (terminology) of an application domain and what to the facts (assertions) of that domain. This has been stated by MacGregor [MacGregor 1991b] as follows:

“However, the formal distinction between terminological and assertional knowledge [...] is somewhat esoteric. Our experience with NIKL and LOOM suggests that drawing such a distinction confuses (all but the most sophisticated) users as often as it helps them”

[MacGregor 1991b, PP 393-394]

The problem here is that mainly "lexical definitions" are extracted from the domain of interest and stored in terminological KBs. However, a lexical definition (a lexical definition is used to report the customary or dictionary meaning of a word, and a stipulative definition is used to

establish or announce or choose one's own meaning for a word [Robinson 1972]) is an assertion that certain people use a certain word in a certain way [Robinson 1972]. The unavoidable use of assertions in terminological definitions gives rise to this confusion. This means that, detailed, subtle decisions need to be taken whether particular assertions are to be used as definitional properties [Speel 1995 pp73-74].

Even though these modeling problems remain but the distinction between terminology and factual knowledge often results in transparency when complex domains are studied. Many examples illustrate this. Other examples where division works out well in practice are medical applications, such as UMLS [Humpherys, Lindberg & Hole '91]; [Lindenberg & Humpherys '90]; [MacCray & Hole '90]; [Rector '93a, '93b]; [Ensing et al.'94] & Speel et al. '91b,'91]. In addition, conceptual terminologies have been developed in the fields of design [Alberts '93a, '93b] and engineering [Gruber & Olsen '94]; [Top & Akkermans '94]. These efforts have resulted in a new discipline in the knowledge technology field, namely "ontology development" [Neches et al. '91]; [Gruber '91]; [Gruber, Tannenbaum & Weber '92]; [Mars '94b].

Though the work on formal distinctions between various non-logical symbols is very important, the application of this theory to realistic, technical domain, like sales advisory in automobile sector, seems to be very difficult. Therefore, a more practical approach for distinction, as described by Brachman [Brachman 90], is followed:

"According to the model theoretic semantics a concept denotes a class of objects and an individual denotes a single object.
Individuals have unique identities and are countable. An individual can be described by concept expressions that apply to it, but there is a uniqueness assumption that guarantees that two individuals with different names (even with the same description) will be different individuals.
Another point is that the facts in the world can change and thus individuals can change, too. In contrast to individuals, concept definitions and their relationships to each other do not change. A more subtle issues is that retraction and addition of facts about individuals do not change the concept classification hierarchy. That means, individuals and their classification can change through assertion and retraction of facts; thus the concept hierarchy is immune to changes in individuals. From this one could conclude that the concepts are not classified based on properties of individuals; If that would have been true then due to a change in the object, which is the part of a concept definition, the concept had to be classified a new. In contrast to that an individual is classified based on the known properties of all individuals, including its role fillers."

In our domain of sales advisory in the automobile sector, different models of coaches, trucks, cars and their respective accessories constitute the main chunk of the knowledge. From this one can formulate the following question:

> At which abstraction level should the vehicle models (car, trucks & coaches) and their respective accessories be represented as concepts and at which level as objects?

Answer to this question depends upon how this model is going to be used i.e. which goals are being pursued by the application that is utilizing this knowledge base. In our application, we want to differentiate among the vehicles (car, trucks & coaches) of an individual customers. Thus, every configured and sold vehicle must have a unique identity. Apart from that, we

want to provide the consultant with the possibility of describing the accessories individually for each vehicle (car, truck or coach). For example, the consultant should be able to assign the accessory "seatcovers" a "color" and "material" . Another aspect comes into picture if the proposed advisory system is to be interactive. Then it is sensible to provide the user (customer/consultant) of the system with a list of possible choices (e.g. **oneof** different motors).
For further discussion of this problem, let us introduce a concept of accessory and the horsepower (hp).

Example[1]

```
% accessory is a primitive concept
        accessory  :< anything.

%hp is a primitive role which can have maximum one rolefiller.
        hp  :< domain(motor) and range(number)   type feature.
```

The next example should explain our decision to model certain entities as objects, as concepts and some at still higher level as meta-concepts to show that this depends upon different aspects and which view is appropriate according to the intended usage.

The coach series O404 has three different motors which differentiate according to their respective performance and the models are: OM401 with 290 hp, OM441 with 340 hp and OM402 with 381 hp. We now have two basic ways to represent such a piece of knowledge:

1. We introduce a concept motor and describe the individual motor models as objects with specific horsepower(hp) as it's characteristics , as follows:

```
motor :< accessory and exactly(1, hp).
```

Following is the description of individual motor types, i.e. creating the objects:

```
om401  ::  motor   hp   :  290.
om441  ::  motor   hp   :  340.
om402  ::  motor   hp   :  381.
```

With this representation a query about the different models of the motor can directly be answered with the help of A-BOX query language of BACK and looks like this:

```
<< getall(motor).
>>[om401, om441, om402]
```

This leads to that, that many coaches, as long as they have a same motor type, can have a relationship to the same object. Thus individual description of each accessory is not possible. Now, if we change the description of this motor concept, then it will have consequence for all the coaches where this motor has been fitted in.

[1] Note: For the ease of reading I'll follow the examples from the coach world.

Example:

```
% A coach has maximum one Buyer
        buyer :<  domain(coaches) and  range(string)  type feature.
%  A coach has maximum one motor
        coach :< anything
          and exactly(1, has_motor).

        coach1  ::  coach
          and  buyer   : customerA
          and has_motor : om401.
        coach2  ::  coach
          and  buyer   : customerB
          and has_motor : om401.
```

Suppose, we want to allow that one can give discount on each accessory depending upon an individual customer. Normally price negotiations are not performed like this but this example allows us a simple representation of the problem. One way to represent
"Customer A should get a discount for DM 2500 on a motor" could be as follows:

```
discount_in_DM  :< domain(accessory)  and   range(number)  type feature.
om401   ::   discount_in_DM  :  2500.
```

Naturally, this is going to fail, as we've attached discount to the object (motor) which for many customers is a rolefiller for the role has_motor. Thus, every customer will get the same discount. One can think of attaching the discount relation not to the individual accessory (motor) but to the coach (vehicle itself):

```
motor_discount  :<  domain(coach)  and range(number)  type feature.
```

Accordingly, the following description is going to lead to a success

```
coach2  :: motor_discount  :  2500.
```

This is still not satisfying, as discount is given on motors and their description doesn't say anything about it. An outcome of this would be that we'll have to introduce a discount relation for each and every accessory. As this problem is going to be there for many other relationships as well, thus following such an implementation we'll get a very complex structure.

1. One way to solve such a problem would be to introduce the reference objects for each accessory. As a new coach is created then to make a copy of that object and later work with that copy. Normally KL-ONE systems do not posses such functionality.

2. In addition to the general concept of motor, we introduce now sub concepts for each motor type. For such a representation, the respective objects are created as instances of the sub concept on the fly(during runtime of the consultation system).

   ```
   motor  :<  accessory and exactly(1, hp).

   om401 :< motor and hp  : 290.
   ```

```
om441  :< motor and hp   :  340.
om402   :<  motor and  hp   :  381.
```

Now, if we want to furnish a coach with a specific motor type, we'll have to create a new object of that desired motor concept. For the illustration purposes, we use the same example as above:

```
coach1  ::  coach
  and  buyer  :  customerA
  and has_motor : om401.
coach2  ::  coach
  and  buyer  :  customerB
  and has_motor : om401.
```

Within this description two new motor-objects have been created. As these motor objects are unique, thus the relation discount_in_DM can also be interpreted as desired.

In such a modeling, a different way to answer the query for the different motor types should be chosen. As there are no objects present before the descriptions of the coaches, thus it is not possible to make use of the query language for the object level. Though with the help of the subsumption relation one can determine which concepts subsumes the concept motor i.e. which other concepts are there in our taxonomy which are consistent with all other constraints in actual A-BOX. In the above very limited taxonomy we obtain the names of the three motor concepts as well.

This does not represent a general solution. It is not difficult to imagine that in our knowledge base there exist certain concepts which do not belong to the set of expected solutions but still are specific to the motor concept.

Example:

```
consumes  :< domain(motor) and range(oneof ([normal, super, diesel]))
                type feature.

motor :<  accessory.

petrol_motor  := motor and all(consumes, oneof([normal, super])).
diesel_motor  := motor and consumes : diesel.

om401  :<  diesel_motor and hp : 290.
```

In this terminology, the concepts petrol_motor and diesel_motor are specific than the motor concept. The experiment to find only the most specific concept under motor is problematic as to the extension of the terminology. For example, the concepts which describe the motors exactly could be of interest to the production but may not be of interest to the customer and which could further lead to that the desired solutions cannot be found. The position of the concept in the inheritance hierarchy cannot be used to answer the previously posed query as long as the knowledge base is used in the arbitrary context i.e. either for sales or for production. Here we want to restrict ourselves to the essential knowledge required to support the sales process, thus the problems described later do not pose at all.

The problem of viewing concepts from different levels as objects is described by Brachman [Brachman 1990] as follows:

"Whatever level we fix for our individuals, any other descriptions in the domain that could be considered individuals from some other point of view can be handled in one of two less-than-ideal ways. First, they could simply be represented as concepts.....An alternative would be to allow both objects to be individuals. But since CLASSIC does not currently support a "meta-description" facility, this representation would be incomplete in an important way, in that CLASSIC would maintain no relationship at all between the two individuals."

The problem mentioned in the quotation was posed during the implementation of BACK as well, and there also was no such meta-capability provided.

With this, we have posed a requirement that accessories are to be viewed as objects on two different levels of abstractions. However, this can not be completely represented in the present day terminological systems as already stated. The second view, as introduced above, represents a better solutions for our affairs. This allows, above all, the separation of factual knowledge from the domain knowledge. The T-BOX-level then serves to stipulate our knowledge about accessories and vehicles in general. These concepts are instantiated only if there is a concrete consultation problem to be solved .

5.1.2 Primitive vs. Defined Concepts

The benefits of the classification and it's context with the declaration of defined concepts has already been described. The concrete decision, whether a given concept can be introduced as primitive or defined depends heavily on the intended application and cannot always be made unequivocally.

Thus, for example a concept drivers_door defined through the mounting_position can be introduced as follows:

```
drivers_door  := exit
    and location : front
    and orientation : left.
```

The drivers door is always on the left hand side of the vehicle. In this case, however it is also clear that this type of model can only be employed where there is a right-hand traffic system and thus the steering-wheel is on the left side of the vehicle. Another possibility could be the definition of equal concepts, without using position as a sufficient condition:

```
drivers_door  := exit
    and  for : driver.
drivers_door => location : front.

drivers_door and all(accessories_of, left_hand_drive) => orientation : left.
drivers_door and all(accessories_of, right_hand_drive) => orientation : right.
```

These two examples show that it is very difficult to find the sufficient conditions for individual concepts and at their formulation one has to be careful about their inherent meaning. In case of confusion these types of concepts remain primitive.

The reasons to introduce a defined concept can be of different nature. The simplest reason for this is that the representation language is expressive enough to allow a necessary and natural description of the concept. Another reason is to be found in that the objects on the basis of typical characteristics are to be recognized as instances of concepts, even though not even a single sufficient condition is given. For example, we could express, a vehicle on the basis of it's motor could be assigned to a model_series. This attribute does not pose a sufficient condition, as the same motor can theoretically be fitted in a different vehicle as well. However, it is possible to introduce a concept that contains this characteristic as sufficient condition and simultaneously it could be used as the premise of a rule, whose conclusion is the desired model_series:

```
o404  : coaches
  and exactly(1, has_version, motor)
  and  exactly(1, has_version, gears)
  ...
  .
o404_label := coaches
    and all(has_accessory and range(motor), om401)).
o404_label => o404.
```

While modeling the coach world, we introduced as many as possible defined concepts to increase the structure of the knowledge (Fig.5.1).

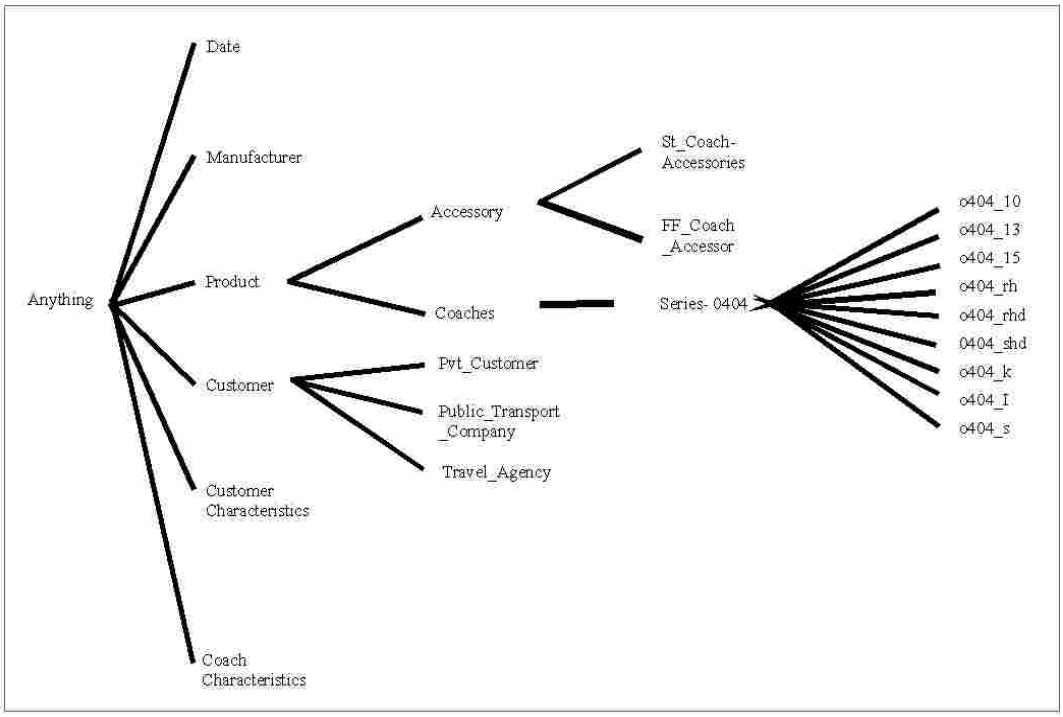


Fig. 5.1 represents a cross-section from our coach worl

5.1.3 Concept vs. Roles

The decision whether something should be a concept or a role is not always clear [Brachman 90]. Major portion of the coach-world could be represented explicitly (eindeutig). All accessories and vehicles undoubtedly were identified as concepts and all attributes e.g. horsepower(hp), transmission of the gears etc. as roles. However, unclear is the representation of the terms "factory fitted accessories" and "custom accessories".

To explain this question we must first define what is to be exactly understood by factory-fitted and custom accessories. In our domain factory fitted accessories are all those components which are included in the basic price of a vehicle and do not have the price of their own. Custom accessories are those which either replace the existing factory fitted accessories (in such cases they can have -ve price, i.e. decrease in price) or supplement the vehicle (in such cases the price is always +ve, i.e. it costs extra).

Now, one could think of introducing these terms as concepts while using price as an attribute. Following example should make this more clear:

```
factory_fitted_accessory := accessory and all(has_price, 0).

accessory and all(has_price, gt(0))  =>  custom_accessory.

accessory and all(has_price, lt(0))  =>  custom_accessory.
```

The definition of the concept "custom accessories" should be clear now. The gt-expression creates an interval for all numerical values greater than zero and lt stands for values less than zero. Here, we've used the I-links to represent this, because the disjunction in concept definitions cannot be represented and as a result to that there is no unequal-operator in BACK. Following example shows the application of such concepts.

```
o404 :<  coach
  and all(has_accessory and range(abs), factory_fitted_accessory).
```

Or as an implication link:

```
abs and all(accessory_of, o404)  =>  factory_fitted_accessory.
```

The natural language interpretation of the above mentioned definition would be: " o404 is a coach and all its accessories which belong to the category of "abs" are factory fitted accessories. Where as the rule in natural language expresses that "every object which is an instance of a concept abs and accessory of o404, will also be an instance of concept "custom accessory" and thus gets the price zero"

Now a query like "whether abs is a factory fitted accessory in o404 coach" can be checked through the subsumption relation.

```
abs and all(accessory_of, o404)  ?<  factory_fitted_accessory.
```

One could think of expressing this through a role relation as well, as both these concepts imply the existence of a vehicle. That means no accessory in itself can be a factory fitted

accessory, only in the context of a vehicle it makes sense. Thus, the definition of necessary roles could be as follows:

```
has_accessory :<  domain(vehicle)  and  range(accessory).
factory_fitted_accessory  :=  has_accessory  and range(all(has_price, 0)).
factory_fitted_accessory_of := inv(factory_fitted_accessory).
```

The definition of factory fitted accessory represents that all possible rolefillers must be accessories and should have the price zero[1]. The definition of a role for custom accessory is not possible without using the disjunction or Role-Value-Maps(rvm) (which do not exist in BACK in this form). This serves to put the set of values of rolefiller of two roles to equal.

```
all(has_price, gt(0))  =>  rvm(accessory_of, factory_fitted_accessory_of).
all(has_price, lt(0))  =>  rvm(accessory_of, factory_fitted_accessory_of).
```

Yet another example of the usage of role:

```
o404 :< exactly(1, factory_fitted_accessory, abs).
abs and all(accessory_of, o404)  ?<  all(factory_fitted_accessory_of, o404).
```

At this point one should be able to differentiate between the later application of the knowledge base (i.e. for sales or production). We want to show to our customers that which all factory fitted accessories are already there in the standard version of the vehicle. Representation as a role can be advantageous if the precondition that all the accessory objects have already been created and attached to the vehicle concept is fulfilled. While representing this as a concept, the wrong usage (interpretation) of the term is possible:

```
<<  getall(factory_fitted_accessory).
```

This query gives all the accessories which were ever fitted into any vehicle (pre-condition to that is that we keep many vehicles simultaneously in the working memory) and which have the price zero. To avoid such misunderstandings we have opted for the representation as a role.

5.2 Domain Modeling in BACK

This section concerns about setting up a model of the problem domain. Our domain model consists of three parts: (1) an ontology describing the relevant classes with their attributes as well as simple integrity constraints that are to hold between them, (2) assertions about objects and relationships between them, and (3) a set of monotonic inference rules, by which additional relationships between objects can be derived. The ontology and assertions are the static part of domain model and the rules belong to the dynamic part.

Further, here we would like to provide with the details of the modeling of our domain of sales advisory in the automobile sector. Here, we'll explain why the decisions were made in favor of one or the other form of representations. Further, especially discuss the fundamental and problematic portions of the modeling and make it clear with the help of examples.

[1] note: the range of the role factory fitted accessory is given by the conjunction of concepts accessory and (all(has_price, 0))

The "Berlin Advanced Computational Knowledge Representation System" BACK has been developed as KL-ONE-based hybrid reasoning system. (see section 4.3: what does our KB consists of?). Before we go in to details of actual modeling, we would like to present some basic modeling constructs in BACK.

Concepts: In BACK system, a concept serves two main purposes.

1. It serves in indexing the instances of the concept.
2. It serves the purpose of describing the types and cardinalities of relations the instances of the concept can and must be a part of as given in the concept's role restrictions, i.e. a set of rules governing the behavior of the instances with respect to other instances of the model [Woodfin '93].

As an index to instances, the concept name can be used in retrieving the instances of the concept. When instances are created and added to the model, they are attached to the internal representation of the concept that best describes them. BACK allows the user to use the same name given in the definition of the concept in retrieving the instances attached to the concept's internal representation.

As a set of role restrictions given in TBox concept definition, the behavior of concept's instances with respect to other concepts instances in a larger hierarchy of concept is defined. The three BACK language mechanisms that can be used to give a role restriction in a concept definition are **atleast**, **atmost** and **all**.

Roles: BACK Roles allow the user to define the domain and range relations between concept instances. A typical role definition would state the domain with language element domain and the range with range. The domain must be a concept. The range can be a concept or an attribute type. In BACK V5 defined roles are permitted as are the role defining operators for the inverse, transitive, composite and the closure of role filler sets [Quantz'90]

5.2.1 Modeling in Detail:

Let us start with the most general information which is to be represented in the knowledge base. First of all we take the cross-section of the domain which we want to represent i.e. the coach world (Fig. 5.2)

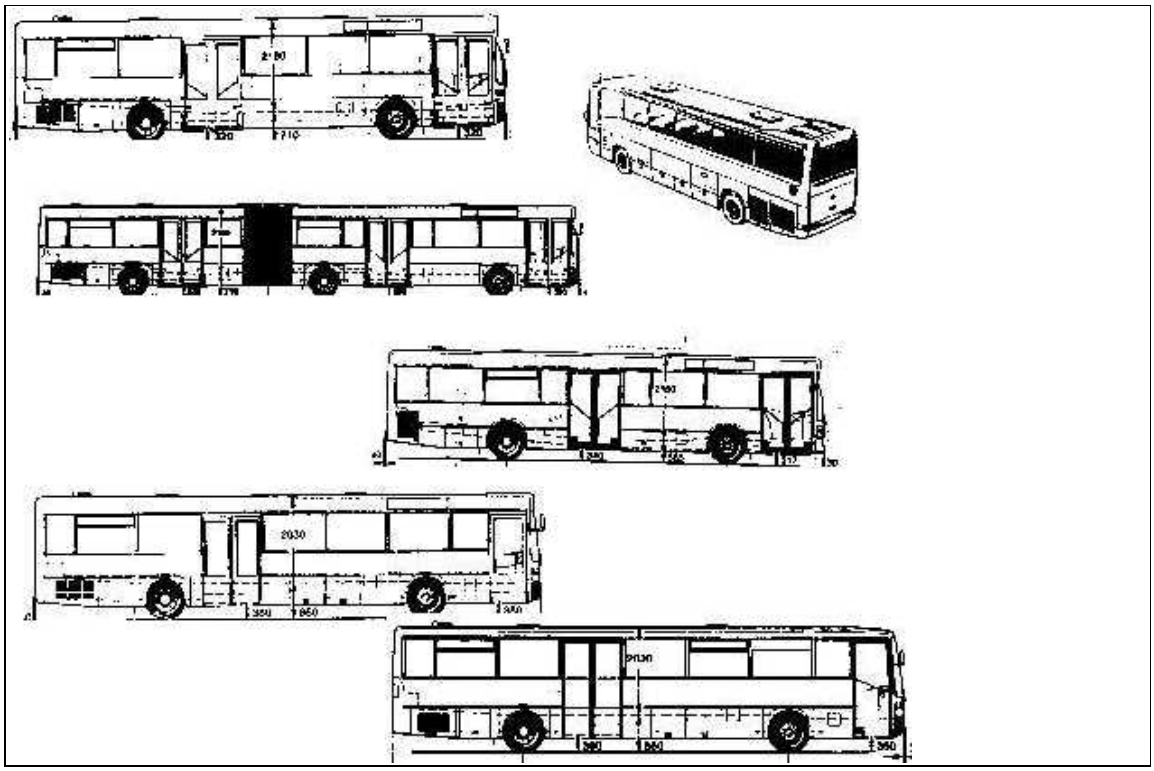


Fig. 5.2 represents different categories of coaches

Here and in automobile sector in general, it is the information regarding the *vehicle* is most general. Furthermore, the information regarding their respective *accessories* is important. In order to keep the model general, both these concepts are introduced as sub concepts of the concept *product*. Further we want to make a statement about the delivery date of these accessories. For that we have introduced a concept of *date* with the following three roles day, month and year. As it is not possible to stipulate sufficient attributes for such concepts, thus they are introduced as primitive concepts.

```
date            :<  anything.
manufacturer    :<  anything.
Product         :<  anything.

accessory       :<  product.
vehicle         : <  product.
installation    :<  product.

has_accessory   :<  domain(product)  and range(accessory).
accessory_of    := inv(has_accessory).

manufactured_by   :< domain(product) and range(manufacturer)  type feature
manufacturer_of  := inv(manufactured_by).

has_part  :<  domain(installation)  and range(accessory).
part_of     :=  inv(has_part).
```

```
day   :<  domain(date)   and range(1.....31)       type feature.
month :< domain(date)   and   range(1....12)     type feature.
year  :<  domain(date)  and range (1....2100)  type feature.
```

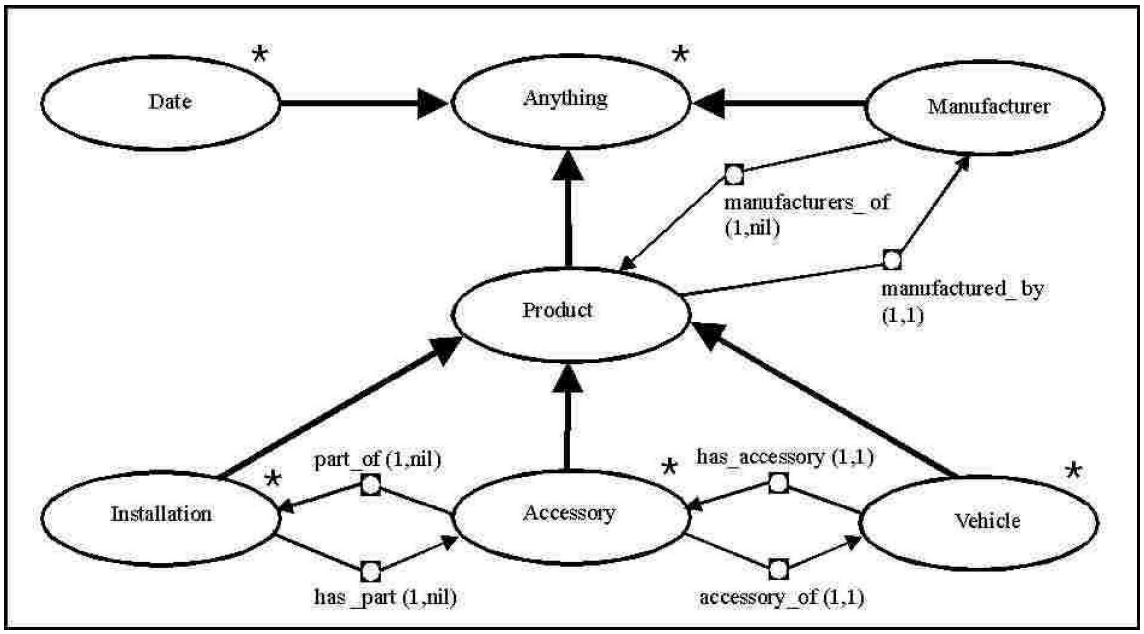


Fig. 5.3: basic concepts for coach sales

Through this small example one can already explain certain problems in detail. From the basic concepts defined till now, it is possible, without an error being reported by the representation system, to create an object which is an instance of both the concepts *vehicle* and *date*. As a result, we need to explicitly represent that these concepts are disjoint. For this BACK offers two possibilities. The first possibility is to fix it with the help of **not-**operator at the time of introduction of a new concept. The model thus could be changed as follows:

```
manufacturer  :<  anything
    and not(date).

product  :<  anything
  and not(date)
  and not(manufacturer).
```

However, this type of representation has two drawbacks. One that not-operator in BACK can only be used for primitive concepts. The other is that by large number of disjoint concepts this type of representation becomes very complex (i.e. the description of the concepts would be too long due to ever increasing number of **not**-clauses).

The second possibility, to achieve the disjointedness among concepts is to define these concepts unequivocally. This can be achieved by introducing the features. The concerned concepts must be assigned to different rolefillers. As already mentioned, a feature is a role whose maximum number of rolefillers is limited to one.

% type serves only for differentiation of concepts

```
type  :<  domain(anything)   and range(string) type  feature
date   :< anything   and  type  :  'date'.
manufacturer   :< anything and type art  : 'manufacturer'.
product  :< anything and type  art : 'product'.
```

Now, if some one would try to create an object, which is an instance of both the concepts *date* as well as of *manufacturer*, then the system would detect an inconsistency. This can be read in the natural language interpretation as follows "an object is a date and manufacturer and has maximum one type (as type is a feature) and has minimum two type(s), as date it has a type 'date' and as manufacturer it has a type 'manufacturer'." The benefit of this approach lies in that the descriptions of the concept do not get inflated. However, an extra role is necessary which explicitly serves to make the concepts disjoint as pairs. Should later further concepts are to be differentiated on a lower level, then extra features must be introduced. Both concepts *vehicle* and *accessory* can be seen as the examples of this. Here, both concepts are descendent of the concept product and thus would inherit the type 'product'. With this, right from the beginning they are not disjoint and one could not assign a different rolefiller for this role.

For the purpose of clarity, while modeling the large number of different concepts, we decided for the second method. For all other cases we used the **not**-operator. However, none of the above discussed methods provide an adequate solution as far as the consistency of the knowledge base is concerned, as the disjointedness of concepts play an important role in creating a consistent solution. A different approach, which solves this problem, has been chosen in "constructive problem solving method (CPS)" and is described in [Klein 94].

Now, suppose we want to extend the present model within the considered coach world to include all the different coach-models, which differentiate from each other on following three attributes: *manufacturing_height*, *total_length* and *comfort_level.*

To represent the possible instances of an attribute , atomic values construct is used. In BACK it is known as attribute. A set of such attribute-values build an attribute domain. This could either be closed or open i.e. all the atomic values which are supposed to belong to this set should be explicitly mentioned, or it could be open, if it is not possible to give all the possible values at the time of declaration. For such a purpose the construct of attribute_domain is provided for. Moreover one can build union, intersection and difference of sets from already existing attribute_domains and which could be used for declaration of new sets.

```
coach :<  vehicle
o404  :<  coach

code_manufacturing_height     :=  attribute_domain([rh, rhd, shd]).
code_number_of_standard_rows  :=  attribute_domain([10, 13, 15]).
code_comfort_level            :=   attribute_domain([k, l, s]).

has_ manufacturing_height:< domain(o404)
       and range(code_manufacturing_height) type  feature.

has_manufacturing_length  :<  domain(o404)
```

```
        and range(code_number_of_standard_rows) type  feature.
has_comfort_level  :< domain(o404)
     and range(code_comfort_level)  type feature.
```

Now the individual coach models result in from the combinations of instances of each attribute. The properties are described here as the closed set of attributes, as they are fixed right from the beginning and they can not change. Now, a specific coach model can be introduced as a defined concept using these properties, as they represent the sufficient characteristics for the classification of an object to these concepts. First, some more concepts are introduced which have only one instance of a property and with that one can represent a whole lot of coach-models. Later, one can easily formulate the manufacturing constraints and rules to fix prices e.g. lots of rules are valid either for all "10" series or for all "L" -coaches (coaches with comfort level = L).

Example:

```
o404_10  :=  o404  and  has_number_of_standard_rows  :  10.
o404_l    :=  o404  and   has_comfort_level                :  l.
o404_rh  :=  o404  and   has_manufacturing_height       :  rh.

% a specific coach model from 10 RH-L series would be:

o404_10rhl  := o404_10   and o404_rh  and  o404_l.
```

The above mentioned definition would have been sufficient if all the combinations of the properties could represent all the possible coach models. But this is not the case. For example a coach 10 RH-S is not at all manufactured. Even if we don't introduce this concept but it would have been possible to assign an object to this combination. Because of this reason such combinations are explicitly represented as inconsistent states. As each concept which is subsumed by a concept *nothing* represents an inconsistent state and can be expressed with the help of an I-Link.

```
o404_10  and o404_rh and o404_s  => nothing.
```

Now, every attempt to create an object which contains instances of the above mentioned three properties is going to result into an error message from the representation system. Apart from that, if we use this inadmissible combination (e.g. while formulating the rules), then the representation system will indicate inconsistency on introduction of new concept terms. Eventually, we can now represent the individual production patterns in BACK. They are essentially characterized by the fact that they have a specific motor fitted in:

```
o404_10rhk_01  :=  o404_10rhk
    and all(has_accessory   and    range(motor),  om401).
```

There are a whole lot of characteristics which every coach of a specific type shows and they are e.g. total length, trunk volume, allowed total weight, and turning circle. These characteristics, however do not pose a sufficient conditions for the classification and therefore formulated as I-Links:

```
total_length    :< domain(vehicle)  and range(number)   type   feature.
trunk_volume  :< domain(vehicle)  and range(number)   type   feature.
turning_circle  :< domain(vehicle)  and   range(number)  type  feature.

o404_10  =>  total_length  :  9220
   and turning_circle :  15888.
```

% The trunk volume of the coaches of o404-series with 10 standard rows and comfort level K or L is same (equal).

```
o404_10  and  all(has_comfort_level, aset([k, l]), code_comfort_level))
  =>
  trunk_volume  :  5.06.

o404_10  and  o404_s  =>   4.55.
```

The aset-construct in the rule creates a concept term which contains the subset (namely k and l) of all the possible comfort levels.

This type of description shows following benefits: later, if one creates an object and assign the value 10 to the role *number_of_standard_rows* then the system infers through the domain of the role that the newly created object is a coach from the series O404. Further, this concept would be classified under the concept o404_10 because it fulfills the posed conditions and eventually through the rules other values are assigned to the object.

This facilitates a new form of query of information. One creates an object with known characteristics and the system classifies it and applies the given rules. This can cause a new classification...and so on. Eventually, one can see the resulted description of the object that means showing the concrete values of specific roles.

In the following section, the accessories are going to be modeled. For every type of accessory a concept and necessary roles have been introduced.

Example:

```
physical_location := atrribut_domain     % an open set of atomic values
location              :< domain(accessory)  and   range(physical_location).

% the transmission relation as string and as numerical value

has_transmission :<  domain(back_axis) and range(string)   type  feature.
has_ivalue          :<  domain(back_axis) and range(number)   type feature.

axis   :<    accessory
  and   exactly(1, location)
  and all(location, aset([front, middle, back], physical_location)).

front_axis  :=  axis  and  location  :   front.
middle_axis  :=   axis  and  location  :  middle.
back_axis  := axis and location  :  back.
```

```
back_axis44_12  := back_axis
  and has_transnission   :  '44:12'.

has_transmission  :  '44:12'  =>  has_ivalue  :  3.6666.
```

At this point it is clear that the it is necessary to involve those components of a coach in the modeling which come in a coach as a standard fitting and cannot be changed by the customer. This, in above example, is the case with front axis, If we don't consider this in our modeling then it is possible at some later stage the domain model will be not understandable.

The front axis in O404-series always belongs to the standard fittings and there is no possibility of a variation for the customers. But for back axis it is totally different. Now, if we describe the components of the vehicle which can be changed, then we'll have to formulate certain manufacturing constraints which states that a coach of this series has to have only one axis.

One of the essential point of the modeling of vehicle components is that to organize their structure intelligible respectively prominent. In a delivery external tie-up document there are lots of accessories which have their own individual code but only when put together they characterize the seating arrangement of a coach. The reason to divide them in individual accessories/codes is the easy formulation of manufacturing constraints and to allow the manual calculations of the prices. Fig.5.4 shows the net representation of this piece of knowledge. This representation also shows that the concept of accessory has not been fixed equivocally. In the figure, seating_arrangement shows the number of seats and order of the seats while different parts/components of individual seats are represented separately.

In a computer supported advisory system, it is appropriate to preserve the description of the seating arrangement as a whole, so that all the related information can be concentrated. This has been shown in Fig. 5.5.

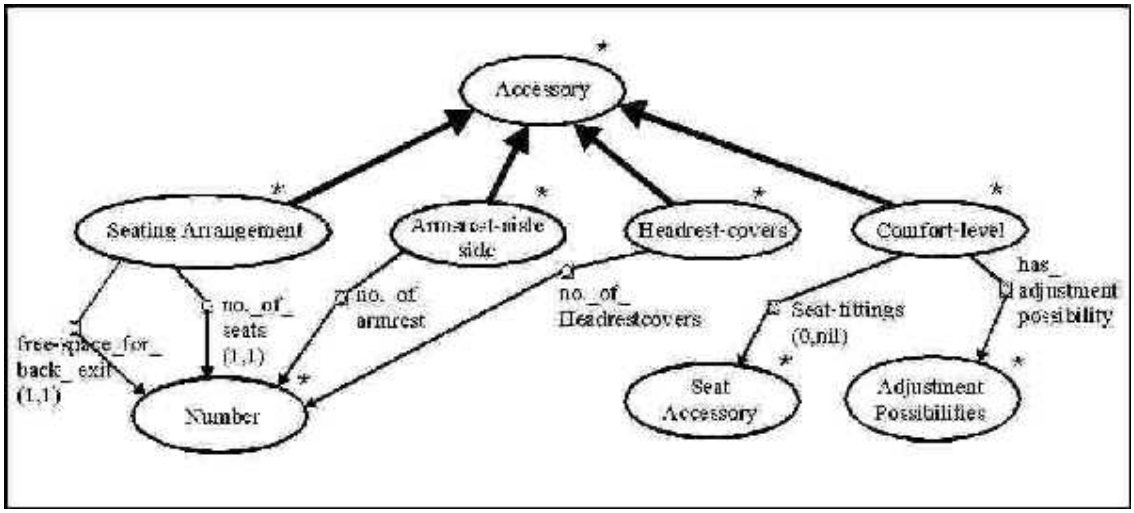


Fig. 5.4: The division of seating_arrangement in different accessories make the representation complex

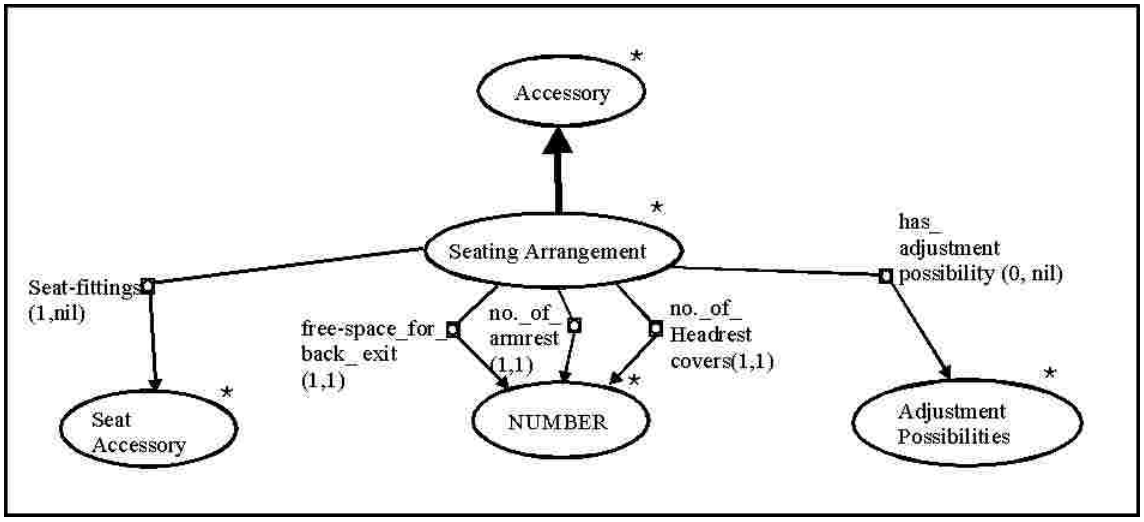


Fig. 5.5: The above representation at least contains all the characteristic attributes

Fig 5.5 shows the modeling chosen by us. Here, all components which were previously represented as separate accessories become the attributes of the seating arrangement. Here also some compromises were made, thus still better representation would have been where all the seats in their last consequence and also the accessories available for them are represented as individual concepts (fig. 5.6) . Thus, for example, the arm rests would only be available for seats in the aisle section and the safety belts at the speed of 100km/h would then be required for seats behind the two exits and the middle seat of the last row. With this one would have made possible to describe each and every seat individually. For details see [Seils 94].

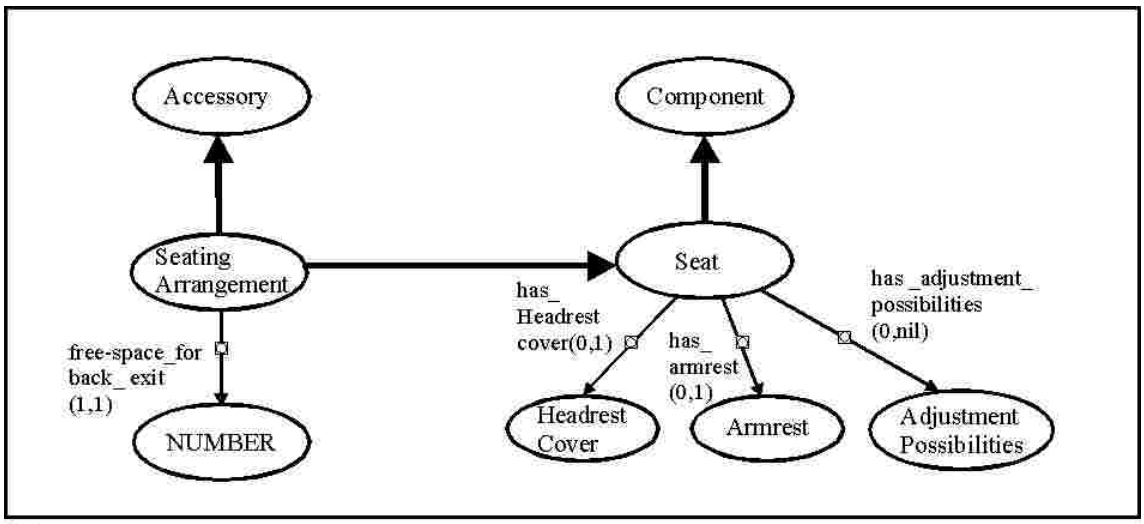


Fig 5.6

Fig. 5.6: Seats and their respective accessories represented as concepts

5.2.2 Need of defaults as preferences:

There are 35 basic models of coaches available within the considered coach series. During the consultation, the sales person has to select an appropriate basic model according to customer wishes. This basic model actually fulfills all the requirements for consistency and

completeness but further adapted to the special needs of the customer by adding or deleting the accessories.

Extension or changes can now be made to the vehicle. In this context, the question arises that how these sub configurations can be represented. Here default values play an important role. A default is a fact or a rule which is valid as long as there is no other assumption which leads to an inconsistency.

However, following differences are to be considered. Normally, defaults are used for inferring from the incomplete knowledge but here this is not the case. We use them here more or less to represent a kind of preferences (factory fitted accessories as standard fittings).

In our domain there are three different types of situations where one would use defaults:

- Suppose when we want to represent „for a given coach which accessories are available as standard fittings". There, e.g. A coach of series O404 10 RH-L in standard version would come with 35 passenger seats. However, a customer can select a different seating arrangement according to which there are 39 seats arranged in a different way. This works differently for the factory fitted accessories which cannot be modified. These can thus be represented directly as manufacturing constraints.

- If an individual accessory can further be configured as e.g. in the case of seating arrangement an initial description is useful. For the seating-arrangement it could mean that a specific comfort level is included in the prices as a standard fitting and other levels of comfort can be chosen by the customer.

- For a whole lot of accessories there are different prices documented in the sales catalogue. The quoted price results in from the actual configuration of the vehicle. With the help of defaults one can appropriately represent the basic price of an accessory and in case the actual configuration requires some other price then it can be overwritten by some new value.

Due to the strict inheritance mechanism the introduction of exceptions in terminological systems is problematic. At least, they can't be represented directly in the concept description. In [Brachman & Schmolze 85], this problem has been described as follows:

"The lack of cancellation of Value Restrictions might appear problematic from point of view of representing "exceptions". However, if we were to allow cancellation of components within Concepts, then these components would be reduced in status from necessary conditions to default assertions. We feel that such nonnecessary conditions are more appropriately expressed outside of the taxonomy."

[Brachman & Schmolze 85]

Theoretical approaches in this direction are following the goal to represent exceptions - and through it the defaults - in terminological systems as rules which can be prioritized [see Royer & Sauerl '91, Baader & Hollunder '92 & Quantz & Royer '92] . The defaults can then be represented through the I-Links, however with a difference that the conclusion of a rule can be overwritten.

5.2.3 Representation of manufacturing constraints:

In this section we'll try to give the reader a picture about how all types of manufacturing constraints can be translated into BACK constructs. Fundamentally, it is possible either including them directly in the description of each vehicle (naturally as „high" as possible in the hierarchy) or formulating them as implication links.
In following examples A stands for the concept terms for accessories and V stands for the concept terms for vehicles. As we'll limit ourselves to the relation between the accessories and the vehicles, we'll be using following roles:

has_accessory :< domain(vehicle) and range(accessory) .
accessory_of := inv(has_accessory) .

Semantic:
Accessory A_1 can only be selected in connection with the accessory A_2
Syntax:
some (has_accessory, A_1) => some (has_accessory, A_2) .

This first rule can completely be interpreted as follows: If an object is found which is an instance of a concept vehicle and has a constraint on the role has_accessory so that there is a minimum of one rolefiller which is an instance of accessory concept A_1 and from that one infers that similarly there must be minimum of one rolefiller for has_accessory role of the vehicle which is an instance of the accessory concept A_2. For triggering the rule it is not necessary that actually an instance of the concept A_1 exists and also from its triggering no instance of concept for A_2 will be created.

Semantic:
Accessory A_1 cannot be selected in connection with Accessory A_2
Syntax:
some(has_accessory, A_1) => no(has_accessory, A_2).
%**or**
some(has_accessory, A_1)
and (has_accessory, A_2) => **nothing**.

Now, an attempt to fit both the accessories will lead to an inconsistency due to the unification of the number restriction of the role has_accessory.

Semantic:

Only one of the accessories A_1, ... ,A_n can be selected.

This rule cannot be directly translated. The accessory in this list is always of same type and disjoint in pairs. It is therefore sensible to introduce a new general concept and through this induce a respective number restriction to the vehicle concept.

Syntax:
A_0 :< accessory .
A_1 :< A_0 .

A_2 :< A_0 .
.....
A_n :< A_0 .
vehicle :< anything and atmost(1, A_0).
vehicle => atmost(1, A_0).

This structural change leads to a better understanding of the rule. The following example should once again illustrate the general process:

```
signal_installation              :<   accessory.

hostess_bell                     :<   signal_installation.
passenger_signal_installation    :<   signal_installation.

o404                             :<   coach and
                                      atmost(1, has_accessory, signal_installation).
```

A yet another benefit, which is deduced from here is the better maintenance of the knowledge base system. Now if a new signal installation is being added then the rule is neither extended or changed.

From the following example, however certain problems can be detected:

```
Vehicle                          :<   atleast(2, has_accessory, axis)
                                      and exactly(1, has_accessory, front_axis)
                                     and exactly(1, has_accessory, rear_axis).
```

% further, following is valid for the coaches of O404-series....

```
o404                             :< all(has_accessory   and   range(axis),
                                    all(location, oneof([front, rear])).
```

%...In contrast to other coach-series this coach-series do not have the middle-axis

Now, if we had only two different types of axis, say front and rear axis, in our knowledge base, then it would have not been sufficient to describe only that the coach of series O404 has maximum of two axis, because two instances of front_axis concept would also fulfill this condition.

One of the requirements for the advisory system was to get the specialization of such objects. Say, for example, there exist three different types of axis (front, middle, and rear), then it is to be achieved that any object which describes/is classified as an axis must exactly fall under these three types. This task, at the moment, cannot be solved by the present day terminological formalisms because they do not work under the „closed world assumption“.

Further, it is still not sufficient to describe that a vehicle has exactly one front and one rear axis. Even though the concepts front_axis and rear_axis are disjoint but BACK does not infer that a manufacturable vehicle has to have two axis, this is due to an incompleteness of BACK. This may seem to be harmless but however in our domain has lot of consequences on the total manufactureability checks. This is due to the „open world assumption“ that the system still

does not infer that there could be two axis, even though there are no other sub-concepts for axis available.

Semantic:
Accessory A_1 comprises (includes) Accessory A_2.
This piece of knowledge can not be represented directly in a rule and needs a structural change. One could represent e.g. A_2 as a part of A_1. This relation cannot be compared with the relation has_accessory as A_2 is now a direct part of an accessory and not of vehicle. Therefore we used another role, called has_direct_part.

Syntax:
A_1 :< accessory and exactly (1, has_direct_part, A_2).
% **or**
A_1 => exactly (1, has_direct_part, A_2).

During the consultation process, however, one could choose/select at first only that accessory which is already included (in our case A_2) and later that accessory which includes the other accessory. In this case an object created for A_2 must automatically be used as the rolefiller of the role has_direct_part. This is to make sure that after the selection of A_1, there exists no two instances of A_2 .

Now, if an accessory contains/includes another, then during the price calculations one has to be careful about that the price of the included accessory can influence, not both the accessories are considered in the calculation. In this type of modeling, while calculating the prices for accessories only those accessories are considered which are not the part of an another accessory.

Semantic:
replace accessory A_1 with accessory A_2 if an accessory A_x already exists
Syntax:
This rule cannot be represented directly in BACK, as it accounts more of a control-knowledge and changes the actual data space.

To replace an accessory of a vehicle on the basis of specific reasons cannot directly be represented in the terminological systems. The replacement requires overwriting previously fixed facts and thus corresponds to the problem of representing defaults in such formalisms.

One another problem arises from that, not all the accessories are available for all the coaches. The reason for that could be of strategic or manufacturing nature. Fundamentally there are two cases which could be differentiated and could be represented either via the value-restriction of the role has_accessory or through its inverse role.

Semantic:
Accessory A is only available for coach model B
Syntax:
A => no(accessory_of, B).
% **or**
some(has_accessory, A) => B.

Semantic:

Accessory A is not available for coach model B

Syntax:

A => no(accessory_of, B).

% **or**

B and some(has_accessory, A) => nothing.

Both the above mentioned possibilities handle the easiest cases. In general, the availability of the accessories has to be fixed for a series of coaches. This, though possible but is tedious, because the terminological systems normally do not support the disjointness of concepts. A way out in this situation would be to handle all the cases individually, however, this would very soon lead to a large number of rules and that would increase the maintenance effort. An alternative to that would be to describe them with the help of certain characteristic attributes as lists of coaches for which that accessory is available .

Example:

```
%  This special wind screen is available only for models of  K-and L-
%  coaches
windscreen_transparent_compound-glass_one-piece
     =>
     all(accessory_of,
       o404 and all(has_comfortlevel, aset([k,l], modelvariant))).
```

In above example a problem can arise in case of an exception. For example, multi-*channel_place-module-installation* can be delivered for all coaches of series O404 with exception of O404-K-coaches and of O404-10RH-L. Now the set of coaches for which the accessory is available cannot be built without making use of disjointness property.

```
% this type of description is not allowed in terminological systems !!!!!

multi-channel_place-module-installation
                                  => all(accessory_of,
                                     o404_s   or
                                     o404_rhd  or
                                     (
                                       o404_l  and  all(has_basic_number_of_rows,
                                        aset([13,15], basic_number_of_rows))
                                      )
                                                  ).
```

It has been only possible in this case to formulate it as a negative statement and by using the **and** operator:

```
multi_channel_place-module_installation
     =>
     no(accessory_of, o404_k)
     and no(accessory_of, o404_10RHL).
```

Here, it is possible to get fairly understandable representation even specific cases where an accessory, for purely technical reasons, cannot be fitted into a given coach model.

5.2.4 Rules to fix the price

Within this application domain the continuous actuality of the price calculation plays an important role. As we have already mentioned that the price of an individual accessory can depend upon the final configuration i.e. to calculate the price some rules are required. In the following section various price rules are represented and their representation in BACK is also given. In certain cases these rules are combined to form a complex rule.

Semantic:
an accessory A has a unique price for all the coach models.
Syntax:
A => has_price : 7860

Semantic:
An Accessory A has different price for different coach models (C_1 & C_2)
Syntax:
A and all(accessory_of, C_1) => has_price : 1500.
A and all(accessory_of, C_2) => has_price : 4200.

Semantic:
The price of an accessory A_1 is dependent upon the fact that accessories A_2 and A_3 are also fitted in the same coach.
Syntax:
A_1 and some(accessory_of, C
and some(has_accessory, A_2)) => has_price : 25550.
A_1 and some(accessory_of, C
and some(has_accessory, A_3)) => has_price : 15000.

Semantic:
An accessory gets its price due to the number of components it contains and their prices.
Syntax:
for this kind of calculations most of the representation systems provide some external functions (s. section 5.2.5). In BACK its known as **test** and **compute.**

At the moment, we have only one possibility to calculate the price in the last case and that is to do it in the advisory system. However, it posses a problem i.e. how to find such accessories for that this type of calculations is necessary. Here, we have to deal with such question as „which accessories get their price due to the sum of their components ?“ As this type of information can't be included in the representation system, thus it has to be placed outside the representation system.

Another problem lies in the fact that the price of a basic coach model contains the prices for all the factory fitted accessories(standard fittings) and the individual prices of the accessories can't be explained/executed anymore. The price for the accessories which replace parts of factory fitted accessories(standard fittings) is then fixed relative to that standard accessory.
This leads to introducing some „artificial“ accessories. For example, „all coaches of series O404 with comfort level L or S contain ASR (Anti-Slip-Regulator) as standard fitting. But a customer can decide otherwise. This fact is represented in the list of standard accessories as „ommited_ASR“ and has a negative price. This can be modeled in BACK as follows:

```
asr        :<   accessory.
omitted_asr  :<   accessory.

asr and all(accessory_of,
     all(has_comfort_level, aset([l, s]), code_comfort_level)))
     => has_price   :  0

 omitted_asr and all(accessory_of,
     all(has_comfort_level, aset([l, s]), code_comfort_level)))
     => has_price   :  -2000.

some (has_accessory asr)  =>  no(has_accessory, omitted_asr).
```

The last rule says only that both the accessories can't be fitted simultaneously. This does not cause the automatic de-selection of other if one is selected. This type of modeling makes the representation complex and difficult to understand. Now, the omission of an accessory during the interaction with the system should be indicated by the de-selection of the **same** accessory *asr* and the price calculation has to be adjusted accordingly. A solution to this problem would be to assign each and every accessory a price and apart from that whether this accessory is included in the basic price of the vehicle (like all the factory fitted accessories) or not (like all the custom accessories).

The price due to the „deep" structuring in this present modeling can not only change due to other accessories or vehicles but can also be modified due to the change in its own properties. In case of seating arrangement it can happen due to the addition of certain accessory parts. This leads us to divide the price of an accessory into parts, for example, basic price and one or more additional prices (it can sometime be negative as well) or an end price.

```
has_basic_price    :< domain(product)   and    range(number)   type
feature.
has_added_price    :< domain(product)  and    range(number).
has_end_price      :<   domain(product)  and   range(number)    type
feature.
```

The missing capability to overwrite the already existing values (rolefillers) leads to that the intersection set of conditional parts of the rules should always be an empty set.

Example:
A second dynamo costs normally DM 4000. However, it costs nothing if there is an air-conditioner fitted at the rear of the coach, because its price is then included in the price of that air-conditioner. Following two rules represent this :

```
dynmo  and some(accessory_of, B
               and  some(has_accessory, rear_airconditioner))
     =>
     has_price   :    4000.
rear_airconditioner  =>   has_price    :    30900.
```

Theoretically, this problem can probably be solved by following representation:

```
rear_airconditioner :<  accessory
    and  rvm(included_in_price,
        accessory_of
        comp[1] has_accessory and range(second_dynamo)).

rear_airconditioner    =>    has_price   :    30900.
second_dynamo          =>    has_price   :    4000.
```

Now, if a rear air-conditioner is ordered as an accessory for a vehicle then it would automatically include, if need be, the available second dynamo. Thus, a general rule of following type could be integrated in the advisory system that all accessories which are the rolefillers of the role included_in_price do not have any effect on the end price of the vehicle.

Independent of above problems in our domain there are different types of prices which on one side influence the interactions with the users and on the other side the price calculation of end price:

List Price: represents the price of an accessory in normal case without considering the VAT. This can be directly considered for calculating the end price.

Price on Request: means that the actual price of the accessory is not fixed right from the beginning and it has to be inquired through a request to the production facility. The advisory systems has to ask the price from the sales person and then consider it for further calculations.

Net Price: means, that the accessories are being produced by some other manufacturer thus are not to be considered for any kind of discounts.

Standard Price: means that the price of an accessory can deviate from the given price, however it lies within this range. The end price can be fixed only at the confirmation of the contract. In such cases the advisory system indicates the eventual price deviations.

Therefore, it is sensible to introduce an extra role for the different types of the price and to fix that all the accessories have to belong to either of these categories.

```
price_catagory  :<  domain(product)  and range(
          oneof ([list_price, price_on_request, net_price, standard_price]))
          type  feature.

accessory  =>  exactly(1, price_catagory) .
```

Here, also the possibility of representing list price as defaults to each accessory would be of great use.

---

[1] Note:- The comp-operator allows the concatenation of two roles. Thus the natural language interpretation of this expression would be: “The rear air-conditioner includes in its price the second dynamo for only those coaches where the rear air-conditioner is fitted”.

5.2.5 External Functions:

In the previous section we described how the prices of individual accessories can be represented in BACK and what problems arise from that. However, there are still two more problems which are open. The first is to calculate the actual total price of a vehicle and second the calculation of prices of those accessories which are further structured e.g. the seating arrangement.

In BACK++, to check and calculate the rolefillers, the two functions **test** and **compute** are available. However, to these functions underlie a series of restrictions. The called procedures should not change the knowledge base. Apart from that the **test** functions should be monotonous, i.e. if a test for an object is successful then the result should not change the knowledge base by adding some more information. The **compute**-function calculates one or the set of role fillers for a given role. However, this should return the solution only then when all the required information for calculation are already present and the inducing of further information to the knowledge base does not lead to a change in another set of rolefillers.

In order to show the usage of compute for our purposes and to make this problem clear, its application is shown in a simple case. We could, for example with the help of this function, compute the number of necessary armrests on the aisle side for a selected seating arrangement, without considering them directly into the concept definition.

The compute-construct in BACK++ accepts the input of a role and an external computation function which has to be declared beforehand. In our domain we'll do the calculations only then, if the customer has really selected the armrests as an option. To do this we introduce the following rule:

```
% declaration of a compute function

    backdeclare(compute_fct (calculate_number_of_armrest)).

    seating_arrangement  and  seat_components  :  armrest_aisle
        => compute (number_of_armsrest, calculate_number_of_armrest).
```

This rule says that the necessary number of arms rest can be calculated if an object is found which is an instance of concept *seating_arrangement* and shows *armrest_aisle* as a rolefiller for its role *seat_components* . (the armrest could then be selected either directly by the customer or indirectly through the comfort level).

The corresponding prolog predicate, which calculates the prices, can be formulated as follows:

```
    calculate_number_of_armrest (seating_arrangement, [number_of_armrest])
    :-
        backretrieve (X  =  [rf (number_of_rows) ] for seating_arrangement),
        X  =  [ [number_of_rows] ],
        number_of_armrest  is  number_of_rows  *  2  -  3.
```

The existing functionality is sufficient to solve this simple problem. But the calculation of prices in our domain poses higher requirements. The price calculations should take place incrementally in order to give the sales person the overview of actual total price. Now let us

imagine that a sales person configures a vehicle step by step during the consultation with the customer and therefore all the corresponding functions are also to be activated as some accessory is being added or the one which changes the attributes relevant for price calculations. However, this approach to continuously compute the prices can lead to a performance problem.
Apart from performance problem there are many other reasons to not to do the calculations continuously. During the consultation session the advisory system could check which configuration steps are allowed (because they would fulfill the requirements for completeness and correctness) as next in order to present them to the customer as a choice. Within this process it is not necessary to do the price calculations unless we want to suggest a cheaper solution.

In this situation it would be thinkable to have a calculation function which can be triggered only then, when it is explicitly required by the application. Possibly, one could allow different classes of rules (user defined), which could be switched on and of by the user in addition to their actual firing conditions. Whether this approach is agreeable to the fundamentals of terminological logics however could not be assessed here.

A theoretical approach [Kor 92] describes that how these external functions can be included in the BACK formalism. This approach includes the management of dependencies which occur during the price calculations and are utterly necessary for our domain. Another difference lies in that the parameters for the external functions can be given at the time of their call. This has an advantage that for their implementation there is no need to have the knowledge about the construction of the knowledge base (i.e. the internal structure).

5.2.6 Availability of Accessories

partially some accessories or their variants are marked as not available in the accessory lists. The reason for that could be many fold. It is possible that the accessory is not available at the moment because of some production problems or due to some delivery problems. On the other side in this situation it deals with absolutely new accessories or variants However, under certain circumstances both of these accessories offer an advantage over the offers which cannot be fulfilled by the present product spectrum. With their inclusion in the accessory list a sales person has the opportunity to offer such accessories which are not yet available at the time of signing of the contract.

Here, one could choose from two different ways. The first alternative would be to avoid the selection of the accessory by the intervention of the sales person or by not showing the corresponding accessory at all. The second possibility would be to hint the eventual delay in delivery. With the later possibility the sales person gets the opportunity to leave the decision to the customer whether the customer would like to accept the delay in the deliver or he/she would not like to have this accessory at all or he/she selects an alternative for that. Where as the first representation becomes the manufacturability constraint, in the second alternative one deals with the meta-information which effects the control of the advisory system.

For marking the accessories (deliverable or not_deliverable), we can then introduce one corresponding feature:

```
availability  : oneof([deliverable, not_deliverable]).
has_status  :<  domain(accessory)  and range (availability)  type  feature.
```

There lies an another possibility also where one introduces every time a corresponding concept in case this term appears quite often.

```
availability  :=  accessory and status  :  deliverable.
not_deliverable  := accessory  and  status  :  not_deliverable.
```

We could now, assign a status to an accessory with the help of a rule. In the following example we want to express that a specific cover_material (Trim) is at present not available and therefore no accessory can be delivered which is covered with this material. Nevertheless, one has to be sure that all the accessories which utilize this material have the same role for that or the one which is inferred by it (i.e. the corresponding special one).

```
material :<  product.

cover_material   :=  material
                    and  oneof( [pinfleece,
                         syntheticvelour,
                          knittedvelour,
                          wollenplush,
                          fabric,
                          transparency,
                          leather] ).

covered_with   :< domain(accessory)   and  range(cover_material)  type  feature.

covered_with   :  pinfleece    =>   not_available.
```

Now, if we prefer the first alternative then we'll have to declare all those configurations as inconsistent in which this cover material is used and further on following manufacturing constraint has to be defined:

```
some(has_accessory, not_available)  =>  nothing.
```

The literal interpretation of the above rule could be „every vehicle, which has got an accessory whose status is not_deliverable, represents an inconsistent state. Naturally, once again at this point we need a facility to represent that in general (per default) all the accessories are available if not stated otherwise. In these cases marking of the accessories is sensible than to remove these accessories completely from the knowledge-base. The later approach means a higher maintenance effort due to the continuous addition and deleting of concepts which corresponds to them.

5.2.7 Additional Information

In the previous sections we have represented the crucial information regarding our domain namely the hierarchy of vehicles and accessories and the basic rule variants. What we have not discussed till now is that how the requirements/wishes of the customer can restrict the space of possible vehicles and their accessories and how this could be represented. The first and easiest of all possibilities is to leave it to the sales person to find out the right basic model and the necessary accessories on his/her own. For this case we could show at least some information regarding the usage, place of operation and some additional features of some

specific accessories (e.g. environmentally friendly or easy to maintain) on request. This can be achieved through the introduction of a new role, as shown in the following example:

```
usage :<  domain(accessory)  range(string).

heating_equipment :  (”facilitates cold start till – 40 0C”
                and ”improves the kaltaufpause”
                and ”saves ignition and battery”
                and ”reduces emission” ).
```

This can then be retrieved as follows:

```
backretrieve (X = [ rf (usage) ] for heating_equipment).
```

This approach does not allows us to represent the information which is implicitly there in the usage texts, however we can bind the general descriptions to general concepts and specialized description to special concepts.

In an another step one could offer a function which compares the two accessories. This comparison can be used on one side to compare two accessories from own product spectrum or for the accessories from the competitors, as far as the necessary information are available (the sales person can then select either of the two). Here, one can use the function provided by BACK to compare two concepts but it has to be extended for necessary filter functionality.

The advisory system can now, with the help of small number of queries, try to achieve a bigger restriction of the search space (possible vehicle_production_pattern) and to evaluate the resulting solutions according to the requirements already fixed. Through the requirements of the customers one can not only infer the required vehicle but one can also infer the individual accessories as well. The following example shows how this can be modeled in BACK:

Here we take the customer into account:

```
customer :<  anything.
          and exactly(1, customer_number)
          and exactly(1, name)
          and exactly(1, address)
          and some(offers).

offers :< domain(customer) and range(offer).
offer_date :< domain(offer) and range(date) type feature.

places_of_operation := oneof([city_traffic, regional_traffic,
                              long_distance_traffic]).
place_of_operation :< domain(requirements) and
range(places_of_operation).
potential_vehicle :< domain(offer) and range(vehicle).
```

```
offer  :<  anything
  and exactly(1, offer_date)
  and exactly(1, potential_vehicle)
  and some (place_of_operation).

place_of_operation  :  long_distance_traffic
    =>
    all (potential_vehicle,
      all(has_accessory  and  range(seating_arrangement),
          (long_distance_seating_arrangement)).
```

The above mentioned rule expresses the following: ”if the customer has stated that he/she would use the coach for long distance then the set of possible coaches is going to be all the coaches whose seating arrangement is suitable for long distances”. We could the define the *long_distance_seating_arrangement* as the seats which have got a side and back adjustment device.

5.3 Domain modeling in FLEX

The FLEX system can be seen as an extension of DL system BACK. The main differences are that FLEX supports full disjunction and negation, weighted defaults, situated object descriptions, term-valued features, and flexible strategies. On the other hand, FLEX does not support some of the functionality provided by BACK, e.g. revision. Further, in this section we will briefly sketch, how the characteristic of FLEX, which are not shared by all DL-systems, are to be used in our application domain.

Characteristics of FLEX: following are the characteristic of FLEX, which are not shared by all DL-systems:

an expressive term language comprising qualifying number restrictions, role inversion and composition, role value maps, disjunction and negation;

term-valued features, which behave similar to features having complex type values in Unification Grammars;

situated descriptions which allow the representation of alternate states of affair and form the basis for default reasoning;

weighted defaults as proposed in [Quantz 93] and formally investigated in [Quantz, Suska 94];

flexible inference strategies.

In the following we will briefly sketch these characteristics and the next section of this chapter will explain in detail how these characteristics of FLEX are to be used in our domain (automobile sales advisory). Here, we’ll be using the examples from the HTV(Heavy Transport Vehicle)/Trucks-World (Fig.5.7).

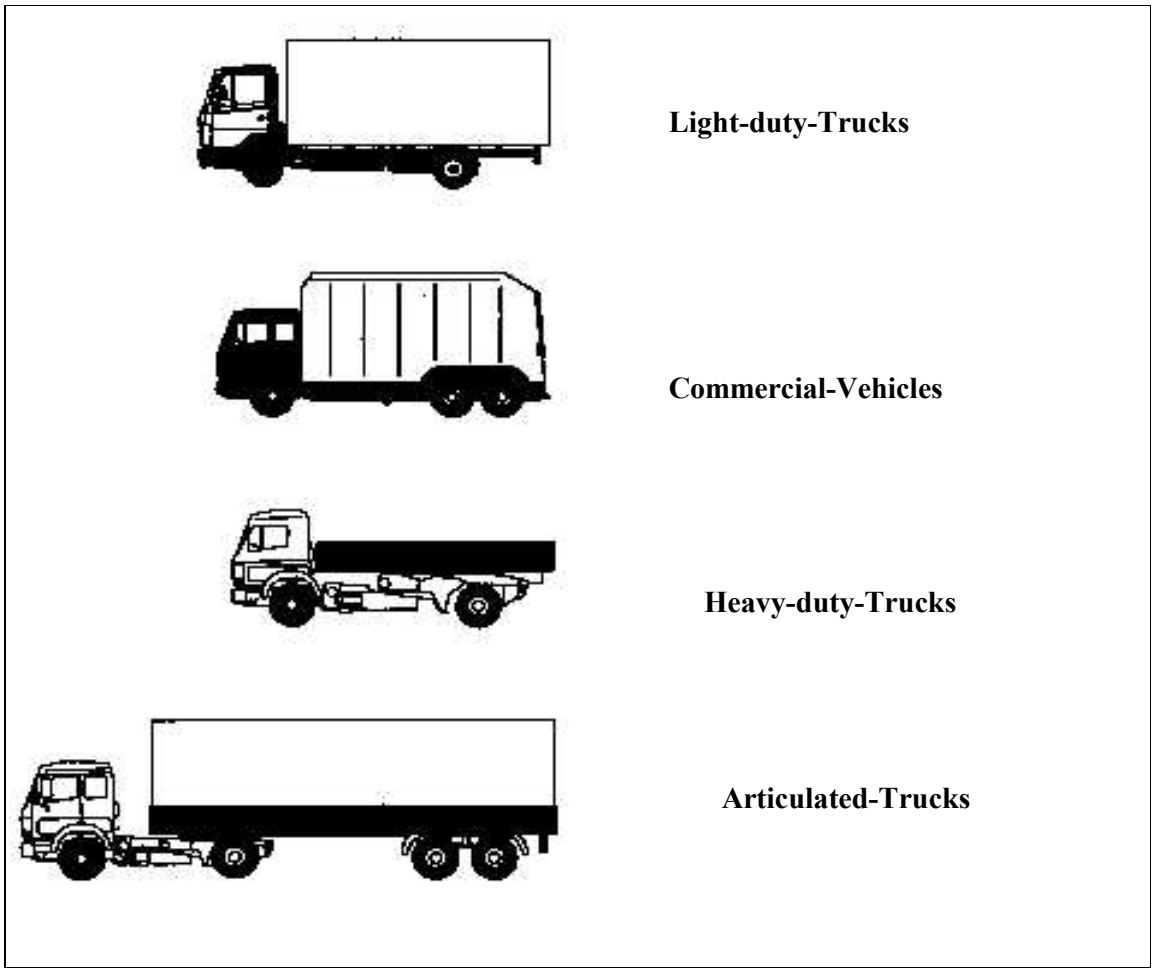


Fig. 5.7: The HTV/Truck World

5.3.1 Types and Objects:

One major difference between DL and the feature logics used in Unification Grammars [Carpenter 92] is the fact that the former distinguish between objects and types, whereas the latter only deal with types. The advantage of this distinction is that the object level of DL, which is missing in UG, allows a straightforward persistent storage of objects and their types. This provides further the basis for the integration of *epistemic operators* and *weighted defaults* into DL. However, one disadvantage of this distinction is that in standard DL only objects can be fillers of roles but certain features, e.g. 'has_motor' in our domain, are meant to take concepts as role fillers. A correct treatment of such a problem would be a higher order extension to DL, allowing relations whose arguments can be concepts or roles. Instead of providing such an extension, term-valued features were integrated into the FLEX system. This solves the problem at syntactic level. Further FLEX contains a special construct relating term-valued features to "normal" features as proposed in [Quantz '92]. This construct can be used to express the constraint that the filler of a role is an instance of the filler of a term-valued feature (see section 5.3.6)

5.3.2 Situated Descriptions:

can be used to represent alternative states of affair, to model backtracking and to perform reasoning by cases. The basic idea is to describe objects relative to a situation. Standard DLs support only a single ABox (i.e. set of object descriptions), but FLEX allows partitioning of the ABox into several situations. In a configuration application, where one incrementally describes an object, this allows one to jump back to an intermediate state simply by using the respective situation. This reflects in an example given in section 5.3.7.

5.3.3 Weighted Defaults:

have been integrated into FLEX system to support the modeling of rules which allow for exceptions. Syntactically, defaults are similar to strict rules, i.e. they relate two concepts, one being the premise, the other the conclusion. Further, weighted defaults allow to express orderings between multi-sets of defaults, i.e. weak defaults can accumulate their weights and override a stronger default. This is a rather useful property in many applications [Quantz, Schmitz 94; Schmitz, Quantz 95] and in our application of sales advisory as well. The example provided in section 5.3.8 shows how weighted defaults are used in our domain.

Before we go into the details of modeling in FLEX, here we would like to remind our readers about those parts of our domain where we could not express what we really wanted to express in/with BACK (see section 5.2.2 need of defaults as preferences). In general there are three different types of situations in our domain where one could use defaults:

- for a given vehicle which accessories are available as standard fittings
- if an individual accessory can further be configured
- different prices for the same accessory.

5.3.4 Flexible Inference Strategies:

FLEX offers a number of ways to control its inference behavior, which is based on sequent-style inference rules [Royer, Quantz 92 & 94]. The basic idea is a declarative representation of these inference rules, which allows for most rules to choose between:

- an application during normalization;
- an application during subsumption checking;
- no application at all.

Furthermore, the inference behavior can be controlled by switching off propagation rules for roles via the filler construct, and by setting FLEX states.

This possibility to tailor the inference capabilities of FLEX can be useful in our „Sales Advisory“ application in following ways:

- in different sales situation using different inference strategies
- switching inference strategies according to different levels of user (i.e. sales people with different level of experience).

5.3.5 Modeling in Flex:

As usual we once again start with the most general information which is to be represented in the knowledge base. Here the HTV/Truck world serves as the cross-section of the domain to be represented. In the following we give the sample representation from this domain and try to

explain where & how the above mentioned FLEX characteristics not only help in modeling the domain but in overcoming certain problems faced in previous section.

```
flexinit.
  product  :< ctop.
  accessory <>     vehicle.
  accessory :< product.
  factory_fitted_accessory :<  accessory.
  custom_accessory  :< accessory.
                vehicle   :< product.
  truck   :< vehicle.
  d2000    :< truck.
  d5000    :< truck.
  d10000  :< truck.
```

Above, the concepts are not used as atomic labels but are semantically related to each other and build the conceptual hierarchy. The initial command FLEXINIT is used to inform the system that this is the beginning of a model and subsequent tells are called *primitive concept introductions*. Further, the <> operator marks 'accessory' and 'vehicle' as disjoint concepts. In this model the system will reject a description in which a 'product' is both 'accessory' and 'vehicle'.(Fig.5.8)

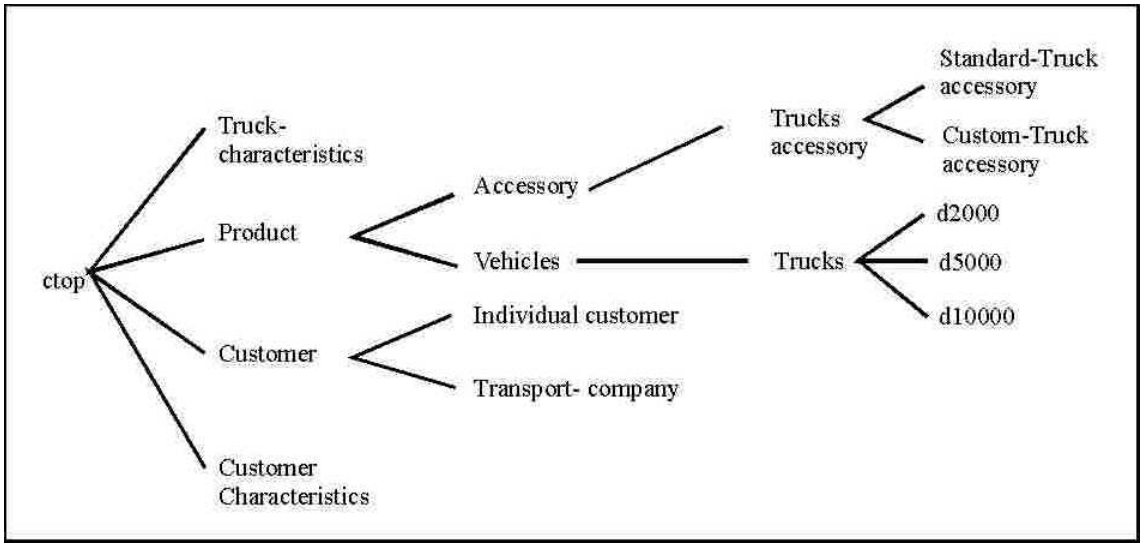


Fig. 5.8: First three levels of concept hierarchy representing our HTV/Truck-world

Similar to the coach world, we introduced as many as possible defined concepts, while modeling the truck world, as well. To illustrate the important distinction between primitive and defined terms consider the following example:

```
        comfort_accessory  :<  accessory.
    ff_comfort_accessory   :< comfort_accessory and factory_fitted_accessory.
custom_comfort_accessory := comfort_accessory and custom_accessory.
```

Here, 'ff_comfort_accessory' is introduced as a *primitive* concept, whereas 'custom_comfort_accessory' accessory is a *defined* concept *. Thus we know any 'ff_comfort_accessory' is a 'comfort_accessory' and 'factory_fitted_accessory', we know necessary conditions about 'ff_comfort_accessory'.

```
a_heater :: ff_comfort_accessory.
a_heater ?: comfort_accessory.
a_heater ?: factory_fitted_accessory.
```

For 'custom_comfort_accessory' we also know sufficient conditions, i.e. any object which is an instance of both 'comfort_accessory' and 'custom_accessory' will inferred to be an instance of 'custom_comfort_accessory'.

```
an_airconditioner :: comfort_accessory and custom_accessory.
an_airconditioner ?: custom_comfort_accessory.
```

Objects which are instances of both 'comfort_accessory' and factory_fitted_accessory, on the other hand, are not inferred to be instances of 'ff_comfort_accessory' [1]:

```
seat_belts :: comfort_accessory and factory_fitted_accessory.
seat_belts ?: ff_comfort_accessory.  % this query fails.
```

Now we turn the attention of the readers towards the definition and usage of roles in defining the concepts in our domain. Note that roles are used here as properties of concepts or objects and not as independently existing entities. We thus introduced following roles:

```
has_price :< domain(product).

has_accessory :< domain(vehicle) and range(accessory).
```

In general roles can be described by specifying the type of their first argument (DOMAIN) or their second argument (RANGE). Since many roles are functional, i.e. can have at most on filler per object, thus are represented as features, as shown in following example:

```
manufacturing_year :< domain(vehicle) and range(number) and feature.
```

Thus it is not possible to specify two different years of manufacturing for a particular vehicle.

5.3.6 Term-Valued Features

Let's consider in following example the type of a truck, e.g. roadtrarin_truck, long_distance_truck, ldt10000. Instead of treating these types as object, it is appropriate to treat them as concepts which make it possible to order them in a conceptual hierarchy:

```
truck_type :< ctop.
roadtrain_truck :< truck_type.
```

[1] Note:- Standard explanation of this distinction is that for the primitive terms only necessary conditions are specified, whereas for defined terms necessary and sufficient conditions are specified.

```
heavy_duty_truck   :< roadtrain_truck.
long_distance_truck :< heavy_duty_truck
has_truck_type  :<  domain(trucks)  and term_valued.
```

Given this hierarchical model we obtain the obvoius inferences (both for objects and on the terminological level)[1]:

```
ldt10000    ::   has_truck_type:long_distance_truck.
X           ?:   has_truck_type:heavy_duty_truck.
     % binds X to ldt10000.

has_truck_type:long_distance_truck  ?<   has_truck_type: road_train.
```

5.3.7 Situated descriptions

can be used to represent alternative states of affair, to model backtracking and to perform reasoning by cases. The basic idea is to describe objects relative to a situation. Standard DLs support only a single ABox (i.e. set of object descriptions), but FLEX allows partitioning of the ABox into several situations. In a configuration application, like ours, where one incrementally describes an object, this allows one to jump back to an intermediate state simply by using the respective situation.

We will now illustrate situated descriptions, i.e. the use of situations in objects descriptions, especially in our domain of automobile sales advisory. Consider the following scenario as an example “A customer wants to transport perishable vegetables in the local market (say situation s1). All of a sudden the customer changes his mind and wants to use sometimes this truck for inter regional distribution of same perishable goods too(say situation s2). Now in situation 1 the refrigerating system RS5T would have been sufficient to fulfill the customers requirement and similarly refrigerating system RS7T-v2 is required to fulfill the customers wishes in situation 2.

```
referigerating_system  :<  ctop.
  unknown_referigerating_system  :<  referigerating_system.
              RS7T-v2        ::  unknown_referigerating_system  in s1.
   s1     <<=    s2.

   RS5T            :: unknown_referigerating_system  in s2.
   RS7T-v2       ?:  unknown_referigerating_system.
        % this query fails

   RS7T-v2        ?:  unknown_referigerating_system  in s1.
   RS7T-v2        ?:  unknown_referigerating_system  in s2.
   RS5T           ?:  unknown_referigerating_system.
        % this query fails
   RS5T           ?: unknown_referigerating_system  in s1.
        % this query fails
   RS5T           ?: unknown_referigerating_system  in s2
```

[1] Note that in principle arbitrary concept or role terms, and not just names, can be specified as fillers for term_valued features. This facility provides ordering of objects to a concept hierarchy as a benefit to our domain model.

5.3.8 Weighted Defaults:

1. explicit overriding in descriptions (most of the defaults used in our domain are of this type): In order to model criteria for deciding to buy a truck or not, we used defaults in the following way:

```
boolean  :=  oneof([yes,no]).
buy_it   :<  domain(truck) and range(boolean)  and  feature.
```

In our domain one important criterion to buy is the price of a truck, which we modeled as :

```
has_price      :<    domain(truck)  and range(number)  and
          feature.
```

```
truck and the(has_price,lt(100000))  ~20~>   buy_it:yes.
```

The following object tell illustrates the effect of the default:

```
truck_1   :   has_price:98000.
```

Now ‘truck_1’ is an instance of the left-hand side of the above default and to trigger the application of the default one has to specify the defaults-option in the flexget-query:

```
flexget(truck_1,fillers(buy_it,initial),box=defaults,Results).
     % bindes Result to [yes]
```

This facilitates that the default can be straighforwardly overridden, e.g. by adding

```
truck_1   ::   buy_it:no.
```

and which yields

```
flexget(truck_1,fillers(buy_it,initial),box=defaults,Results).
     % bindes Result to [no]
```

2. overriding by strict rules (this type of defaults can help the sales person in our domain to deal with different sales situations): consider the following situation ‘‘where a sales person wants to tell the customer that the required truck is fitted with the refrigerating system of type ‘RS5T’ or ‘RS7T-v2’, so that the customer can make his/her decision to buy or not to buy the truck’’. The resulting model will look like this:

```
fitted_referigerating_system   :<   domain(truck) and range(referigerating_systems)
              and  feature.
    truck and the(fitted_referigerating_system,unknown_referigerating_system)
       =>    buy_it:no.
```

Thus having

```
truck_2   ::   has_price:98000.
```

yields

```
flexget(truck_2,fillers(buy_it,initial),box=defaults,Results).
     % bindes Result to [yes]
```

but adding

```
truck_2   ::   fitted_referigerating_system: RS7T-v2.
```

yields

```
flexget(truck_2,fillers(buy_it,initial),box=defaults,Results).
    % bindes Result to [no]
```

since 'RS7T-v2' is an unknown referigerating system in 's1'.

3. overriding by other defaults (this type of defaults help in such buying decisions as e.g. buy a truck costing less that DM 100,000.): given this type of default would mean buying any truck less than DM 100,000 regardless of its requirements or objectives. However, a decision to buy a truck dependes upon more important reasons, i.e. to set 'buy_it' to 'no' per default:

```
truck     ~100~>   buy_it:no.
truck_3       ::      truck and has_price:98000.
```

In this model two defaults are applicable to 'truck_3', one having weight 20, and other weight 100. Now according to the semantics of the weighted default, in case of conflicting defaults, default application is to be performed such that the sum of the weights of the overridden default is minimized.

```
flexget(truck_3,fillers(buy_it,initial),box=defaults,Results).
    % bindes Result to [no]
```

With the above model, we thus have to weight the defaults modeling criteria for buying a truck such that their cumulative weight is greater than 100. Following example makes it more clear, how this type of defaults have been modeled in our domain:

```
distribution_objective  :<  objective.
financial_objective     :<  objective.
cover_the_region_efficiently  :< distribution_objective.
save_operating_costs          :< financial_objective.
truck and has_objective:distribution_objective ~75~> buy_it:yes.
truck and has_objective:financial_objective ~110~> buy_it:yes.
truck and the(has_price, ge(150000) ~20~> buy_it:no.
truck and some(has_requirement,capacity) ~20~> buy_it:no.


6tons  ::  capacity
16tons  ::  capacity
truck_4 ::  has_objective:cover_the_region and has_price:98000 and
      has_capacity:6tons.
truck_5 ::  has_objective:cover_the_region and has_price:120000
truck_6 ::  has_objective:cover_the_region and has_price:120000 and
      has_capacity:6tons.
truck_7 ::  has_objective:save_operating_cost and has_price:120000.
truck_8 ::  has_objective:cover_the_region and has_price:150000 and
      has_capacity:16tons.
```

truck_7 :: has_objective:save_operating_cost and has_price:150000.

These tell yield the following results:

```
flexget(truck_4,fillers(buy_it,initial),box=defaults,Result).
    % binds Result to [yes]
  dbox:dbox_print_results(truck_4,initial).
flexget(truck_5,fillers(buy_it,initial),box=defaults,Result).
    % binds Result to [no]
  dbox:dbox_print_results(truck_5,initial).
flexget(truck_6,fillers(buy_it,initial),box=defaults,Result).
    % binds Result to [no]
  dbox:dbox_print_results(truck_6,initial).
flexget(truck_7,fillers(buy_it,initial),box=defaults,Result).
    % binds Result to [yes]
  dbox:dbox_print_results(truck_7,initial).
flexget(truck_8,fillers(buy_it,initial),box=defaults,Result).
    % binds Result to [yes]
  dbox:dbox_print_results(truck_8,initial).
flexget(truck_9,fillers(buy_it,initial),box=defaults,Result).
    % binds Result to [no]
  dbox:dbox_print_results(truck_9,initial).
```

Following table summarizes the effect of these defaults:

has_objective	has_price	has_requirement	sum of „yes“-weights	sum of „no“-weights	buy_it
—	<100000	capacity	40	100	no
—	<100000	—	20	100	no
—	≥100000<150000	capacity	20	100	no
—	≥100000<150000	—	0	100	no
—	≥150000	capacity	20	120	no
—	≥150000	—	0	120	no
Distributing_objective	<100000	capacity	115	100	yes
Distributing_objective	<100000	—	95	100	no
Distributing_objective	≥100000<150000	capacity	95	100	no
Distributing_objective	≥100000<150000	—	75	100	no
Distributing_objective	<100000	capacity	95	120	no
Distributing_objective	<100000	—	75	120	no
Financial_objective	≥100000<150000	capacity	150	100	yes
Financial_objective	≥100000<150000	—	130	100	yes
Financial_objective	≥150000	capacity	130	100	yes
Financial_objective	≥150000	—	110	100	yes
Financial_objective	<100000	capacity	130	120	yes
Financial_objective	<100000	—	110	120	no

Fig. 5.9: effects of defaults.

5.4 Domain Modeling in CLASSIC

The CLASSIC knowledge representation system belongs to a new generation of description based knowledge representation systems. It emphasizes simplicity of the description language and completeness and tractability of its inference algorithms. CLASSIC is a full-featured knowledge representation system inspired by KL-ONE and most immediately descendent from KANDOR [Patel-Schneider 1984]and is closely related to BACK [Peltason et al.'87] & FLEX [Quantz '95]. In CLASSIC a knowledge-base is treated as a deductive database, in our case one with an object-centered flavor. Because of its intended role as a database-style repository, CLASSIC intentionally limits what the user can say. As a benefit all inferences can be done in a timely manner.

CLASSIC, being a knowledge representation system, stores a collection of information and determines the consequences of this information. As it is based on description logic, the information stored by CLASSIC consists of statements and other expressions from this description logic and the consequences computed by CLASSIC are those sanctioned by the description logic. CLASSIC, being a full-featured knowledge representation system, allows for both the addition and the retraction of information. It can also explain how it obtained the consequences it computed.

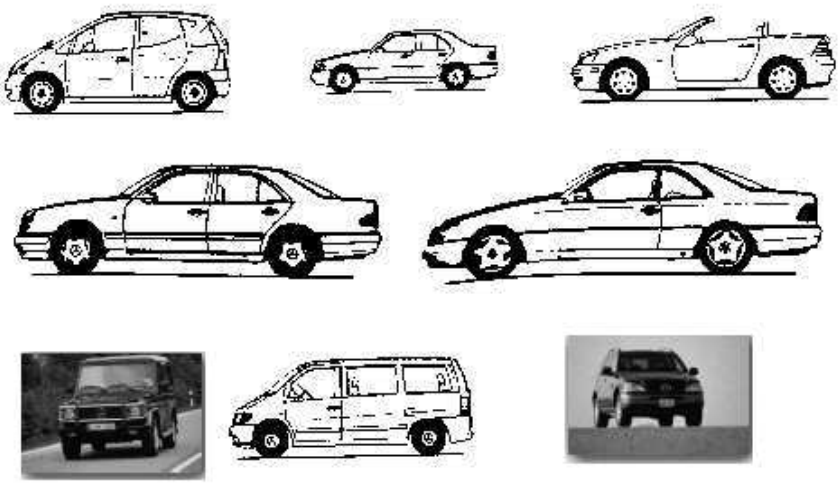
CLASSIC has a number of novel features that distinguish it from other KL-ONE-like systems[see for details, Borgida et al., 1989; Brachman et al., 1990] but apart from that it provides a number of extra-logical features, including multiple knowledge bases, forward chaining rules, procedural extensions to the description logic in the form of test functions and computed rules, and hooks that allow programs to be informed of the actions that CLASSIC performs.

The CLASSIC family of Description Logic-based knowledge representation systems represent information about a domain in terms of descriptions, concept, roles, rules and individuals. NEOCLASSIC is the most recent member of the CLASSIC family and heavily uses C++ object oriented programming. Though it features a tight integration with C++ programming language but allows a close control of its knowledge representation capabilities without compromising their integrity.

The LISP version of classic is relatively expressive. It is based on normalize-compare concept and is incomplete like BACK and FLEX(++). Parallel to LISP-version there exists a C++ version NEOCLASSIC, whose expressive power is not so strong. CLASSIC has following characteristics:

- all, atleast, atmost, same-as
- only primitive roles without the value restrictions for both Domain as well as Range
- Inverse roles can explicitly be specified
- Roles could be features, thus they are specified outside the role hierarchy i.e. that it is not possible that a role has a feature as its subrole
- SAME-AS as feature thread
- Three types of rules
- Revision

In the next section, we concentrate less on these characteristics individually and focus instead on how to make good use of CLASSIC for our domain (Automobile Sales Advisory). Further, we will shortly discuss the above mentioned extra logical features provided by CLASSIC in our context (i.e. in sales advisory). Here, we'll be using the examples from the passenger-car-world (here onwards referred to as cars), which forms one of the less complex (in the sense that it is easy to understand for a layman) cross-section of our domain (fig 5.10).



SALOONS; ESTATE-CARS; ROADSTER & CABRIOLETS; COUPES; OFF-ROADER; CROSS-COUNTRY; RECREATION-VEHICLES; LARGE-SALOONS

Fig. 5.10: Various types of passenger cars

5.4.1 Modeling in CLASSIC

Once again we start with the most general information from our domain and this time the different models of the cars, their accessories and their manufacturing constraints, prices and information regarding their availability form the basis of our domain model. In other words the car world serves as the cross-section of the domain to be represented. Here, we give the sample representation from our domain and explain all the extra logical features of CLASSIC in the context of automobile sales advisory. Fig. 5.11 shows the first three levels of the hierarchy of our sample KB.

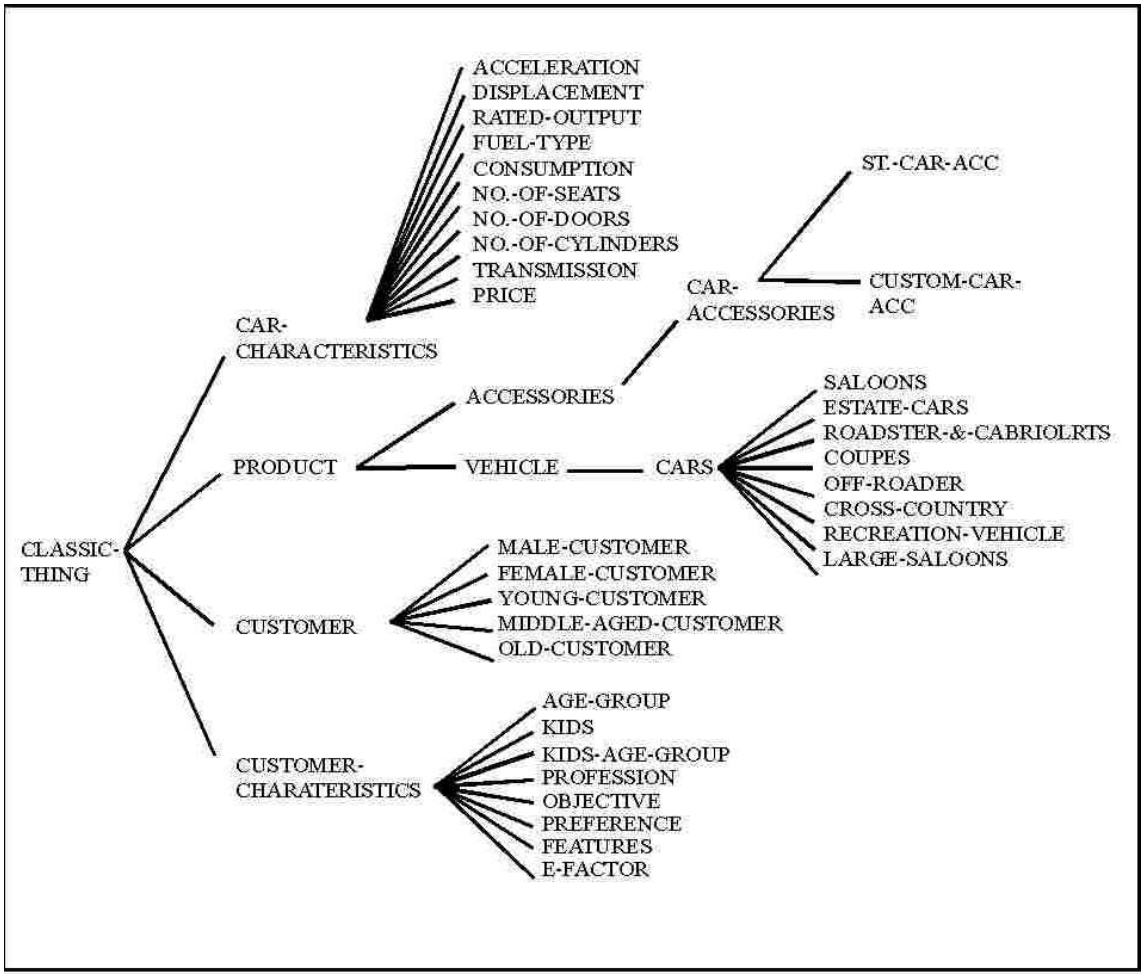


Fig. 5.11: First three levels of the hierarchy of our sample KB

Concepts and descriptions in CLASSIC are divided into two realms: **CLASSIC** and **HOST**, **CLASSIC** concepts are used to represent classes of real-world objects of a domain (e.g. in our case product, vehicle, accessories, customer etc.)., while **HOST** concepts are used to describe objects in the host language (e.g. LISP or C++), such as NUMBERS, LISTS, ATOMS, STRUCTURES and STRINGS.

5.4.2 The Sales Advisory Knowledge Base

A knowledge base is simply a collection of concepts, individuals, roles and rules, and the relationships between them {CLASSIC ref. Manual}.

Thus, We'll start our modeling task by defining the concepts and individuals from or domain of automobile sales advisory. Then the roles, which specify the properties of individuals and then all the restrictions which are posed to these concepts, individuals and roles. Subsequently we'll define rules to enhance the concepts, adding restrictions, fillers, parents etc.. Basically, rules are being used to add to an individual information that is not definitional. CLASSIC provides three different types of forward-chaining rules, which are referred to as simple rules, description rules and filler rules.

5.4.3 Notation

Before we start giving the examples from our actual knowledge base here we would like to give the reader the following conventions(as suggested in CLASSIC tutorial) used to make our definitions clearer:

KB entities:

concept names are in upper case, e.g. CARS
individual names are capitalized, e.g. C-190, D-220
role/attribute names are in lower case, e.g. no._of_doors, fuel_type
partition names (for disjoint primitives) are in lower case, e.g. type
indices (for primitive and disjoint primitives) are also in lower case

CLASSIC system:

CLASSIC function names are in lower case, e.g. cl_define_concept
CLASSIC language keywords are in upper case, e.g. AT-MOST

LISP:

LISP built-in functions and primitives are in lower case, e.g. defun
LISP user functions are capitalized, e.g.

5.4.4 Concepts

To begin with, we first wrote down a list of all types of objects, without making any fine-grained distinctions, which are going to play an important role in modeling our domain. Then by considering this list we made a major cut by distinguishing between objects that had independent existence and those which depend on other objects for their existence. The former are concepts and latter roles in our KB. In general there are three types of concepts: primitive concepts, disjoint primitive concepts and defined concepts. CLASSIC offers three functions for defining these concepts: cl-define-primitive concept, cl-define-disjoint primitive-concept and cl-define-concept.

In our domain following four concepts PRODUCT, CUSTOMER, CAR-CHARACHERISTICS and CUSTOMER-CHARACTERISTICS are defined as disjoint primitive concepts[1] as children of CLASSIC-THING within a disjoint grouping called classic-thing-type. The following example shows these definitions in CLASSIC syntax:

(**cl-define-disjoint-primitive-concept** 'PRODUCT 'CLASSIC-THING 'classic-thing-type)
(**cl-define-disjoint-primitive-concept** 'CUSTOMER 'CLASSIC-THING 'classic-thing-type)
(**cl-define-disjoint-primitive-concept** 'CAR-CHARACTERISTICS 'CLASSIC-THING 'classic-thing-type)
(**cl-define-disjoint-primitive-concept** 'CUSTOMER-CHARACTERISTICS 'CLASSIC-THING 'classic-thing-type)

Similarly, the disjoint primitive concepts MALE-CUSTOMER and FEMALE-CUSTOMER under CUSTOMER in the gender disjoint grouping and YOUNG-CUSTOMER, MIDDLE-

[1] Note: A disjoint primitive concept is just like a primitive concept, except that it belongs to one or more disjoint groupings, and any concept within the same disjoint grouping are known to be disjoint from each other (classic reference manual p.12).

AGED-CUSTOMER and OLD-CUSTOMER under CUSTOMER in the age disjoint grouping. The following example should make the model clearer:

(**cl-define-disjoint-primitive-concept** 'MALE-CUSTOMER 'CUSTOMER 'gender)
(**cl-define-disjoint-primitive-concept** 'FEMALE-CUSTOMER 'CUSTOMER 'gender)
(**cl-define-disjoint-primitive-concept** 'YOUNG-CUSTOMER 'CUSTOMER 'age)
(**cl-define-disjoint-primitive-concept** 'MIDDLE-AGED-CUSTOMER 'CUSTOMER 'age)
(**cl-define-disjoint-primitive-concept** 'OLD-CUSTOMER 'CUSTOMER 'age)

We then introduced CAR and CAR-ACCESSORIES as disjoint primitive concepts under PRODUCT. Where CAR has following description:

CAR: (**AND** VEHICLE
(**AT-LEAST** 1 fuel-type) (**AT-MOST** 1 fuel-type)
(**ALL** fuel-type(**ONE-OF** Diesel Electric Petrol))
(**AT-LEAST** 1 has-engine) (**ALL** has-engine ENGINE)
(**AT-LEAST** 1 no.-of-doors) (**AT-MOST** 1 no.-of-doors)
.....
(**ALL** no.-of-doors (ONE-OF 2 4 5)))

Note: Primitive concepts in CLASSIC are those concepts which cannot be fully specified by necessary and sufficient conditions. The function **cl-define-primitive-concept** was used to define a primitive concept (CAR). The syntax is

(**cl-define-primitive-concept** <symbol> <CLASSIC-descr>)

where the symbol is the name of the concept being defined and the description is the concept definition, and is formed using the grammar in Appendix 2.

So far, we have created not only atomic primitive concepts but much more complex, but still primitive concepts, such as CAR. Once we have the basic concept of a CAR defined, we can describe the more special types of cars we would like to recognize automatically. The following example illustrates some fully-defined car subconcepts.

(**cl-define-concept** 'SALOON
'(AND CAR (FILLS no.-of-doors 4)))
(**cl-define-concept** 'ESTATE
'(AND CAR (FILLS no.-of-doors 5)))
(**cl-define-concept** 'COUPE
'(AND CAR (FILLS no.-of-doors 2)))

In this way other subconcepts, MICRO-COMPACT-CAR, ROADSTER&CABRIOLETS, OFF-ROADER, CROSS-COUNTRY, REC.-VEHICLE & LARGE-SALOONS were created.

ROLES: Once the concepts and individuals are defined then to specify the properties of an individual we make use of roles. In other words Roles are entities that represent the properties of **CLASSIC** individuals. They map **CLASSIC** individuals to other (**CLASSIC** or **HOST**) individuals. The roles of **CLASSIC** individual can either be „filled" by individuals (called the role fillers) or have their potential fillers restricted by certain concepts (i.e. as type descriptions), or both, where the fillers and descriptions can be in either the **CLASSIC** realm

or the **HOST** ream. *Attributes* are special types of roles that have an implicit maximum number of fillers of 1. Further, a role can have an *inverse* as well.

The syntax for defining a role is

(**cl-define-primitive-role** <symbol> &key[1] parent inverse (attribute NIL)
(inverse-attribute NIL)
inverse-parent)
where the symbol is the name of the role.

The following examples from our domain should make the use of roles in our domain clearer to our reader. Here we defined the technical characteristics e.g. acceleration (i.e. time taken from 0 to 100 km/h), displacement (in cc), related-output (in hp or kW) etc.[2]

(**cl-define-primitive role** 'acceleration :attribute t)
(**cl-define-primitive role** 'displacement :attribute t)
(**cl-define-primitive role** 'related-output :attribute t)

5.4.5 RULES

are used in our domain to represent the manufacturing constraints. Here, we will try to explain that how all types of manufacturing constraints can be translated into CLASSIC rules. CLASSIC provides three different types of forward-chaining rules: (simple) *rules*, *description rules* and *filler rules*.

A (simple) *rule* consists of an antecedent, which must be a classified concept, and a consequent, which is also a concept, but not necessarily classified. The function
cl-add-rule creates a simple rule. Its syntax is

(**cl-add-rule** <symbol> <classified derived **CLASSIC** concept>
< CLASSIC-descr> &key filter comment)

EXAMPLE: suppose we want to create a LUXURY-CAR rule, which states that all cars with displacement above 4.0liters are s-class-saloons. Suppose that the concepts CARS and S-CLASS-SALOON exist. We would create a rule with antecedent CARS, consequent S-CLASS-SALOON, and filter (ALL displacement (MIN 4.0)). There is no need to create the concept CARS-WITH-DISPLACEMENT-ABOVE-4.0 in order to create this rule. To create the LUXURY-CAR rule above, this function would be called as follows:

(**cl-add-rule** 'LUXURY-CAR @CARS 'S-CLASS-SALOONS
:filter '(ALL displacement (MIN 4.0))
:comment "A car with displacement of at least 4.0 is likely to be
a s-class saloon.")

[1] The &key keyword, used in function headers, says that the following arguments may optionally be specified by the user.

[2] **NOTE:** Similarly, age-group, kids, kid-age-group etc. were created as roles for CUSTOMER.

A *description rule* is similar to the simple rule, except that instead of the rule being defined with a consequent that is specified at rule creation time, and then merged into an individual when the rule is fired, the rule is defined with a function and arguments, which, when the rule is fired, generate a concept description to be merged into an individual.

A description rule is defined with the function **cl-add-description-rule**. The syntax is

(**cl-add-description-rule** <symbol> <classified derived **CLASSIC** concept>
< CLASSIC-desc> &key filter comment)

EXAMPLE: Suppose that we have a primitive concept PRODUCT with the following description

(AND (AT-LEAST 1 price)
(ALL a INTEGER)
(AT-LEAST 1 vat)
(ALL vat INTEGER)

where *price* and *vat* are attributes. We want to enforce the constraint that filler of *vat* can be no larger than filler of *price*.

To implement this constraint we used the function **generate-vat-max** to generate ALL restriction on b:

```
(defun generate-vat-max (ind)
  (let* ((a-role (cl-named-role 'price))
    (a-filler (first (cl-fillers ind a-role))))
    `(ALL vat (MAX ,a-filler))))
```

Now one must define a filter function which determines that there is currently a filler for a given role. This function returns T if there is currently a filler, NIL if there can be no fillers, and ? otherwise.

```
(defun cl-test-role-filled? (ind role-name)
(let* ((role (cl-named-role role-name)))
   (cond (( cl-fillers ind role) T)
    ((cl-ind-closed-role? ind role) NIL)
    (T '?))))
```

Then the **generate-vat-max-rule** description rule can be defined as follows:

```
(cl-add-description-rule  'generate-vat-max-rule  @PRODUCT{p}
            'generate-vat-max   :filter '(TEST-C  cl-test-role-filled?  price))
                                  :comment "vat's filler <= price's filler"
```

A *filler rule* is like a description rule, except that in addition to the function and arguments, it takes a role, and the function generates a list of fillers for the role when the rule is fired.

A filler rule is defined with the function **cl-add-filler-rule**. The syntax is

(**cl-add-filler-rule** <symbol> <classified derived **CLASSIC** concept>
<role> <fn> &key args filter comment)

EXAMPLE: Suppose we know that a rear_door (REDOOR) of a coach, has a height which is 2 times its width, where *height* and *width* are attributes, and REDOOR is a concept. One would define the function calculate-length as follows:

```
(defun calculate-height (ind height-role)
  (declare (ignore height-role))
  (let* ((width-role (cl-named-role 'width))
         (width (first (cl-fillers  ind  width-role))))
    (list (* width 2))))
```

One would then add the rule as follows:

```
(cl-add-filler-rule  'redoor-height  @c{REDOOR}  @r{height}
          'calculate-height  :filter'(TEST-C  cl-test-role-filled?  width))
```
[1]

5.4.6 Extra Logical Features:

As in any application, one needs a domain ontology in which to work. Now, that our automobile sales advisory application contains a knowledge base including a concept taxonomy and instance descriptions, we'll discuss the following extra logical features instead of going into further details of modeling and provide some examples in our domain that illustrate each of these concepts:

5.4.7 Explanation:

CLASSIC can justify all its belief or in other words it can explain how it obtained the consequences it computed. Not only can the user view any piece of information, he/she can also have any deduction explained. In our example, if the user asks.....
The explanation facility can also answer other questions such as why one object does or does not „subsumes" (is or is not more general than) another object, why a rule fired on an object, or why an error occurred (i.e. explanation can be used to help explain error conditions and incoherencies). Inferences can also have templates associated with them that may be used to present explanations in terms that are acceptable to the user.

Now, viewing explanations in CLASSIC from the sales advisory application point of view, only the first type of explanation facility that allows the user to ask for an explanation of anything that CLASSIC has derived about a CLASSIC object (concept individual or consequent of a simple rule), is of importance. Therefore we'll discuss it in little bit of detail. In this context we'll not consider the explanation used to understand errors (i.e. error conditions and incoherencies) & which are particularly used to debug the knowledge base system.

CLASSIC, for first type of explanation facility, provides two different kinds of explanation functions: normalization functions and subsumption explanation functions.

---

[1] Note: An illegal use of a filler rule would be a rule that fires, and the function generates NIL as the result, but when more information is added to the individual, the function generates a list of individuals. In this case, a filter should be used to keep the rule from firing until enough information is in the individual for the function to produce the final result.

5.4.7.1 Normalization Explanation Functions:

The normalization explanation inferences (those calculated by these functions) explain where a particular piece of information on an object came from. The main explanation function is **cl-exp**[1]. The syntax is

(**cl-exp** <object> &key (role-path T) (aspect T) (aspect-filler T) (all NIL))

As an example of using **cl-exp**, we could print an explanation of the AT-LEAST restriction for saloon_190_E's radio's blaupunkt_radio as follows

(**cl-exp** @Saloon_190_E :role-path (list @st_accessory @radio) :aspect 'AT-LEAST)

We would print an explanation of why blaupunkt_radio is a st_accessory, as follows

(**cl-exp** @st_accessory :role-path (list @radio) :aspect 'FILLS :aspect-filler @blaupunkt _ rad i o)

5.4.7.2 Subsumption Explanation Functions:

These functions provide explanations for why one object subsumes another and in particular, for each piece of information on the subsuming concept one can explain why the corresponding piece of information on the subsumed concept is at least as specific as information on the subsuming concept. For example consider the following two concepts:

CUSTOMER-1 = (**AND** PERSON
(**AT-LEAST** 1 car)
(**ALL** car MERCEDES))

CUSTOMER-2 = (**AND** PERSON
(**AT-LEAST** 3 car)
(**ALL** car MERCEDES-SALOON-400-SL)
(**ALL** friend (ALL own Mercedes-Luxry-Class)))

If we now ask the system to explain why C1 subsumes C2 it will return the following information (in an edited form):

primitives: (PERSON CLASSIC-THING)
PRIMITIVE-SUBSET
Role Restrictions:
CAR
at-least (3):
AT-LEAST-ORDERING: At-least : 3 satisfies at-least : 1
all: MERCEDES-SALOON-400-SL

[1] **Note:** cl-complete-exp and cl-set-up-exp are other functions which calculate the normalization explanation information for the object.

SUBSUMING-ALL-RESTR: MERCEDES subsumes
MERCEDES-SALOON-400-SL.

This information first states that the set of primitives on CUSTOMER-2 is at least as specific as the primitives on CUSTOMER-1 since the list of primitives on CUSTOMER-1 = {PERSON, CLASSIC-THING} is a subset of the set of primitives on CUSTOMER-2 = {PERSON, CLASSIC-THING}. It indicates that the **AT-LEAST** restriction on CUSTOMER-2 is more specific than the **AT-LEAST** restriction on CUSTOMER-1, and that the **ALL** restriction on the car role is more specific on CUSTOMER-2 since MERECEDES subsumes MERCEDES-SALOON-400-SL. Note that the CUSTOMER-2's friends own a Mercedes luxury class cars is ignored since there is no corresponding constraint on CUSTOMER-1 and extraneous information on the subsumed concept is ignored since it can only make the concept more specific.

5.4.8 Multiple Knowledge Bases:

Classic maintains a collection of named knowledge bases. One of these knowledge bases is the current knowledge base. This idea of having multiple knowledge bases is of interest to a domain like ours (automobile sales advisory), because due to large number of vehicles and their various models and their respective accessories (standard & custom) plus other various price and manufacturing constraints, a single knowledge base would become to large and complex which further leads to maintenance problems. Further the automobile industry is a very dynamic industry where new models of the vehicles and their respective accessories are constantly being introduced and with the same speed old models are being phased out. This poses another requirement for a knowledge base which has to be constantly updated. In our case 30-40% of the knowledge incorporated in the KB is to be changed every year.

A CLASSIC knowledge base continually computes the consequences of all information it contains. Any operation that would result in an inconsistent knowledge base is rejected. Part of this information is a classification and instantiation taxonomy of concepts and classic individuals. The roles, concepts, rules and classic individuals in a knowledge base can be retrieved by name. Other operations on the collection of knowledge bases are:

- initializing the collection of knowledge bases
- cleaning up the collection of knowledge bases by destroying them all
- retrieving the current knowledge base
- setting the current knowledge base
- creating a knowledge base
- finding a knowledge base by name
- removing a knowledge base from the collection of knowledge bases

5.4.9 Forward Chaining Rules:

Rules in CLASSIC represent simple-forward chaining rules that run only on named classic individuals in a domain. A rule in CLASSIC is a formal object that represents a mapping between the representation of its antecedent and the representation of its consequent. A rule requires that all CLASSIC individuals in the knowledge base that belong to its antecedent also belong to its consequent.
The consequent of a rule can be specified as a description, as a function from classic individuals to descriptions or as a function from classic individuals and roles to sets of individuals. In the first case the consequent of the rule is the description provided, in the

second case it is the description returned when the function is run and in the third case the description is a fills description created from the role and the set of individuals.

5.4.10 Procedural Extensions

(test functions and computed rules): Test restrictions allow CLASSIC to use procedures to specify concepts. In this way it is possible to extend the CLASSIC operators available. This is useful for number of reasons. First, it allows CLASSIC to express concepts that it cannot with its other operators. Second, it allows CLASSIC to manipulate structures that it could not easily do otherwise.

In our domain both test functions and computed rules find application to do the necessary price calculations (see section 5.2.4) and for doing date reasoning e.g. for the availability of different models and their respective accessories (see section 5.2.5). Though, in general doing date reasoning correctly and completely is a very difficult task and the amount of reasoning necessary can require a great deal of time. Therefore in our application we use operators like date-before, date-after and date-between, which would be more than sufficient.

Examples:

500SL in color red can only be delivered after 15.8.99
digital_display for 500SL is available between 4.4.99 - 15.9.99
500SL in any other color can be delivered before 31.7.99

5.4.11 Hooks:

that allow programs to be informed of the actions that CLASSIC performs. CLASSIC provides a facility that allows functions from outside CLASSIC to be run in response to particular activities performed by CLASSIC. These activities include the creation of roles, concepts, rules and individuals; the addition of information to or the retraction of information from an individual; the firing of rules; the transfer of information from one individual to another and the discovery of a new individual that satisfies a concept. This facility can be used to build user interfaces that maintain an up-to-date view of the CLASSIC knowledge base.

6 Evaluation & Discussion (related work)

Once the Framework has been laid down for Knowledge Base Consultation systems, where the system is based upon two major hypotheses, one which says that success of any counseling/advisory system pre-supposes a strong and descriptive product model, hence a descriptive knowledge representation and second it has to be integrated to its environment, hence the socio-cognitive approach. In the penultimate section, we have provided the specifications for such a consultation system. In this chapter, we compare and analyze the related work to ours in order to discuss the differences, commonalties and their benefits.

Our work relates to several fields of research in AI & Databases. We shortly discuss the relationship to configuration, to knowledge representation systems (especially to description logics) and to other automobile sales advisory systems (e.g. SalesPlus from BAAN Denmark).

6.1 Configuration

After decades of research in AI/Knowledge Based Systems, there is now a better understanding of the strengths and shortcomings of the technology. There are fewer exploratory users and more users demanding practical systems. Thus a wide variety of knowledge-based systems. The systems generally fall into a few categories: configuration, diagnosis & troubleshooting, planning & scheduling, process monitoring & control and software engineering.

Configuration tasks have a long history in AI going back at least to the pioneering R1/XCON expert system for configuring computer systems. Recently the area has been revitalized by the renewed industrial interest [Feigenbaum '93, AAAI '96 & IEEE expert '98] as companies are experiencing a manufacturing paradigm shift from mass production to mass customization. One of the critical elements manufacturers need, to make the transition from mass production to mass customization, is a configurator. Configurators are software applications that enable customers to order exactly what they need based on allowable choices.

6.1.1 What is configuration?

Configuration involves selecting and arranging parts to fit problem constraints. Informally, configuration can be defined as a special case of design activity with the following key features:

- The artifact being configured is assembled from instances of a fixed set of well defined component types, and
- Components interact with each other in predefined ways [Mittal & Fraymann '89].

In other words, selecting and arranging a combination of parts which satisfy given specifications is the core of a configuration task [Sabin & Weigel '98]

In **mass production**, pre-packaged configurations equate to individual bills of material in the MRP/ERP system. This assumes that a manufacturer can identify specific configurations best suited for specific customer needs. Usually, engineering and marketing meet and agree to restrict the choices that a customer is permitted to make.

In **mass customization**, there are no pre-packaged configurations and therefore no bills of material representing each order configuration. Mass customization views the probability of

any two order configuration being identical as coincidence. The customer's order requirements are derived from the configurator. Decisions are made on the limits described within the configurator. The order itself (the sales order in the MRP/ERP system) captures and identifies the elements needed to satisfy the customers requirements and subsequently transmits this information to the factory. This is accomplished without the need for a unique bill of material describing each customers unique order requirements.

6.1.2 Existing Paradigms

The specification of configuration tasks, in general, involves two distinct phases, the description of the domain knowledge and the specification of the desired product. The domain knowledge describes the objects of the application and the relations among them. The specifications for an actual product describe the requirements that must be satisfied by the product and, possibly, optimizing criteria that should be used to guide the search for a solution. The solution has to produce the list of selected components and, equally important, the structure and topology of the product. The following sections contain examples of such configurations

6.1.2.1 Rule-based Reasoning

here the systems, following the Digital's landmark R1/XCON system, use production rules as a uniform mechanism for representing both domain knowledge and control strategy. The production-rule programming paradigm explicitly provides the dynamic, runtime decision-making essential for coping with the following characteristics:

what actions must be performed to obtain a valid configuration and when an action can appropriately occur in relation to other actions.

These systems derive solutions in a forward-chaining manner. At each step, the system examines the entire set of rules and considers only the rules it can execute next. Each rule carries its own complete triggering context, which identifies its scope of applicability. The system then selects and executes one of the rules under consideration by performing its action part.

The lack of separation between domain knowledge (compatibilities, dependencies, and so on) and control strategy (procedural knowledge) and the spread of knowledge about a single entity over several rules make the knowledge-maintenance task extremely difficult.

6.1.2.2 Model-based Reasoning

The main assumption behind the model-based reasoning is the existence of a system's model, which consists of decomposable entities and interactions between their elements. According to Walter Hamscher [Hamscher '92, '94] the most important advantages of model-based systems are:

- a better separation between what is known and how the knowledge is used.
- enhanced robustness (increased ability to solve a broader range of problems)
- enhanced compositionality (increased ability to combine knowledge from different domains within a single model), and
- enhanced reusability (increased ability to use existing knowledge to solve related classes of problems).

The main motivation for using model-based configuration systems lies in the nature of configuration, which is by definition a synthesis task. Therefore the ability to cover the entire range of possible solutions systematically is essential. Following are some model-based approaches to configuration:

6.1.2.3 Logic-based approaches

One important family of logic-based approaches is based on description logic. DLs are formalisms for representing and reasoning with knowledge. They unify and provide a logical basis for well-known traditions of frame-based systems, semantic networks and KL-ONE-like languages, object-oriented representations, semantic data models, and type systems.
The clear semantics and simple logical operations have made description logic popular in theoretical studies as well as practical applications (e.g. configuration). DLs can offer support for such (configuration) applications both during the knowledge-acquisition phase and the problem-solving phase.

The advantages of DLs during the knowledge acquisition phase are clear. The classification facility automatically organizes descriptions into an explicit taxonomy, based on subsumption inferences. In addition, DL systems automatically maintain consistency as new descriptions are classified and added to the knowledge base, or existing descriptions are modified and reclassified.

During the problem-solving phase DLs can support configuration in one of two ways. The first possibility is for the DL system to provide runtime support for other configuration engines. The organization of subsumption-based taxonomy enables efficient retrieval of descriptions. In addition DLs can deal with partial, incomplete descriptions (of systems, component types, functionality and so on). Furthermore, DL extensions can provide special types of reasoning - for example, explanation [Mcguinness & Borgida ’95].

The second possibility is for the DL system to solve the entire configuration problem [Wright et al.’95]. The terminological knowledge base contains descriptions of the classes (component types) as well as rules. These rules, attached to concepts on different levels of abstraction, can extend and refine an individual configuration as required. (More sophisticated systems might use an external, domain-specific control mechanism to increase the efficiency). After the interface has guided the user through some simple questions, the system uses these inputs to make initial selections. The application through follow-up questions guided by CLASSIC, arrives at a complete (abstract) solution.

This approach has one potential drawback: the trade-off between the efficiency of the reasoning tasks and the expressiveness of the knowledge-representation tool is crucial. If the formalism aims at a certain level of expressiveness (by allowing existential quantifications or disjunctions, for example), subsumption becomes NP-complete. On the other hand, restricting expressiveness to ensure tractability makes the formalism unable to represent complex systems, which is often necessary in representing practical configuration tasks.

Other examples of logic-based systems for configuration include Prose [Wright et. Al ’93] and Beacon [Searls & Norton ’90].

6.1.2.4 Constraint-based approaches

The first attempt to define a generic, domain-independent model for configuration tasks - and one of the most important works towards a general model of configuration - was presented by Mittal and Fraymann [Mittal & Fraymann '89]. In their work they identify the main characteristics of configuration tasks and introduced the definition (as given above).

In their approach, each component is defined by a set of properties and a set of ports for connecting to other components. Constraints among components restrict the ways various components can be combined to form a valid configuration. In addition to the components description, the specification for a configuration task also includes a description of the desired product and optimization criteria.

Given a specification, the goal is to build one or more configurations (that is, a set of components and a description of the connections among them) that satisfy both the specification and optimization criteria. If none exist, the goal is to detect inconsistencies in the requirements.

This restricted form of the configuration task can be represented as the constraint satisfaction problem [Tsang '93], in which components and their ports are the variables and the constraints restrict the way components can be integrated in a solution.

Because the mapping between functional roles and the set of components available is typically many-to-many, the configuration task is dynamic in nature. To cope with this situation Mittal and Falkenhainer introduced an extension to the classical constraint-satisfaction problem paradigm, called dynamic-constraint satisfaction problems [Mittal & Falkenhainer '90]. Since then lot of work has been done on the advantages and to improve upon the limitations of the framework presented so far (see Bowen & Bhaler '91; Haselblock '93; Freuder '92].

Sabin and Freuder [Sabin & Freuder '96] propose a different configuration framework, based on the notion of composite constraint satisfaction. A composite CSP allows values for variables to be entire subproblems. When a variable is instantiated, the variables and constraints representing the composite value are created and added to the constraint network. Object-oriented programming offers a natural implementation framework for composite CSP. (See Freuder & Sabin '97; Sabin & Freuder '97; Sabin & Freuder '97 also].

6.1.2.5 Case-based reasoning:

In the case-based approach, one tries to solve the current configuration problem by finding a similar, previously solved problem and adapting it to the new requirements. CBR relies heavily on a tacit assumption that similar problems have similar solutions. Thus, if we retrieve a configuration similar to the current configuration, we can make only minor adaptations. The basic processing cycle in a CBR system is:

- *Input customer requirements*. The goal of this first step is to extract the features of the input to be used for the retrieval of a similar case.
- *Retrieve a configuration*. Configurations in the case base must be labeled in a way that makes it easy to find them when needed.
- *Adapt the case to new situation*. The central issues for case adaptation are determining the kind of knowledge necessary for the adaptation and how to control the adaptation process.

- *Store the new configuration*. Each time the system solves a new configuration problem, it can store that problem and its solution in the memory as a case.

6.1.3 Discussion:

A technical configurator isn't just for sales organization: A technical configurator might give one the opportunity to achieve greater efficiency in engineering, manufacturing and sales but it would be a mistake to view a configurator as a tool just for sales. In sales, configuration is typically a high-level configuration in which an external problem solver — usually salesperson or customer — interacts with the configurator to make the creative decision. Problem solvers need to perform this kind of configuration on their laptops as well as from within the MRP/ERP system. Also, high level configuration over the internet is gaining importance.

Low-level Configuration: in contrast, is manufacturing-oriented — a non-interactive procedural process that selects the necessary parts, checks their availability and determines the necessary routings at a level of detail that is no longer interesting to the customer. A customer can specify leather seats for a car, for example, but would hardly be allowed to specify the parts needed to mount them, nor would he want to. Low-level configuration is possible only in the integrated MRP/ERP system.

High-level (Sales) Configuration: the primary task of a configurator is to support external problem solvers in finding a suitable variant. Where each variant must be functionally complete, technically feasible and satisfy the customer's requirements as far as possible.

6.2 Knowledge Representation Systems (especially Description Logics)

Description logics are logics for representing and reasoning about classes of objects and their relationships. They can be seen as successors of frame systems and semantic networks and have been investigated for more than a decade from different points of view, in particular, expressive power and computational complexity of reasoning. Further, DLs have been successfully used in the following broad areas of expert systems[Swartout et al '91] especially configuration expert systems [Wright et al.'93]; conceptual retrieval/knowledge base browser applications [Brachman et al. '93, Devanbu et al. '90, Selfridge '91, etc.]; natural language/translation [Kelly et al. '93, Nonnenman et al. '92] & Planning [Devanbu et al. '91, Johnson et al. '89]. Traditional application areas also include software engineering, databases and information systems. In this section, I'll not describe how DL's have been applied in several traditional fields, including software engineering, configuration management, databases and information Systems but how DL's have been applied in several newer fields including indexing, domain engineering, dialogue manager & tutoring systems.

Following case studies illustrate how to tailor representations (admissible languages, in our case, DLs) to different applications:
Planning in DL
Using DL for Indexing Audiovisual Documents.
DL Based Support to Domain Engineering
Combining Expression and Content in Domains for Dialogue Managers
Extending Tableaux Calculus with Limited Regular Expression for Role: An Approach to NL Processing.

6.2.1 Planning in DL

Planning is the problem of synthesizing a course of action that, when executed, takes an agent from given initial state to a desired goal state. This is an important part of intelligent agency and has thus received significant attention with in AI for more than 30 years. A large number of algorithms, with differing empirical tradeoffs, have been developed over this period (see [Khambapati'97], where he discusses about the classical planning problem in AI and also provides a chronology of the many existing approaches).

The research in this area is far from complete, many exciting new algorithms are continuing to emerge in recent years (see [Weld'98], for recent advances in AI Planning). There have been various formulations that attempt to solve the problem e.g.
Logic based approaches
Operator-based approaches
Time-based approaches
Case-based approaches
Constraint-based approaches
Distributed planning
Reactive approaches.

For a general introduction to problem solving and then putting planning, scheduling, learning etc. in the context of problem solving, see [Kumar'95]. Planning or *classical planning problem*, in general, can be specified by describing the initial state of the world, the desired goal state and a set of deterministic actions. The objective is to find a sequence of these actions, which, when executed from the initial state, lead the agent to the goal state [Khambapati'97]. Now, if we look at the way the classical planning is modeled, we come across several different paradigms (e.g. task-based planning, action-based-planning and logic-based planning).

Over the last few years there has been a growing interest in reasoning and representing about actions and planning (i.e. which representations to use for planning). In this context, the use of Description Logics has been proposed to represent epistemic operators, actions and plans [Artale and Franconi'94, Badea'95, Devanbu & Litman'91]. For an in-depth analysis of the various approaches to encoding actions and planning in Description Logics see [Badea'99].

6.2.2 Indexing Audiovisual Documents

In their recent publication Carrive and Colleagues [Carrive et al. '98] address the problem of indexing broadcast audiovisual documents such as video, films, news etc.. They understand indexing „as the operation which allows whole or part of a document to be the result of a request". In practice, the indexing of such documents can be done from simple methods such as associating a few keywords with a whole document to much more sophisticated ones, such as describing a document deeply with conceptual graphs [Simonnot '96]. They make use of a collection of so called shots[1] to build automatically high level descriptions of subsets of this collection which can then be used for annotating, indexing and accessing the document. As temporal dimension is a specific characteristic of all audiovisual documents, thus they propose to represent documents and high level descriptions with the framework of description logics

[1] Shots are usually considered to be the smallest syntactic units of film language [Katz '91] and defined as what is filmed during one run of camera without edit.

enriched with temporal relations. To start with, they define this problem as a classification problem (classifying temporal structures) and the propose an algorithm to automatically classify sub-sequences of shots based on a bottom-up construction (using multi-layered information) of descriptions using the rule mechanism of CLASSIC system.

6.2.3 Domain Engineering

Compatangelo and his colleagues [Compatangelo et al '98] believe that an automated support tool for Domain Engineering should behave like an Intelligent Knowledge Management Environment and not as an expert system. It should only help the analyst in the discovery and in the explication of contradictions, redundancies and hidden properties detected in the domain knowledge base under development and/or analysis. In their view, present IKMEs for domain engineering suffer due to the mixing up the following two representation levels:

1. a user-oriented conceptual level where domain specific elements are represented (e.g. data, processes and flows in Data-Flow diagrams;
2. an underlying logic-oriented epistemological level where inferences are drawn from general-purpose elements (e.g. DL concepts and roles).

Further, the explicit separation of operational tasks between humans and computer systems should imply a corresponding separation of levels (1) and (2) in support tools. The authors, however, claim that this is not the case, that is one of the two above levels is not explicitly present (i.e. embedded in most of present IKMEs).

They propose an engineering approach to the development of IKMEs for conceptual modeling and analysis which explicitly deals with both representation levels at the same time. This gives rise to a multilevel approach which is more suitable from an engineering perspective. It allows the domain-dependent description of concepts in terms of different syntactical constructors (e.g. meta-concepts), each endowed with its own set of specialized inferences. These user oriented concept descriptions are transformed into a corresponding set of logic-oriented ones, as well as domain-specific inferences are mapped into general-purpose ones.
The benefit of their approach lies in their decoupling of the conceptual level from the underlying epistemological level, thus allowing for a more open architecture of support tools for the construction and analysis of large domain models.

6.2.4 Dialogue Managers

In their ongoing work [Ludwig et al '98], the authors suggest abstracting dialogue managers from their domain, in order to implement a dialogue manager development tool which takes (among other data) a domain description as input and delivers a new dialogue manager for the described domain as output. It thereby concentrates on two areas: firstly, the construction of domain descriptions with description logic and secondly, the interpretation of utterances in a given domain.

In their work they study the effect of natural language expressions on the state of discourse; they discuss the use of description logics for defining dialogue domain formally and describe how inference is performed on the basis of such a framework.

Their work depends on the following two essential ingredients of any semiotic system (in particular, any language):

- Expressiveness: what are the vocabulary, phonetics and syntax of the language considered?
- Content: what knowledge can be expressed by a given system? How (i.e. by which expressions) is this knowledge organized in the semiotic system?

Further, to deal with the key problem of how to connect content and expressions in order to capture the meaning of expressions, they make use of Russell's proposal to divide the vocabulary into two classes:

- Object Words: define basic notions (in a train information scenario e.g. train, time or station)
- Dictionary Words: can be defined using other words with already defined meaning (e.g. departure time or arrival station)

This leads them to an obvious analogy between object words and primitive concepts in description logics, dictionary words and derived concepts. They exploit this analogy to describe the application domain for the dialogue manager by means of a terminology in DL. By doing this, they could define the meaning of basic and derived notions that are relevant in the domain under consideration.

6.2.5 Natural Language Processing

In [Rudolf et al. '98], the authors main aim is to provide a natural language interface for a database. This interface for database is quite different from general natural language translation processing. Here, the domain is limited and defined in an entity/relationship model. The vocabulary is predictable in a large proportion and queries often have the same construction. However the customization of a natural language interface for a new database is not straightforward because the connection between a lexical base and the conceptual model depends on the naming convention in the model. Another issue concerns query construction.

Thus to deal with the main challenge, in other words, to provide easy portability and fast customization for a new database in a natural language interface for the database, they worked on the separation of the syntactic analysis and the semantic one. They designed a simple syntactic analyzer that could be plugged into a database model with the minimum effort. They used a consistency test and classification capabilities of description logics to solve ambiguities and semantic shortcuts. For this purpose they studied the use of limited regular expressions in Tableaux Calculus.

6.2.6 Tutoring Systems

Present day intelligent tutoring systems can be divided into two types: (1) Systems designed for *training* certain abilities e.g. debugging electronic circuits, solving algebraic equations or interpreting ground tracks. (2) Explanation Systems: designed to teach the fundamental principles (as taught in university courses), which may not have necessarily have an immediate application but which can be trained. Here, the knowledge mainly consists of the *explanation* of concepts and their interrelations, the presentation of examples, definition and the refinement of categories.

A typical difference between these two kinds of systems is the tutoring strategy: „teaching" in explanation system versus „monitoring" in training systems. Thus, every intelligent tutoring

system needs at least two kinds of knowledge, which can be clearly separated. One kind is the domain knowledge has to be taught to the user; the other kind is tutoring knowledge which tells the system how to teach. The latter kind is only used by the system, it need not be explained to the user. Hence it may be represented implicitly, such as in the form of rules [Teege '98]. Thus the main application of Description Logics in intelligent tutoring systems is their use for representing the (declarative) domain knowledge in explanation systems.

The applications of DLs in intelligent systems as proposed by Teege [Teege '98] made this important observation that the main operation needed are description comparison and description decomposition. The subsumption relation in DLs is well suited for the kind of comparisons needed in IT systems. The author proposes a new subtracting operation [Teege '94] which can be used for description decompositions.

Two other applications of DLs in intelligent systems, according to Teege could be using description logics which also support the representation of information about the individuals in an assertional part. In an explanation system individuals are of major importance for giving examples. The second aspect is the association of information with concepts. In the domain knowledge of an explanation system concepts will have lots of associated information, such as a definition, typical usage, relevance and others. The intelligent tutoring system accesses and retrieves this information via the concept. Hence, this situation corresponds to the use of description logics as query languages [Teege '98] (see also [Lenzirini & Schaerf '91a] & [Lenzirini & Schaerf '91b]).

6.2.7 Discussion

The main aim in presenting a wide variety of case studies showing a large spectrum of applications of description logics in various domains was to solidify our argument that DLs have left the stage of laboratory systems and finding their way into solving real world applications.

6.3 Related work in application domain (especially automobile sales advisory systems):

One of the critical elements of the manufacturer's need to make the transition from mass production to mass customization is a configurator. In sales, configuration is typically a high-level configuration (see section 6.1.3) in which an external problem solver — usually salesperson or customer — interacts with the configurator to make the creative decision. This is what differentiates our work from the others. In this section we briefly mention two systems, which exemplify the type of systems built in the domain of sales advisory. The first one, **salesPlus** is a product configuration tool based upon constraint techniques; the second, **Prose** is a knowledge based configurator that supports sales, engineering and manufacturing at AT&T network systems and makes use of description logic-based technology.

6.3.1 salesPlus (Beologic/Baan Denmark):

is a tool which exploits the constraint satisfaction algorithms to evolve a new methodology for interactive configuration. It provides strong features for the end users; free order of choices, guaranteed fast response times and optimization functions. The tool is currently being used for configurations of complex products like Cars, PABX, financial services etc.
The approach used in salesPlus is constraint-based in an effective way i.e. fewer and intuitive constraints are needed. The inference engine performs an effective problem solving mechanism to maintain configuration consistency and to ensure the correctness of

configuration solutions throughout the entire configuration process. In their methodology the fundamental part of the configuration data (product parts & features etc.) is represented as objects & resources, where a resource is a property or measure of some objects e.g. price. salesPlus has a consistency checker that can detect inconsistencies in individual constraints, bound objects and most importantly inconsistency in the overall configuration data. It handles both Boolean algebra and finite domain arithmetic constraints based upon interval calculus and the patented Beologic array inference technology.

SalesPlus product configurator consists of the following modules:

- ***Definer*** is a modeling tool for generating a product model in which relationships between elements and constraints are declared
- ***Customizer*** is a configuration tool for carrying out a configuration process based on a product model that is created through the *Definer*. It performs an interactive product configuration process to satisfy specific customer needs and requirements
- ***salesPlus API*** - The API provides a link between application interface modules, such as Definer and Customizer and the Kernel. Using the API, a new user interface can easily be built, with the desired additional features.
- ***The Kernel*** - The salesPlus kernel holds an internal representation of product models defined and compiled through Definer, as well as the „heart“ of the system -inference engine.

Car Configuration: The following example shows how the products (Saab Automobiles) are modeled within the salesPlus representation.

First the **objects** are declared.

- The 3 models are SAAB 900, Cabriolet and Turbo Cabriolet. They are declared as an object „Models“ of the type ONEOF.
- Available accessories are automatic gearbox, ABS-brakes, Air Bag etc. These are declared as objects of the type SINGLE.
- Available metallic paints are Citrin Beige, Platana Grey etc. The paints are declared as an object „Stdpaint“ of the type ATMOSTONE.
- The objects that do not relate to physical items are leather and delivery time. Leather is declared as an object of the type SINGLE. Delivery time is declared as one object of the Type ENUM[0-35].

Declared resources are Price, Weight and Horse Power. Declared Menus are accessories, Trims, Paints, Accessories at Dealer.

Constraints

SAAB configuration guide lines:
*A It is impossible to have both Air-conditioning and automatic Air-conditioning.
*B The Turbo Cabriolet comes with Turbo engine, metallic paint, leather trim and cruise control.
*C Ordinary Cabriolets comes with 2.1 liter engine.
.
.

*I Delivery times are 14 days for the SAAB 900, 21 days for the Cabriolet and 35 days for the Turbo Cabriolet.
These configuration guide lines yield the following knowledge base.
NEW: Leather <> Trim(leather_contour] or Trim[Leather_suede];
„Sub grouping of leather trim types“
A: Impossible(AirCondition, Automatic_AirCondition];
B: Model[Turbo_Cabriolet] → Engine[Turbo] and
Metalpaint and Leather and cruise_control;
.
.
.
I: Delivery_time = = 14*Model[SAAB_900] + 21 * Model[Cabriolet]
+ 35 * Model[Turbo_Cabriolet];

The inference engine uses a compiled version of the constraints for efficient execution. Easy maintenance in salesPLUS is achieved by keeping the number of constraints small. A single constraint often replaces many production rules and a company's products should be split into several knowledge bases. It is therefore possible for even quite complicated products to be represented by less than 100 constraints.

6.3.2 PROSE (PRoduct OfferingS Expertise, AT&T)

is a knowledge-based configurator platform for telecommunications products. It uses description logics as its knowledge representation scheme. The prose knowledge base is in a purely declarative form that provides developers with the ability to add knowledge quickly and consistently. Apart from generating configurations from just a few high level parameters, PROSE can also verify configurations produced by customers, engineers or sales people.

Due to following reasons description logics was chosen as a basis for PROSE to address the configuration problem:

- object oriented-modeling;
- rule-representation, organization and triggering;
- active inference and knowledge completion;
- explanation, product training and help-desk support;
- ability to handle incrementally evolving specifications;
- extensible schemas;
- reasoning mechanisms that handle incomplete or ambiguous information;
- inconsistency detection, error handling and retraction;
- modularity.

The PROSE configurator application was developed to deal with following three critical problems: (1) the acquisition of product knowledge, (2) rapid and sometimes unexpected changes in product knowledge, and (3) complexity of software enhancements and maintenance.

Apart from that PROSE has three interfaces (pricing, engineering & customer service) that support different aspects of sales, engineering and manufacturing and thus support the three distinct user communities by drawing upon the same product knowledge base (i.e. materials list).

"Central to the PROSE application, no matter what aspect is being discussed, is the concept of a materials list. A material list is a description of the material needed to assemble and install a configuration. It is used to produce a bill of materials for the shop floor, needed for billing and shipping, important for generating instructions to installers, and the basis for communicating with customers about the product. In essence, the material list serves as a manufacturing specification, telling the factory what to assemble."

[Wright et al. 1993]

6.3.3 Discussion:

Both the systems presented in this section deal with the same problem of configuration in two different domains using two different technologies but central to these application is the "bill of materials". Thus both these application fall under the category of "low level configuration" as compared to our application domain of "sales advisory" which falls under the category of „high level configuration". One more way in which our work differs from the two is that it assists the sales person in decision making and it supports the actual (creative) process of advisory by giving hints and explanations to the sales person.

7 Conclusion & Future Directions

In this final section, we draw conclusions from the discussion in previous chapter and present an analytical evaluation of presented sales advisory system and discuss the implications of the results in the context of KBCS, including its sources of power, scope of applicability and major assumptions illustrated with examples. Finally, here we propose some research directions which may later help improve the KR- and Consultation systems.

7.1 Conclusion

The goal of this research has been to establish the relationship between the principles of Knowledge Representation and Reasoning Systems and their embodiment in working application systems. This has been achieved by undertaking a case study in the complex field of automobile sales advisory. Here we investigated how and why a specific KR system is best suitable for an existing specific real-world complex application. This was accomplished by modeling the domain in BACK/FLEX/CLASSIC (see domain modeling). The selection of a deductive and descriptive knowledge representation system helped us achieve the other aspects of the case study such as, the easy maintenance of a knowledge base (KB), efficient knowledge acquisition, the consistency of the KB. Additional aspects such as uniformity, comprehensiveness (expressiveness), flexibility (of data access), stability and reusability(changeability), were also achieved through the following capabilities of the KR systems: object centered, terminological and deductive, incremental and knowledge retraction and rules & procedural tests.

7.1.1 Benefits of our Approach

Our system accrues several key advantages by using a frame system like BACK, which incorporates classification as its foundation. These include the following:

- *Aggregation of Information about Individuals:* the object-centered approach of a frame language allows all information about an individual to be concentrated in one place. This allows us to integrate information obtained from different sources and different points of view into one description.

- *Semantic Retrieval:* classification helps during query processing by reducing the amount of information the user requires about the knowledge base. In purely syntactic information systems, where the terms used have no descriptions (e.g. relational databases or keyword systems), the user has to know the exact terms that are used in the database. Classification reduces the number of terms that the user needs to know. For example, when searching for a general category of items (e.g. Truck), all subcategories (as expressed in the generalization hierarchy induced by classification of frames; e.g., Truck-Bm673) can also be retrieved.

- *Use of Classification and Inheritance to Support Knowledge Acquisition and Consistency:* When new information is acquired and added to the KB, the system automatically classifies the new items, thereby giving the developer an integrity check. Since all categories implied by the new description are found, the developer can see if there were any omissions or accidental inferences made. Inheritance also makes the

addition of new information easier, since many of the properties of a new item can be derived from its membership in existing classes.

- *Use of the KB as an Index:* In a way, the frame KB acts as a general schema over the particular code data stored in the system. The general knowledge encoded in the frames can be used to guide the user in his/her search for particular items of interest. BACK has several means of using the frame KB in this way, including a graphical display of the frame hierarchy.

In short, the simplicity of the BACK language, and the built-in classifier simplifies the knowledge-engineer's task while continuing to provide semantic retrieval as well as a fairly rich knowledge structure for browsing, navigation, and query reformulation.

7.1.2 Results for KR-community

SAS-BACK's essential limitations are of two principle categories either 1) caused by the limitations of BACK or 2) caused by the nature of the domain to be modeled i.e. a specific piece of knowledge that cannot be principally represented in BACK or any other representational scheme based on KL-One paradigm.

The first category of limitations can further be divided into two subclasses as follows: The first class of limitations is rooted in the fact that BACK is a domain-independent language, not specifically designed to represent knowledge about a configuration system in terms of objects and actions. So there are various aspects of each that cannot be expressed adequately within its representational framework. With respect to OBJECTs it would be useful to represent consists-of/part-of hierarchies. For example, head-lights, fog-lights etc. are a part-of lighting-system and which in turn are a part-of electrical-system. BACK does not support reasoning based on meronymic(part-of) hierarchies.

The second class of limitations is due to certain expressive limitations made in BACK to make the classification algorithm faster and easier to implement. For example, one cannot specify a relationship between the fillers of different slots in a concept. In this section we describe some such limitations:

- Negative Implications:

Consider the following example: With a particular version of a PRODUCT P the specific ACCESSORY Y cannot be ordered. or A Truck of production pattern BM673 and of type Typ809 in version t809010 cannot have a full stereo radio/ cassette recorder manufactured by the company "Becker" and model "Avus Kurier". With an ACCESSORY X (already ordered), another ACCESSORY Y cannot be ordered or a main-switch-for-a-boarding-area-wall (E33) cannot be ordered with batteryslides (E38). Both the above mentioned examples refer to negative implications and our representational scheme BACK does not offer any method to represent these kind of constraints.

- Attach Procedure:

While modeling the domain, we required a mechanism which enables us to represent a piece of knowledge as given below:

A specific PRODUCT P in VERSION V has ACCESSORY X, Y, Z, and the product's price is a sum of basic-price of product-version plus the prices of accessories.

In this case we require a slot Has-Price in concept PRODUCT and then attached to this slot a demon or a procedure which computes the price of this product by summing up the individual prices of accessories and the basic product price. BACK does not support this kind of arrangement.

- Default Values:

In the above mentioned domain we are required to work with defaults, for example, a particular product version has some accessories factory fitted but a customer has the option to customize them. Thus a product of type T673001 has a motor of type M364907 but a customer can explicitly requests a motor of say, for example type M364905. BACK does not have any facility to handle such situations.

- Class Attributes:

By this we mean if we have a concept, say a product of type T, which has an attribute has-component whose value is X, then what is required is that every instance of concept T should automatically have the value X for the slot/attribute has-component.

7.1.3 Results for Sales Advisory (Consultation) Systems

Following benefits are expected in deploying such sales advisory (consultation) systems:

- A reduction in operating cost due to the *elimination of errors* on orders detected by sales person and order rework that is carried out to clear these errors.
- A reduction in operating costs through the *consolidation of data bases* and positions.
- A *decrease in the interval for updating product design changes* into the order process infrastructure by eliminating manual interpretation.
- A *decrease in the order process interval by* allowing a customer to configure, edit and send the order to the factory (manufacturing unit) interactively. This compares favorably with the current process which can take a couple of weeks. For some highly complex products, the order interval may be even longer.

7.2 Future Directions

Finally, here we propose some research directions which may later on help improve the KR- and sales advisory systems as well:

7.2.1 Intelligent Access to the World Wide Web

In toady's world of internationalization and globalization the industry of tomorrow is going to be distributed all over the whole world. Thus, the required and necessary information will be available at different locations. The approaches to knowledge based systems to access information on the World Wide Web, with particular emphasis on the use of DL as a modeling language, would be necessary.[IN97]

7.2.2 Information Integration

Not only the location but also the multiple sources of the information pose a problem. Accessing, relating and combining data from multiple sources is an interesting area of research. Critical factors for the design and maintenance of applications requiring information integration are conceptual modeling of domains, and reasoning support over the conceptual representations. [CDGetal97b][CDGetal98b].

7.2.3 Integration with Multimedia Sources

It is not enough to manage the textual information in present day consultation systems but it has become of utmost importance that they be able to manage non-textual sources of information such as raster and vector images, documents, sounds, movies, animation etc.

7.2.4 Application of Terminological Logics to Case-based reasoning

Reasoning from second principles has emerged as a new research paradigm in problem solving. Instead of searching for a solution by reasoning from scratch, this method bases the entire problem-solving process on the reuse and modification of previous solutions (Case Base Reasoning). Our domain of automobile sales advisory is a highly suitable application for such type of reasoning. A key problem in case-based reasoning is the representation, organization and maintenance of case libraries. While current approaches rely on heuristics and psychologically inspired formalisms, terminological logics have emerged as a powerful representation formalism with clearly defined formal semantics [Koehler 94].

Appendices

Appendix 1

A. Car-Select (Fictive Name): **A Sales Advisory System from a German automobile manufacturer.**

1. Task, Domain & Philosophy: This system is designed to help the sales person of a very famous automobile manufacturer. Here the sales person sits with the customer and make the customer select a suitable car with the help of this system. The system constitutes various functionalities such as providing the information in different media for instance text, pictures and video (in a limited way) and system interactions through mouse clicks on picture items or pull-down menu items. Furthermore, this system contains a trivial customer requirements analysis function, which on the basis of customers requirements (here represented as 9 fixed attributes) suggests a particular model of the car from the current product palette of the manufacturer. The basic idea/philosophy behind the system is to provide a fully automated aid for the sales person with user friendly interface due to the very interactive nature of the task.

2. Knowledge Representation: As the system is developed with the help of a database called "Superbase", thus knowledge is strictly represented as database records where information regarding all the available products i.e. all the models of the cars and their respective accessories, is stored in the form of text, visuals and a short video clip.. There is no information regarding conflict management as system provides a choice of those accessories which do not cause any conflict.

3. Inference Mechanism: This system does not contain any kind of inference mechanism in strict terms of Artificial Intelligence. The problem solving here is done via the simple database queries sent to the system. Which are retrieved and displayed for the user in his/her required form i.e. either text, visual or video.

4. Interactions/User interface: The following two screen shots might provide the user with some insights into the interactions and the user interface of this particular system. The first picture shows the section of the customer requirements analysis where the user is asked to specify his/her needs/wishes in terms of nine fixed attributes represented on the screen in form of small pictures in reverse video. As soon as the user selects one, it changes to normal mode and a larger picture is placed like a tile of puzzle in the window on the left hand side. This process continues till the customer has provided the system with all his/her needs/wishes. Then the system suggests a particular model of the car whereupon the user can select the accessories which he/she requires.

5. Knowledge Acquisition: Knowledge acquisition is manual, that is, all kind of changes in or additions of new knowledge must be hand coded into the system. This causes a lot of trouble in a domain where on average 30-40% of product knowledge is liable to change every year.

6. Knowledge Consistency/Maintenance: There is no such facility available in this system.

B. Auto-Prog(Fictive Name): **An automobile Sales Advisory System from another German car manufacturer.**

1. Task & Domain: This system was also developed to help the customer of a famous Bavarian car maker to select for himself/herself a specific model of car and the accessories available. This system also makes use of database technology and does not have any information regarding the product and accessories in picture or video form. It uses a very trivial way to interact with the user; the user is presented with different information in different types of lists and is asked to select one according to his/her needs/wishes. The User Interface is also simple, lots of buttons each representing a different function and text display window to display textual information regarding the product or accessories.

2. Knowledge Representation: All the information is stored in forms of lists, be it categories of cars available or specific models of a particular series. The accessories of each model are stored separately as are the constraints in the form of rules.

3. Inference Mechanism: Similar to the system **(Car-Select)** as mentioned in Appendix 1, this system also does not contain any kind of inference mechanism in strict terms of Artificial Intelligence. The problem solving here is done via presenting the user with lists containing different information and the user selects according to his/her needs/wishes. If during this process some kind of conflict occurs among two selected accessories, then the users is informed of this by a simple message stating which two particular accessories do not go with each other and user has to take back (deselect) one of the two.

4. User Interface: the user's interactions are acomplished through the user interface which consits of various buttons and an empty text display window. The user starts by pressing one of the 10 buttons. The user is first asked to select one of the series(categories) offered, then the different models of this particular series and finally the accessories. All these choices appear in the text display window where the user can make his/her selections. At the time of selecting accessories, easy conflicts are resolved by the system as it automatically deselects an already selected accessory, but in case of difficult conflicts the system asks for the user's help.

5. Knowledge Acquisition: Similar to the system **car-select**, Knowledge acquisition here is also manual; that means all kind of changes or addition of new knowledge must be hand coded into the system.

6. Knowledge Consistency/Maintenance: No such facility is available for this system.

C: In this section, we present three systems which are presently in use at different locations (2 in Germany and 1 in the USA) of the same automobile manufacturer but for different products (2 for Trucks and 1 for Buses). All these systems were developed by different departments and with different goals and philosophies. These systems may have lots of differences but two things they have in common, first the task (sales consultation) and the development platform (Personal Computer).

i) WinHTV (Fictive Name) A Sales Advisory System from yet another automobile manufacturer based in Germany:

1. Task, Domain & Philosophy: This system was evolved after a series of prior versions and is used by the sales force of a leading truck manufacturing company during the sales consultation. This system is the first of it's kind to be developed on a PC based upon MS-Windows. This gives the system a fairly user friendly interface. This system contains information regarding the available trucks and accessories along with their prices in textual form and some information is stored as pictures as well. Due to the technical complexity of the product -which thus leads to the complexity of the task- this system is provided with a wide range of functions; these are then grouped together and can be accessed by the user from the menu bar as menu items.

2. Knowledge Representation: Knowledge Representation in this system is identical to the technical manuals previously used by sales people. Here Information is separated into two groups, namely the product (different models of trucks) and the accessories available for these trucks along with their suitable manufacturability constraints. This information is stored in the form of lists and the user can operate upon them as a "pick and choose" principle. To explain certain complex terms, this system contains information in the form of visuals as well.

3. Inference Mechanism: Similar to both the above mentioned systems **(Car-Select & Auto-Prog)**, this system also does not contain any kind of inference mechanism in strict terms of Artificial Intelligence. The problem solving here is also done via presenting the user with lists containing different information which the user selects according to his/her needs/wishes. Here the only difference to the other systems is that the information is provided in a formatted window with a range of options to either retract a previously selected item or to have an extra piece of information regarding that particular item. If during this process some kind of conflict occurs among two selected accessories then the user is informed of this in a message window stating which two particular accessories do not go with each other and user has to take back (deselect) one of the two.

4. Interactions/User interface: For the uniformity purposes, we have again provided two snapshots of the system WinHTV so that the reader can have a better picture of the systems user interaction capabilities. Figure 1 shows the general screen layout with an open menu bar which shows what functions does the system offers and an icon bar shows some extra functionalities plus an access to certain existing facilities e.g. during the configuration if one has to select some accessories then one can click on the "book" icon, similarly "spray can" icon allows to select colors. Figure 2 shows how in one window the user can keep on selecting the appropriate new accessories and in an another window the user can keep track of all the accessories selected so far.

5. Knowledge Acquisition: Similar to both the systems **Car-Select & Auto-Prog**, Knowledge acquisition here is also manual, that means all kind of changes or the addition of new knowledge must be hand coded into the system.

6. Knowledge Consistency/Maintenance: No such facility is available for this system.

ii) Trucker (fictive Name): A sales advisory system from a US-Based Truck Manufacturer. This system is almost a duplicate copy of WinHTV the only difference this system has is that it makes use of picture material extensively and does not have a Windows based user interface. The knowledge representation, acquisition and consistency/maintenance and inference mechanism is fully identical. The only difference lies perhaps in the user interactions.

iii) Bus-Consultant (Fictive Name): A sales advisory system from a Bus manufacturer:

This system belongs to the X-Consultant family of the systems developed by the research organization of a famous automobile manufacturer. These systems follow a completely different philosophy, in which the developers claim to have considered the social environment and the end user in their design. One such system "Car-Consultant" is described at length in chapter 2.3. Therefore I will not go into details but simply mention that this system helps the sales person to configure a "bus". Furthermore, to maintain the uniformity of the appendices, I have included two screenshots of this particular system as well.

Appendix 2

Note: The source of following grammer/syntax is the respective users manual for each system.

The CLASSIC Grammar

The following defines the syntax for typing in a CLASSIC concept or individual description:

<descr> ::=	`THING` | <**CLASSIC**-descr> | <**HOST**-descr>
<**HOST**-descr> ::=	`HOST-THING` | <**HOST**-concept-name> |
	(**AND** <**HOST**-descr>$^+$) |
	(**ONE-OF** <**HOST**-individual>*) |
	(**TEST-H** <fn> <arg>*) |
	(**MIN** <number>) |
	(**MAX** <number>)
<**CLASSIC**-descr> ::=	`CLASSIC-THING` | <**CLASSIC**-concept-name> |
	(**AND** <**CLASSIC**-descr>$^+$) |
	(**ONE-OF** <**CLASSIC**-individual-name>*) |
	(**TEST-C** <fn> <arg>*) |
	(**ALL** <role-name> <descr>) |
	(**AT-LEAST** <positive-integer> <role-name>) |
	(**AT-MOST** <non-negative-integer> <role-name>) |
	(**FILLS** <role-name> <individual-name>$^+$)
	(**SAME-AS** (<attribute-name>$^+$) (<attribute-name>$^+$))
<**HOST**-concept-name> ::=	<symbol> *;; defined as a host concept*
<**CLASSIC**-concept-name> ::=	<symbol> *;; defined as a classic concept*
<role-name> ::=	<mrole-name> | <attribute-name>
<mrole-name> ::=	<symbol> *;; defined as a role, but not functional*
<attribute-name> ::=	<symbol> *;; defined as a functional role*
<individual-name> ::=	<**HOST**-individual> | <**CLASSIC**-individual-name>
<**HOST**-individual> ::=	<string> | <number> |
	'<CommonLisp-expr> | (quote <CommonLisp-expr>)
<**CLASSIC**-individual-name> ::=	<symbol>
<fn> ::=	a function in the **HOST** language (Common LISP) that takes an individual, plus the extra arguments, and returns `T`, `?`, or `NIL`.
<arg> ::=	an expression passed to the test functions.

Alternative Keyword

= may be used instead of the **SAME-AS** operator

BACK Syntax

Syntax Overview

In the syntax overview we use the following conventions of the extended Backus-Naur Form (EBNF):

- optional arguments are put into brackets, e.g., [optional];
- when brackets are intended as terminal symbols, they are quoted, e.g., '['list']';
- iteration is indicated by braces, e.g., {,element}*;
- parentheses are always terminal symbols, e.g. **backinit**(tbox);

Some constructs in the syntax overview are only applicable in a restricted way (as described above). We mark this by the following signs:

π argument must be primitive;

τ operator may only be used in TBox tells and queries, i.e. not in terms used to describe objects;

? operator may only be used in ABox queries, i.e. in object ?: concept or in the **getall** part of retrievals;

α operator may only be used in ABox expressions, i.e. in terms used to describe objects;

! operator may only be used in tells;

$\top$ operator may only be used at the top-most level.

Interaction

⟨*interaction*⟩ ::= **backinit**[(⟨*box*⟩)]
| **backtell**(⟨*tell-expression*⟩)
| **backask**(⟨*ask-expression*⟩[**noibox**])
| **backstate**[(⟨*state*⟩)]
| **backretrieve**(⟨*retrieval*⟩[**noibox**])
| **backmacro**(⟨*macro-definition*⟩)
| **backread**(⟨*file*-NAME⟩)
| **backload**(⟨*file*-NAME⟩)
| **backwrite**(⟨*file*-NAME⟩[,⟨*box*⟩])
| **backdump**[(⟨*file*-NAME⟩)]

⟨*retrieval*⟩ ::= [PROLOG-VAR =] [⟨*generator*⟩] ⟨*arguments*⟩
| **difference**(⟨*entity*⟩,⟨*entity*⟩)

⟨*generator*⟩ ::= ⟨*action*⟩
| '[⟨*output-function*⟩{,⟨*output-function*⟩}*']' **for**

⟨*action*⟩ ::= **describe**
| **describe_fully**
| **defined_as**
| **introduced_as**
| **self**
| **msc**

⟨*output-function*⟩ ::= ⟨*action*⟩
| **vr**(⟨*role*-NAME⟩)
| **vr**(**inv**(⟨*role*⟩))
| **nr**(⟨*role*-NAME⟩)
| **nr**(**inv**(⟨*role*⟩))
| **rf**(⟨*role*-NAME⟩)
| **rf**(**inv**(⟨*role*⟩))

⟨*arguments*⟩ ::= ⟨*arg-spec*⟩[/⟨*disambig*⟩]
| **getall**(⟨*concept*⟩)
| **getall**(⟨*aset*⟩)
| **getall**(**string**)

⟨*arg-spec*⟩ ::= ⟨*entity*⟩
| '['⟨*entity*⟩[/⟨*disambig*⟩] {,⟨*entity*⟩[/⟨*disambig*⟩]}*']'

⟨*disambig*⟩ ::= **conc**
| **obj**
| ⟨*domain*-NAME⟩^**cls**
| ⟨*domain*-NAME⟩^**obj**

⟨*macro-definition*⟩ ::= ⟨*macro*⟩ *= ⟨*term*⟩
⟨*macro*⟩ ::= ⟨*macro*-NAME⟩[(PROLOG-VAR{,PROLOG-VAR}*)]

⟨*state*⟩ ::= **verbosity = silent**
| **verbosity = error**
| **verbosity = warning**
| **verbosity = info**
| **verbosity = trace**
| **introduction = forward**
| **introduction = noforward**
| **revision = true**
| **revision = false**
| **retrieval = fail**
| **retrieval = succeed**
| **tboxrevision = fail**
| **tboxrevision = succeed**
| **aboxfilled = false**
| **aboxfilled = true**
| **aboxfilled = abox**
| **iboxfilled = false**
| **iboxfilled = true**

⟨*box*⟩ ::= **tbox**
| **ibox**
| **abox**

Tell/Ask Expressions

⟨*tell-expression*⟩ ::= ⟨*definition*⟩[**type** ⟨*modifier*⟩]
| ⟨*rule*⟩
| ⟨*description*⟩
| ⟨*revision*⟩
| ⟨*declaration*⟩

⟨*definition*⟩ ::= ⟨*term*-NAME⟩ := ⟨*term*⟩
| ⟨*concept*-NAME⟩ :< ⟨*concept*⟩
| ⟨*role*-NAME⟩ :< ⟨*role*⟩

⟨*rule*⟩ ::= ⟨*concept*⟩ => ⟨*concept*⟩

⟨*description*⟩ ::= ⟨*obj-ref*⟩ :: ⟨*concept*⟩
| PROLOG-VAR :: ⟨*concept*⟩

⟨*revision*⟩ ::= **forget**(⟨*rule*⟩)
| **forget**(⟨*obj-ref*⟩ :: ⟨*concept*⟩)
| **forget**(⟨*obj-ref*⟩)
| **redescribe**(⟨*obj-ref*⟩ :: ⟨*concept*⟩)
| **name**(⟨*obj-ref*⟩,⟨*object*-NAME⟩)

⟨*declaration*⟩ ::= ⟨*domain*-NAME⟩ := **attribute_domain**
| ⟨*domain*-NAME⟩ := **attribute_domain**(⟨*attribute-list*⟩)

⟨*modifier*⟩ ::= **concept**
| **role**
| **feature**
| **aset**
| **number**
| **string**

⟨*ask-expression*⟩ ::= ⟨*term*⟩ ?< ⟨*term*⟩
| ⟨*obj-ref*⟩ ?: ⟨*concept*⟩
| **disjoint**(⟨*term*⟩,⟨*term*⟩)
| **subsumes**(⟨*term*⟩ ⟨*term*⟩)
| **equivalent**(⟨*term*⟩,⟨*term*⟩)
| **incoherent**(⟨*term*⟩)

Terms

⟨*entity*⟩ ::= ⟨*term*⟩
| ⟨*value*⟩

⟨*term*⟩ ::= ⟨*conceptual-type*⟩
| ⟨*role*⟩
| ⟨*macro*⟩

⟨*conceptual-type*⟩ ::= ⟨*concept*⟩
| ⟨*aset*⟩
| ⟨*number*⟩
| **string**

⟨*value*⟩ ::= ⟨*obj-ref*⟩
| ⟨*attribute*-NAME⟩
| ⟨*number*-INSTANCE⟩
| ⟨*string*-INSTANCE⟩

⟨*obj-ref*⟩ ::= ⟨*object*-NAME⟩
| **uc**(⟨INTEGER⟩)$^{\alpha}$
| **theknown**(concept)$^{\alpha}$

Concept Terms

⟨*concept*⟩ ::= ⟨*concept*-NAME⟩
| **anything**
| **nothing**
| ⟨*concept*⟩ **and** ⟨*concept*⟩
| ⟨*concept*⟩ **or** ⟨*concept*⟩$^{?\top}$
| **not**(⟨*concept*⟩)$^{\pi}$
| **all**(⟨*role*⟩,⟨*conceptual-type*⟩)
| **atleast**(⟨INTEGER⟩,⟨*role*⟩)
| **atmost**(⟨INTEGER⟩,⟨*role*⟩)
| **oneof**('['⟨*object*-NAME⟩{,⟨*object*-NAME⟩}*']')
| ⟨*role*⟩:⟨*filler-expr*⟩

Role Terms

⟨*role*⟩ ::= ⟨*role*-NAME⟩
| ⟨*role*⟩ **and** ⟨*role*⟩
| **not**(⟨*role*⟩)$^{\pi}$
| **domain**(⟨*concept*⟩)
| **range**(⟨*conceptual-type*⟩)
| **inv**(⟨*role*⟩)
| ⟨*role*⟩ **comp** ⟨*role*⟩$^{\tau}$
| **trans**(⟨*role*⟩)$^{\tau}$

Attribute Set Terms

⟨*aset*⟩ ::= **aset**
| ⟨*aset*-NAME⟩
| ⟨*aset*⟩ **union** ⟨*aset*⟩
| (⟨*aset*⟩ **intersection** ⟨*aset*⟩)
| ⟨*aset*⟩ **without** ⟨*aset*⟩
| **aset**('['⟨*attribute-list*⟩']')
| **aset**(⟨*attribute-spec*⟩,⟨*domain*-NAME⟩)

⟨*attribute-spec*⟩ ::= '['⟨*attribute-list*⟩']'
| ⟨*attribute*-NAME⟩ .. ⟨*attribute*-NAME⟩

⟨*attribute-list*⟩ ::= ⟨*attribute*-NAME⟩{,⟨*attribute*-NAME⟩}*

Number Terms

⟨*number*⟩ ::= **number**
| ⟨*number*-NAME⟩
| ⟨*number-range*⟩
| (⟨*number*⟩ **intersection** ⟨*number*⟩)
| ⟨*number*-INSTANCE⟩

⟨*number-range*⟩ ::= ⟨*lower-limit*⟩
| ⟨*upper-limit*⟩
| ⟨*number*-INSTANCE⟩.. ⟨*number*-INSTANCE⟩

⟨*lower-limit*⟩ ::= **gt**(⟨*number*-INSTANCE⟩)
| **ge**(⟨*number*-INSTANCE⟩)

⟨*upper-limit*⟩ ::= **lt**(⟨*number*-INSTANCE⟩)
| **le**(⟨*number*-INSTANCE⟩)

Filler Expressions

⟨*filler-expr*⟩ ::= ⟨*value*⟩
| (⟨*description*⟩)$^{!\alpha}$
| **close**(⟨*filler-expr*⟩)
| **someknown**(⟨*concept*⟩)$^{?\alpha}$
| **allknown**(⟨*concept*⟩)$^{\alpha}$
| ⟨*value*⟩ **and** ⟨*filler-expr*⟩
| ⟨*value*⟩ **or** ⟨*filler-expr*⟩$^{?\alpha}$
| (⟨*filler-expr*⟩)

Macro Library

⟨*macro-concept*⟩ ::= **some**(⟨*role*⟩,⟨*conceptual-type*⟩)
| **some**(⟨*role*⟩)
| **the**(⟨*role*⟩,⟨*conceptual-type*⟩)
| **no**(⟨*role*⟩,⟨*conceptual-type*⟩)
| **no**(⟨*role*⟩)
| **exactly**(⟨INTEGER⟩,⟨*role*⟩)
| **atleast**(⟨INTEGER⟩,⟨*role*⟩,⟨*conceptual-type*⟩)
| **atmost**(⟨INTEGER⟩,⟨*role*⟩,⟨*conceptual-type*⟩)
| **exactly**(⟨INTEGER⟩,⟨*role*⟩,⟨*conceptual-type*⟩)
| **rvm_some**(⟨*role*⟩,⟨*role*⟩)
| **rvm_no**(⟨*role*⟩,⟨*role*⟩)

FLEX Syntax

Interaction

⟨*interaction*⟩ ::= **flexinit**
| **flextell**(⟨*tell-expression*⟩)
| **flexask**(⟨*ask-expression*⟩[,box=⟨*box*⟩])
| **flexget**(⟨*entity*⟩,⟨*method*⟩,[⟨*options*⟩,]PROLOG-VAR)
| **flexstate**(⟨*state*⟩)
| **flexread**(⟨*file*-NAME⟩)
| **flexdump**(⟨*file*-NAME⟩[,Comment])
| **flexload**(⟨*file*-NAME⟩)

⟨*state*⟩ ::= **verbosity = silent**
| **verbosity = error**
| **verbosity = warnings**
| **verbosity = info**
| **verbosity = trace**
| **introduction = forward**
| **introduction = noforward**
| **class_objects = on**
| **class_objects = off**
| **defspace_dbox = best**
| **defspace_dbox = all**
| **syntax_check = on**
| **syntax_check = off**

Tell/Ask Expressions

⟨*tell-expression*⟩ ::= ⟨*definition*⟩
| ⟨*rule*⟩
| ⟨*description*⟩
| ⟨*default*⟩
| ⟨*disjointness*⟩
| ⟨*sit-extension*⟩
| ⟨*macro-definition*⟩

⟨*definition*⟩ ::= ⟨*term*-NAME⟩ := ⟨*term*⟩[**with filter=**⟨*filter-list*⟩]
| ⟨*term*-NAME⟩ :< ⟨*term*⟩[**with filter=**⟨*filter-list*⟩]

⟨*rule*⟩ ::= ⟨*concept*⟩ => ⟨*concept*⟩

⟨*default*⟩ ::= ⟨*concept*⟩~⟨INTEGER⟩~> ⟨*concept*⟩

⟨*description*⟩ ::= ⟨*object*-NAME⟩ :: ⟨*concept*⟩[**in** ⟨*sit-ref*⟩]
| PROLOG-VAR :: ⟨*concept*⟩[**in** ⟨*sit-ref*⟩]

⟨*disjointness*⟩ ::= ⟨*concept*-NAME⟩<> ⟨*concept*-NAME⟩
| <> '['⟨*concept*-NAME⟩{,⟨*concept*-NAME⟩}*']'

⟨*sit-extension*⟩ ::= ⟨*sit*-NAME⟩<<= ⟨*sit-ref*⟩

⟨*macro-definition*⟩ ::= ⟨*macro*⟩ *= ⟨*term*⟩
⟨*macro*⟩ ::= ⟨*macro*-NAME⟩[(PROLOG-VAR{,PROLOG-VAR}*)]

⟨*ask-expression*⟩ ::= ⟨*term*⟩ ?< ⟨*term*⟩
| ⟨*object*-NAME⟩ ?: ⟨*concept*⟩[**in** ⟨*sit*-NAME⟩]
| PROLOG-VAR ?: ⟨*concept*⟩[**in** ⟨*sit*-NAME⟩]
| **disjoint**(⟨*term*⟩,⟨*term*⟩)
| **subsumes**(⟨*term*⟩ ⟨*term*⟩)
| **equivalent**(⟨*term*⟩,⟨*term*⟩)
| **incoherent**(⟨*term*⟩)
| **satisifes**(⟨*term*-NAME⟩,⟨*filter-list*⟩)

Terms

⟨*entity*⟩ ::= ⟨*term*⟩
| ⟨*value*⟩

⟨*term*⟩ ::= ⟨*concept*⟩
| ⟨*role*⟩

⟨*value*⟩ ::= ⟨*object*-NAME⟩
| ⟨*number*-INSTANCE⟩
| ⟨*string*-INSTANCE⟩

Concept Terms

⟨*concept*⟩ ::= ⟨*concept*-NAME⟩
| ⟨*number*⟩
| **string**
| ⟨*macro*-NAME⟩
| **ctop** | **anything**
| **cbot** | **nothing**
| **prim**(⟨*concept*-NAME⟩)
| ⟨*concept*⟩ **and** ⟨*concept*⟩
| ⟨*concept*⟩ **or** ⟨*concept*⟩
| **not**(⟨*concept*⟩)
| **all**(⟨*role*⟩,⟨*concept*⟩)
| **the**(⟨*role*⟩,⟨*concept*⟩)
| **some**(⟨*role*⟩[,⟨*concept*⟩])
| **no**(⟨*role*⟩[,⟨*concept*⟩])
| **atleast**(⟨INTEGER⟩,⟨*role*⟩[,⟨*concept*⟩])
| **atmost**(⟨INTEGER⟩,⟨*role*⟩[,⟨*concept*⟩])
| **exactly**(⟨INTEGER⟩,⟨*role*⟩[,⟨*concept*⟩])
| ⟨*role*⟩:⟨*value*⟩ | ⟨*role*⟩:'['⟨*value*⟩{,⟨*value*⟩}*']'
| ⟨*role*⟩:⟨*term*⟩
| **rvm_equal**(⟨*role*⟩,⟨*role*⟩)
| **oneof**('['⟨*object*-NAME⟩{,⟨*object*-NAME⟩}*']')
| **rvm_inst**(⟨*role*⟩,⟨*role*⟩)

Role Terms

⟨*role*⟩ ::= ⟨*role*-NAME⟩
| **rtop** | **anyrole**
| **rbot** | **nothing**
| ⟨*role*⟩ **and** ⟨*role*⟩
| ⟨*role*⟩ **or** ⟨*role*⟩
| **not**(⟨*role*⟩)
| **prim**(⟨*role*-NAME⟩)
| **domain**(⟨*concept*⟩)
| **range**(⟨*concept*⟩)
| **inv**(⟨*role*⟩)
| ⟨*role*⟩ **comp** ⟨*role*⟩
| **feature**
| **term_valued**

Number Terms

⟨*number*⟩ ::= **number**
| ⟨*number*-NAME⟩
| ⟨*number-range*⟩
| ⟨*number*-INSTANCE⟩

⟨*number-range*⟩ ::= ⟨*lower-limit*⟩
| ⟨*upper-limit*⟩
| ⟨*number*-INSTANCE⟩.. ⟨*number*-INSTANCE⟩

⟨*lower-limit*⟩ ::= **gt**(⟨*number*-INSTANCE⟩)
| **ge**(⟨*number*-INSTANCE⟩)

⟨*upper-limit*⟩ ::= **lt**(⟨*number*-INSTANCE⟩)
| **le**(⟨*number*-INSTANCE⟩)

Methods

⟨*method*⟩ ::= **all**(⟨*role*⟩[,⟨*sit*-NAME⟩])
| **atleast**(⟨*role*⟩[,⟨*concept*⟩] [,⟨*sit*-NAME⟩])
| **atmost**(⟨*role*⟩[,⟨*concept*⟩] [,⟨*sit*-NAME⟩])
| **concepts**(⟨*sit*-NAME⟩)
| **dir_subs**
| **dir_supers**
| **domain**
| **equivalents**
| **fillers**(⟨*role*⟩[,⟨*sit*-NAME⟩])
| **filter**
| **help**
| **instances**(⟨*sit*-NAME⟩)
| **msc**(⟨*sit*-NAME⟩)
| **range**
| **subs**
| **supers**
| **tvf_filler**(⟨*role*⟩[,⟨*sit*-NAME⟩])

⟨*options*⟩ ::= **box**=⟨*box*⟩
| **filter**=⟨*filter-list*⟩

⟨*box*⟩ ::= rules
| defaults

⟨*filter-list*⟩ ::= ⟨*filter*-NAME⟩
| '['⟨*filter*-NAME⟩{,⟨*filter*-NAME⟩}*']'

⟨*sit-ref*⟩ ::= ⟨*sit*-NAME⟩
| PROLOG-VAR

Appendix 3

Internet and its Implications
for
Automobile Sales Advisory/Consultation

In recent years the world wide web user population and the web itself have undergone dramatic changes (see research reports by International Data Corp. (IDC) [IDC'00]and the Computer Industry Almanac (CIA) [CIA'00] on internet users). The limited form of interactivity has been superseded by the use of various new technologies (e.g. JAVA). This is making internet more acceptable throughout the society. As the number of internet users - from all walks of life - is increasing every day, it has become essential for the companies to be present on the internet. Further due to improvements in security and electronic payment standards online(electronic/internet) commerce has been established. A constantly growing variety of products and services is being offered to a rising number of "netizens" and making the vision a reality, "where one will be able to access every known product and service by simply clicking a button on the home computer".

Electronic commerce offers a huge variety of application areas nowadays. Much effort is put into research and exploration of these areas and one such area, in conjunction with e-commerce, which is getting increasing notice is sales support. However, the effort in this direction has been concentrated on the product configuration (for a discussion on internet configuration systems, see [Rahmer et al.'99]) and the research to address one of the most important problems, responsive online sales support/sales advisory, has generated only few promising approaches. Most electronic catalogs and online shops do not explore the interactivity available on the web [Timm & Rosewitz'98]. Hence, their searching and indexing capabilities are often only a little more useful than their printed counterparts and thus making the search and selection of complex products (e.g. automobile) on the world wide web an equally difficult task for consumers and business professionals as well. Further, at the moment , unlike normal business situations, there is no intelligent support or assistance for the web user. Current product oriented database search facilities are widely used on the Internet and recognized as limited in capabilities for sales support (Advisory/consultation).

Driven by the challenge to improve internet-based electronic commerce and due to the recent technological advancement, one has come up with certain important possibilities to support the web shoppers in two aspects of the sales process currently neglected on the web; helping the customer to select or configure the product that meets her needs and supporting the customer in navigating through the space of possible product alternatives.

Our approach as described in chapter 5 combined with the latest developments in the field of content defining languages over the internet such as meta-language XML(extensible markup language), which is designed to represent the contextual meaning of the data, would be a step forward to bring improvement in the two above mentioned aspects. Thus making it a language specifically crafted for applications or domain description which require heavy-duty, flexible interoperable data systems (see for a short discussion on XML, appendix 4).

E-Commerce & Automobile Industry
The beginning of the e-commerce for consumers was dominated by homogeneous, lower valued and lower involvement products (e.g. books or CDs), it now extends to more complex,

higher valued and high involvement products such as cars. Practically every automobile company is present on the web in some way or the other and offers its various products and services. Various market research surveys/studies [Duddenhöfer'98, Forrester Research'99, Roland Berger'99sept., Gemni Consulting'99, Anderson'99 & BA&H'99] conducted by major consulting groups in recent past have acknowledged this emerging & ever rising trend of buying cars online.

According to these surveys, automobiles are among the five fastest-growing online products, with an annual growth rate of over 200% [BusinessWire'99, Conhaim'98, Nua Internet Survey'98]. Other major findings of these surveys regarding the online population of car buyers and the volume of revenue generated were:

- in 1998, over 2 million consumers used the internet for making a decision about buying a new car and by 2003 it is forecast that internet will influence over 8 million car purchases and 470000 US households will buy new cars entirely over the internet generating revenue up to US$ 12 billion [BusinessWeek'96, BusinessWire'99, Dudenhöffer'99, ForresterResearch'99].

- The web is likely to influence 50% of all new car purchases within five years, forcing leading manufacturers, dealers and online buying services to distance themselves from traditional distributors by catering more and more to wired shoppers.

Along with these trends and the rise of the internet as a means of distribution in the car market has implications on all participants of the distribution channel: for manufacturers, distributors and consumers [Strauss'99]. In other words , Electronic Commerce is undoubtedly helping to reshape the automotive industry.

Instead of going into details about this new heated up battle on the web, where auto makers and dealers are vying with online-car buying services for the loyalty and dollars of the drivers, I'll only, in short, be discussing one of the key success factor which has a direct impact and relevance to the work being reported here i.e. understanding of computer-mediated seller-buyer interactions or sales consultation/advisory.

At present most of the web or electronic commerce has providing price efficiency to their consumers to the core of their functionalities as compared to the traditional business practices. Currently, the main focus of the websites provided by the automobile manufacturers (list of automobile related websites) is the vehicle configuration with limited options and pricing structure for leasing and buying. Not only that, these sites are more or less static and make use of little or no interactive capabilities as provided by the medium.

In recent past all the automotive product manufacturers are expanding their eCommerce capabilities to support all phases of the product buying cycle. As the competition becomes fiercer, margins cut and the consumers empowered by the web (due to unlimited, unbiased online information), the only way the automobile manufacturers can save their products and services being comodotized is by selling the customer a rich buying experience (guaranteeing simultaneous satisfaction and the physical product experience) through designed seller-buyer interactions. This infact is one of the central theses of the work reported earlier in chapter 4.4

Appendix 4

XML: The Lifeblood of Internet Economy

Every now and then, a new technology profoundly alters the competitive landscape and provides the seeds for radical change. It is now clear that the internet is such a technology. The use of internet for commerce has been on rise in the recent past. Like traditional commerce, internet commerce is also going to be based on documents (e.g. web based product catalogues, invoices etc.). The tremendous growth of world wide web is due to its ability to deliver electronic documents easily and cheaply. As web documents have become larger and more complex, however, web content providers have begun to experience the limitations of a medium that does not provide the extensibility, structure and data checking needed for large-scale commerce activity.

To address the requirements of commercial web and enable the expansion of the web technology into new domains of distributed document processing, the world wide web consortium developed an Extensible Markup Language(XML) for applications (such as our application of sales advisory) that require functionality beyond the current hypertext markup language(HTML).

XML is a markup language that lets documents turn from a static file into a repository of information that can be displayed and worked with in many different ways. Moreover, it is a way of adding intelligence to the documents. It lets identify each element using meaningful tags and add information ("metadata") about each element. In another words XML allows specific markup to be created for specific data and has the virtues of HTML without any of its limitations. XML is strong in:

- intelligence
- adaptation
- maintenance
- linking
- simplicity
- portability.

Thus making XML very much a part of the future of web, and part of the future for all electronic information. In short, „XML has been designed for maximum expressive power, maximum teachability and maximum ease of implementation“ [Bosak'97].

XML is a simplified version of SGML and a cousin of HTML. For further details we refer to a comprehensive repository „The XML Cover Pages“ on the web: http://www.oasis-open.org/cover/

Here, just to give the reader an idea how & what XML looks like, we provide an example, a very simple one, from our domain of automobile sales advisory. Imagine the company has to sell cars on-line. Marketing descriptions of the cars are written in HTML, but names and addresses of customers, prices and discounts are formatted with XML. Then the information describing a customer looks as follows:

```
<customer-details>
  <name>Sunil Thakar</name>
  <address>
```

```
    <street>Wartburg Str. 53a</Street>
    <city>Berlin</city>
    <zip-code>10823</zip-code>
  </address>
</customer-details>
```

XML uses matching start and end tags, such as <name> and </name>, to mark up information. Its simple syntax is easy to process by a machine and has attraction of remaining understandable to humans. **Thus providing a low(machine)-level syntax for representing declarative data.** This property brings it closer to our approach (description logics/declarative knowledge representation) and making it an obvious choice for future sales advisory application over the net.

The following two additional properties of XML i.e. representing metadata {data that describes data or in context of web/publishing separating content (data), structure(metadata) & formatting(metadata)} and semantic information (encoding of information at the semantic level/information regarding meaning) would also make it a language of choice to represent knowledge in future web based business (sales advisory) applications[1]. This would elevate the status of the web from *machine-readable* to some what *machine-understandable*. Not only that, as XML spreads, the web should become noticeably more responsive due to the structural and semantic information which can be added with XML, which allows the computing devices – whether powerful desktop computers or tiny pocket planners –connected to the web to do a great deal of processing on the spot. That not only will take a big load off web servers but also reduce network traffic dramatically[Bosak & Bray'99].

For a detailed discussion about semantic transparency[2] and metadata {including several proposals: Web Collections using XML, Channel Definition Format (CDF), Meta Content Framework using XML(MCF) and Resource Description Framework(RDF)[3], we recommend the interested reader to consult the following entries in the literature [Lander'97], [Cover'98], [Guha & Bray'97], [Lassila'97].....etc. A discussion on the explicit semantics (ontologies) in the context of expanding e-commerce/e-business [CSC-features: http://www.cse.com/features/110199_features.html] can be found at http://www.ontology.org.

To sum this up, the power and beauty of XML is that it maintains the separation of the user interface from structured data, allowing the seamless integration of data from diverse sources.

---

[1] **Note:-** AT&T, in May 1999, announced an XML application that will be able to access product and service information from disparate databases and publish that information in a personalized way [Gaskin'99].

[2] **Note:-** Semantic Transparency: to represent information objects in the problem domain such that they correspond *transparently* ("one-to-one") to the user's conceptual model of objects in this domain [Cover'98]

[3] **Note:-** The development of RDF – the Resource Description Framework – and in a way as general **knowledge representation mechanism** for the web – was heavily inspired by PICS (Platform for Internet Content Selection). RDF is a foundation foe processing metadata; it provides interoperability between applications that exchange machine-understandable information on the web. At the core, RDF data consists of nodes and attached attribute/value pairs. Nodes can be any web resources(pages, servers, basically anything for which one can give a URI (Universal Resource Identifier), even other instances of metadata. Attributes are named properties of the nodes and their values are either atomic(text strings, numbers etc.) or other resources or metadata instances. In short, this mechanism allows one to build labeled directed graphs [Lassila'97].

Not only that, XML is also valuable to the internet as well as large corporate intranet environments because it provides interoperability using a flexible, open, standards-based format, with new ways of accessing legacy databases and delivering data to web clients. Finally XML brings so much power and flexibility to Web-based applications that it provides a number of compelling benefits to developers and users:

- More meaningful searches
- Development of flexible web applications
  - Data integration from disparate sources
  - Data from multiple applications
  - Local computation and manipulation of data
  - Multiple views on the data
  - Granular updates
- Open standards
  - Format for web delivery
    - Enhances scalability
    - Facilitates compression

General Advantages of XML for KR

1. Definition of self-describing data in worldwide standardized, non-proprietary format

2. Structured data and knowledge exchange for enterprises in various industries

3. Integration of information from different sources to uniform documents

XML offers new general possibilities, from which

AI knowledge representation (KR) can profit:

XML provides the most suitable infrastructure for knowledge bases on the Web (incl. for W3C languages such as RDF)

1. Additional special KR uses of XML are:

2. Exchange of knowledge bases between different AI languages

3. Exchange between knowledge bases and databases, application systems, etc.

4. Transformation/compilation of AI source programs using XML markup and annotations is possible

Bibliography

Abric'82	J. C. Abric, Cognitive Processes underlying Cooperation: The Theory of Social representation, in V. J. Derlega and Grzelak (eds.), Cooperation and Helping Behavior, New York: Academic Press.
Ait-Kaci & Nasr'86	Hasan Ait-Kaci and Roger Nasr, LOGIN: a logic programming language with built-in inheritance, Logic Programming 3, 185-215
Ait-Kaci'91	Hasan Ait-Kaci, "An overview of LIFE", in Next Generation Information System Technology: Proceedings of the First International East/West Data Base Workshop, Kiev USSR, October 1990, J.W. Schmidt & A.A. Stogny eds., Lecture Notes in Computer Science: volume 504, Springer-Verlag, New-York, Berlin, Heidelberg, 42-58.
Alben'97	L. Alben, Quality of Experience: Defining the criteria for effective Interaction Design, in Interactions 3.3 May+June 1997.
Allgayer & Reddig – Siekmann'90	Jürgen Allgayer and Carola Reddig-Siekmann, What KL-ONE lookalikes need to cope with natural language:scope and aspect of plural noun phrases, in Sorts and Types in artificial intelligence: Proceedings workshop, Eringerfeld, FRG, April 24-26, 1989, Karl Hans Bläsius, Ulrich Hedstück & Claus-Rainer Rollonger, eds., Lecture notes in artificial intelligence; subseries of lecture notes in computer science; edited by Jörg Siekmann; volume 418, Springer-Verlag, Berlin.
Allgayer'90	Jürgen Allgayer, SB-ONE^{+} : Dealing with sets efficiently, in proceedings Ninth European Conference on Artificial Intelligence, Stockholm, 6-10 August 1990, Luigia Carlucci Aiello, ed., Pitman Publishing, London.
Altenkruger'90	KBMS: aspects, theory and implementation, Information Systems 15.
Altobelli & Hoffmann'96	C:F: Altobelli and St. Hoffmann, Die Optimale Online-werbung für jede Branche. Was Nutzer von Unternehmensauftritten im Internet erwarten. Eine erste Analyse zur Online-Werbung für zehn Schlüsselbranchen, Im auftrag der MGM MediaGruppe München und des Spiegel-Verlag
Alty & Coombs '80	J. L. Alty and M. J. Coombs, Face-to-face guidance of university computer users-I: A study of advisory services, in International Journal of Man-Machine Studies, 12, 1980.
Anderson'99	Conceptual Support for the future DaimlerChrysler Telematics Business, Discussion Document, Anderson Consulting, 1999.
Anwar et al.'92	T. M. Anwar, H.W. Beck & S.B. Navathe, Knowledge Mining by impricise querying: a classification-based approach, in proceedings of the 8th International Conference of Data Engineering, Tempe, AZ, February 3-7,

1992.

Artale & Franconi'94 — Alessandro Artale and Enrico Franconi, A Computational account for a description logic of time and action, in Proceedings of the Fourth International Conference on Principles of Knowledge and Reasoning, KR'94, May 24-27, 1994, Bonn. Morgan Kaufmann Publishers, Inc. San Francisco, CA.

Artale and Franconi'94 — A. Artale and E. Franconi, A computational account for a description logic of time and action, proceedings of KR-94, 1994.

Artale et al'94 — Alessandro Artale, F. Cesarini, E. Grazzini, F. Pippolini & G. Soda, Modelling composition in a terminolgical language environment, in proceedings ECAI'94 workshop Parts and Wholes: Conceptual Part-Whole relations and Formal Mereology, Amsterdam, The Netherlands, August 8, 1994.

Attardi & Simi 81 — Attardi G and Simi M. Semantics of Inheritance and attributions in the description system Omega, A.I: Memo no. 642, MIT Artificial Intelligence Laboratory 1981

auf'm Hofe'96 — Harald Meyer auf'm Hofe, What is still to do in order to solve configuration problems in practice?, in proceedings KI-96 Workshop WRKP, 1996, DFKI, Kaiserslautern, Germany.

BA&H'99 — Business Opportunities in the Automotive Telematics Market, Booz Allen & Hamilton, Jan. 2000.

Baader & Hollunder '91 — Franz Baader and Bernhard Hollunder, A terminilogical Knowledge Representation System with Complete Inference Algorithms, in International Workshop on Processing Declarative KNowledge (PDK'91), Kaiserslautern, Germany, Lecture notes in Aritificail Intelligence, volume 567, Springer-Verlag, Berlin.

Baader & Hollunder'91 — Franz Baader and Bernhard Hollunder, KRIS:Knowledge Representation and Inference system, SIGART Bulletin 2, Special issue on implemented knowledge representation and reasoning systems.

Baader & Hollunder'92 — Franz Baader and Bernhard Hollunder, Embedding defaults into terminological knowledge representation formalisms in Proceedings of the third international conference on principles of knowledge representation and reasoning (KR'92), Cambridge, MA. Morgan Kaufman Publishers, Inc., SAn Mateo, CA.

Baader & Hollunder'93 — Franz Baader and Bernhard Hollunder, How to prefer more specific defaults in terminological representation system; or Making KRIS get a move on, in Proceedings of the third international conference on principles of knowledge representation and reasoning (KR'92), Cambridge, MA. Morgan Kaufman Publishers, Inc., SAn Mateo, CA.

Baader et al.'94 — Franz Baader, Martin Buchheit & Bernhard Hollunder, Cardinality Restrictions on Concepts, in DFKI Research Report RR-93-48

Baader'90a — Franz Baader, A formal definition for the expressive power

	of knowledge representation languages, in proceedings Ninth European Conference on Artificial Intelligence, Stockholm, 6-10 August 1990, Luigia Carlucci Aiello, ed., Pitman Publishing, London.
Baader'90b	Franz Baader, Terminological Cycles in KL-ONE-based Knowledge Representation Languages, in proceedings Eighth National Confernce on Artificial Intelligence, Boston, MA, 29 july-3 August 1990, AAAI Press/MIT Press, Menlo Park, CA.
Baader'96	Franz Baader, Extensions of Terminological Knowledge Representation Languages for Technical Applications, in proceedings KI-96 Workshop WRKP, 1996, DFKI, Kaiserslautern, Germany.
Badea'95	Liviu Badea, A unified architecture for knowledge representation and reasoning based on terminological logics, International workshop on terminological logics, Roma 1995.
Baecker & Buxton'87	R. M. Baecker and W. A. S. Buxton An Historical and Intellectual Perspective In R. M. Baecker and W. A. S. Buxton (eds.), Readings in Human-Computer Interaction, San Mateo, California:Morgan Kaufman Publishers.
Balabanovic'97	M. Balabanovic, An adaptive web page recommendation service, in proceedings of the 1st International Conference on Autonomous Agents, Marina del rey, Ca.
Bänsch'96	A. Bänsch, Verkaufspsyschologie und Verkaufstechnik, 6, Auflage, Oldenbourg Verlag, München.
Benaroch & Vasvasud'96	Michel Benaroch and Satish P. Vasvasud, On the Roles of Information Technology in Mass Customization, in INFORMS'96 conference, Washington D.C:, May 1996.
Bergmann & Gerlach 1986	QUIRK - Implementierung einer TBox zur Repräsentation begrifflichen Wissens. WISBER Memo Nr. 11, Universität Hamburg.
Bergmann et al. '87	IRS: The Internal Representation Language. WISBER Bericht Nr. 14, Universität Hamburg.
Bertino et al.'93	Elisa Bertino, Maria Damiani, Paolo Randi & Luca Sampinato, An advanced information management system, in proceedings of IEEE research issues in data engineering-interoperability among multidatabase system workshop, Vienna, Austria.
Bhattacharjee'97	E. Bhattacharjee, Profi-Marketing im Internet. Erfolgreiche Strategien und Konzepte und Tips, Freidurg i. Br.
Borgida '92	Alexander Borgida, „A new look at the foundations and utility of description logics (or terminal logics are not just for thr flight less birds)," Rutgers University, Technical Report DCS TR 295, New Brunswick, NJ.
Borgida et al.'89	Alexander Borgida, Ronald J. Brachman, Deborah L. MacGuinness & Lori Alperin Resnick, „CLASSIC: a structural data model for objects," in proceedings of ACM-SIGMOD 1989.
Borgida et al.'89	Alexander Borgida, Ronald J. Brachman, Deborah L.

McGuinness & Lori Alperin Resnick, CLASSIC: a structural data model for objects, in proceedings of ACM-SIGMOD 1989, International conference on management of data, Portland, OR May 1989, James Clifford, Bruce Lindsay & David Maier eds. ACM Press, New York.

Borgida et al.'93 — Alexander Borgida and Ronald J. Brachman, „Loading data into description reasoner," in ACM-SIGMOD International Conference on Management Data, Washington, DC, May 1993, 217-226

Borgida et al.'94 — Alexander Borgida & Peter F. Patel-Schneider, „A semantics and complete algorithm for subsumption in the CLASSIC description logic," journal of artificial intelligence Research 1, 277-308

Borgida'94 — Alexander Borgida, „Description logics for querying databases," in proceedings of the international workshop on description logics, Bonn, Germany, May 28-29, 1994, DFKI Kaiserslautern/Saarbrücken, Germany.

Bosak & Bray'99 — Jon Bosak and Tim Bray, XML and the Second-Generation Web, in Scientific American, May 1999, http://www.sciam.com/1999/0599issue/0599bosak.html

Bosak'97 — Jon Bosak, XML, Java and the future of the Web, in http://sunsite.unc.edu/sun-info/standards/xml/why/xmlapps.htm

Bowen & Bahler'91 — J Bowen and D. Bahler, Conditional Existence of variables in generalized constraint networks, in proceedings of ninth national conference on artificial intelligence, AAAI Press, 1991.

Brachman & Schmolze'85 — Ronald J. Brachman & James G. Schmolze, „An overview of the KL-ONE knowledge Representation System," Cognitive Science, 171-216

Brachman & Schmolze'85 — R.J. Brachman and J.G. Schmolze, „An overview of the kl-one knowledge representation system" , Cognitive Science 9, 171-216, 1985.

Brachman 77 — R.J. Brachman, „Whats in a concept: Structuralk foundations for semantic networks", international journal of man-machine studies 9, 127-152, 1977.

Brachman 79 — R.J. Brachman, „On the Epistemological Status of Semantic Networks", in [Findler 79], 3-50.

Brachman 90 — Ronald J. Brachman, The future of knowledge representation: extended abstract, in Proccedings AAA1'90, Boston MA, july/august 1990, AAAI press/MIT Press, Menlo Park, CA.

Brachman et al.'82 — Ronald J. Brachman and Hector J. Levesque,"Competence in knowledge representation," in proceedings national conference on artificial intelligence, Pittsburgh, PE, 18-20 August, AAAI, 189-198.

Brachman et al.'84 — Ronald J. Brachman and Hector J. Levesque, „The tractibility of subsumption in frame based description languages," in proceedings national conference on artificial intelligence, Austin, TX, 6-10 August 19984, William

Kaufmann, Los Altos, CA.

Brachman et al.'85 Ronald J. Brachman, Richard E. Fikes & Hector J. Levesque, „KRYPTON: a functional approach to knowledge representation, in readings in knowledge representation, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 411-429.

Brachman et al.'85 Ronald J. Brachman, Victoria Pigman Gilbert & Hector J. Levesque, „An essential hybrid reasoning system: knowledge and symbol level accounts of KRYPTON, „in Proceedings IJCAI'85 Morgan Kaufmann Publishers, Los Altos, CA, 532-539.

Brachman et al.'86 Ronald J. Brachman and Hector J. Levesque,"The knowledge level of a KBMS",Topics in information systems; edited by Michel L. Brodie, John Mylopoulos and Joachim W. Schmidt, Springer-Verlag, New York, NY.

Brachman et al.'90 Ronald J. Brachman, Deborah L. Mcguinness, Peter F. patel-Schneider, Lori Alperin Resnick & Alexander Borgida, „Living with CLASSIC: when and how to use a KL-ONE like language," in principles of semantic networks: explorations in the representation of knowledge, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 401-456.

Brachman et al.'92 Ronald J. Brachman, Peter G. selfridge, Loren G. Terveen, Boris Altman, Alexander Borgida, Fern Halper, Tom Kirk, Alan Lazar, Deborah L. Mcguinness & Lori Alperin Resnick, Knowledge representation support for data archaeology," in proceedings of the first international conference on information and knowledge management, Baltimore, MD, November 1992, 457-464.

Brachman et al.'93 Ronald J. Brachman, Alexander Borgida, Deborah L. Mcguinness, Peter F. patel-Schneider & Lori Alperin Resnick, „The CLASSIC knowledge representation system, or, KL-ONE: the next generation," in proceedings of the international conference on fifth generation systems, Tokyo, Japa, june 1993.

Brachman et. al.'91 R.J. Brachman, A. Borgida, D:L: McGuinness, P.F: ÜPatel-Schneider, and L.A. Resnick, „Living with CLASSIC: When and how tro use a KL-ONE-like language" in principles of semantic networks, ed. john sowa, 401-456, morgan kaufmann, san mataeo, CA. 1991.

Brachman'77 Ronald J. Brachman, „What's in concept: structural foundations for semantic networks," International Journal of Man-Machine Studies 9, 127-152.

Brachman'78 Ronald J. Brachman, A structural paradigm for representing knowledge, „ Bolt Barnek and Newman Inc., BBN report no. 3605, cambridge, MA, revised version of Brachman's Ph.D. thesis, Harvard University, 1977.

Brachman'79 Ronald J. Brachman, „On the epistemological status of

semantic networks," in Associative networks: representation and use of knowledge by computers, Nicholas V. Findler, ed., Academic Press Inc., New York, NY, 3-50.

Brachman'85 — Ronald J. Brachman, „I lied about the trees, or , defaults and definitions in knowledge representation," AI Magazine 6, 80-93.

Brachman'90 — Ronald J. Brachman, „The future of knowledge representation: extended abstract, „ in proceedings Eigth national conference on artificial intelligence, boston, MA, 29 july-3 august 1990. AAAI Press/MIT Press, Menlo Park, CA, 1082-1092.

Brachman'92 — Ronald J. Brachman, „Reducing" CLASSIC to practice: knowledge representation theory meets reality," in proceedings of the third international conference on principles of knowledge representation and reasoning (KR'92), Cambridge, MA, October 25-29, 1992, Morgan Kaufmann Publishers, Inc., San Mateo, CA, 247-258.

Brenda Laurel'93 — Brenda Laurel, Computers as Theatre, Reading, Mass. Addison-Wesley,1993.

Bürckert et al.'96 — Hans-Jürgen Bürckert, Werner Nutt and Christian Seel, The Role of Formal Knowledge Representation in Configuration, in proceedings KI-96 Workshop WRKP, 1996, DFKI, Kaiserslautern, Germany.

BusinessWeek'96 — Making money on the net, in Business week September 23, 1996.

BusinessWire'99 — Driving the virtual highway: How the car buying experience is changing for the better, January 1999.

Cain'98 — J. Cain, Experience-Based Design: Towards a Science of Artful Business Innovation, in Design Management Journal, Fall 1998.

Calvanese et al.' 98b — Diego Calvanese, Giuseppe De Giacomo and Maurizio Lenzerini, Daniele Nardi and Riccardo Rosati, Description Logic Framework for Information Integration, in proceedings KR'98.

Calvanese et al.'97 — Diego Calvanese, Giuseppe De Giacomo and Maurizio Lenzerini, Representing and reasoning on SGML documents. In Proceedings of the tenth International Symposium on Methodologies for Intelligent Systems (ISMIS-97), 1997.

Carlson'97 — R. Carlson, Experienced Cognition, New York: Lawerence Erlbaum associates, 1997.

Carpenter '92 — Bob Carpanter, The logic of typed feature structures: with applications to unification grammars, logic programs and constraint resolution, Cambridge tracts in computer science, volume 32, Cambridge University Press.

Carrive et al. '98 — Jean Carrive, François Pachet, Remi Ronfard, Using Description Logics for Indexing Audiovisual Documents, in proc. 1998 International Workshop on Description Logics (DL'98) IRST, Povo -- Trento, Italy, June 6 - 8, 1998.

Catterall et al.'91 — B. J. Catterall, B. C. Taylor and M. D. Galer, The HUFIT

	planning analysis and specification toolset: Human factors as a normal part of the IT product design processing. In J. Kart (ed.), Taking Software Design Seriously. London: Acedemeic Press.
Chandrasekharan'83	B. Chandrasekharan, Towards a taxonomy of problem solving types, AI Magazine 4, 1983.
Chandrasekharan'86	B. Chandrasekharan, Generic tasks in knowledge-based reasoning: high level building blocks for expert system design, in IEEE expert Vol. 1, Nr. 3, 1986 Humpherys.
Chandrasekharan'87	B. Chandrasekharan, Towards a functional architecture for intelligence based on generic information processing tasks, in Proceedings IJCAI'87, John McDermott, ed., Morgan Kaufmann Publishers, Los Altos, CA.
Chattell''98	A. Chattell, Creating Value in the Digital Era, New York
Checkland and Scholes'90	P. Checkland and JScholes, Soft Systems Methodolgy in Action, Chichester:John Wiley and Sons.
Chin, D.N.'83	D. N Chin., "A Case Study of Knowledge Representation in UC" In the Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Kerlsruhe, Germany, August 1983.
Churchill et al.'85	G. A Churchill, N. M. Ford and O.C. Walker, Sales Force Management: Planning, Implementation, Controll, Homewood I11., Georgetown/Ont.
CIA'00	http://www.c-i-a.com/199809iu.htm
Collins'84	Robert H. Collins, "Artificial Intelligence in personal selling" in Journal of Personal Selling & Sales Management, Vol. 4, (May 1984), pp. 58-66.
Compatangelo et al '98	Ernesto Compatangelo, Francesco M. Donini, and Gianni Rumolo, DL-based Support to Domain Engineering, in proc. 1998 International Workshop on Description Logics (DL'98) IRST, Povo -- Trento, Italy, June 6 - 8, 1998.
Conhaim'98	
Coombs & Alty '80	M. J. Coombs and J. L. Alty, Face-to-face guidance of university computer users-II: Characterizing advisory interactions, in International Journal of Man-Machine Studies, 12, 1980.
Cover'98	Robin Cover, XML and Semantic Transparency, http://www.oasis-open.org/cover/xmlAndsemantics.html
Craw et al.'92	"Specification of Consultant-2" Deliverable 5.5, Machine Learning Toolbox ESPRIT project P2154, 1992.
CSC-features	CSC-features: http://www.cse.com/features/110199_features.html
Davis'84	"Interactive Transfer of Expertise", in B. Buchanan and E. H. Shortliffe, (Eds.), Rule Based Expert Systems, pp. 171-205. Addison-Wesley, Reading, MA., 1984.
Deering et al.'81	"PEARL: An Efficient Language for Artificial Intelligence Programming", In the Proceedings of the Seventh International Joint Conference on Artificial Intelligence. Vancouver, British Columbia. August 1981.
Deering et al.'82	The PEARL Users Manual, Berkley Electronic Research

Laboratory Memorandum No. UCB/ERL/M82/19. March, 1982.

Devanbu & Litman'91 — P. T. Devanbu and D. J. Litman, Plan-based terminological reasoning. Proceedings KR-91, 1991.

Devanbu et al.'89 — Premkumar T. Devanbu, Peter G. Selfridge Bruce W. Ballard & Roland J. Brachman, "A knowledge-based software information system," in proceedings IJCAI'89, Detroit, MI, 20-25 August 1989, Morgen Kaufmann Publishers, San Mateo, CA, 110-115

Devanbu et al.'90 — Premkumar T. Devanbu, Roland J. Brachman, Peter G. Selfridge & Bruce W. Ballard, "LaSSIE: a knowledge-based software information system," in proceedings of the 12th international conference on software engineering, Nice, France, March 26-30, 1990, 249-261. IEEE computer Society Press Reprint.

Devanbu et al.'91 — Premkumar T. Devanbu & Diane J. Litman, Plan-based terminological reasoning," in proceedings of the second international conference on principles of knowledge representation and reasoning (KR'91), Cambridge, MA, April 22-25, 1991, Morgen Kaufmann Publishers, San Mateo, CA, 128-138.

Dewey'63 — J. Dewey, Experience and Education, New York: Macmillan, (reprint) 1963.

Dewey'80 — J. Dewey, Art as Experience, New York: Perigee, (reprint) 1980.

Diaz '91 — Prieto-Diaz 91, Prieto-Diaz, R. Domain Analysis and Software Systems Modeling. Los Alamos, CA: IEEE Computer Society
Press, 1991.

Dietz'96 — W. Dietz, Das Handbuch für das Automobil-Marketing strategien, Konzepte, Instrumente 2. Auflage, Verlag Moderne Industrie.

Dionne et al.'92 — Robert Dionne, Eric Mays & Frank J. Oles, A non-well-founded approach to terminological cycles, in Proceedings Tenth National Conference on Artificial Intelligence, July 1992, San Jose, CA. AAAI Press/MIT Press, Menlo Park, CA.

Donini et al. '91a — Francesco M. Donini, Maurizio Lenzerni, Daniele Nardi & Werner Nutt, The complexity of concept languages," in proceedings of the second international conferences on Principles of knowledge representation and reasoning(KR'91), Cambridge, MA, April 22-25, 1991, Morgen Kaufmann Publishers, San Mateo, CA, 151-162

Donini et al. '91b — Francesco M. Donini, Maurizio Lenzerni, Daniele Nardi & Werner Nutt, „Tractable Concept Languages," in proceedings IJCAI'91, Sydney Australia, August 24-30, 1991, Morgen Kaufmann Publishers, San Mateo, CA, 458 - 463

Donini et al. '94 — Francesco M. Donini, Maurizio Lenzerni, Daniele Nardi & Andrea Schaerf,"Deduction in concept languages: from

	subsumption to instance checking," journal of Logic and computation 4, 423-452.
Doyle & Patil'89	Jon Doyle & Ramesh S. Patil, Two Dogmas of Knowledge Representation: Language Restrictions, Taxonomic Classification, and the Utility of Representation Services.
Doyle et al.'91	Jon Doyle & Ramesh S. Patil, Two theses of knowledge representation: language restrictions, taxonomic classification and the utility of representation services," Artificial Intelligence 48, 261 - 297.
Draft & Lengel'86	R.L. Draft and R.H. Lengel, Organizational Information Requirements, Media Richness and Structural Design, in Management Science, 32, 5, 1986.
Duddenhöfer'98	F. Duddenhöffer, Car Distribution Studie 1998, Gelsenkirchen 1998.
Dudenhöffer'99	F. Dudenhofer and Carina Densing, Übersichts Artikel zu Car Distribution Studie 1999.
Engehausen et al.'96	Anne Engehausen, Simone Pribbenow and Ulf Töter, Multiple Part-Hierarchies, in proceedings KI-96 Workshop WRKP, 1996, DFKI, Kaiserslautern, Germany.
Ensing et al.'94	Marco Ensing, Ray Paton, Piet-Hein Speel & Roy Rada, An object-oriented approach to knowledge representation in a biomedical domain, Artificial Intelligence in Medicine 6, 459-482.
Erickson'97	Th. Erickson, Social Interaction on the net: virtual community as participatory Genre, in proc. Of the thirteenth annual Hawaii international conference on system sciences, Hawaii, 1997, Vol. VI.
Feigenbaum '93	Feigenbaum, Tiger in the cage, Invited Talk, AAAI'93
Ferguson et al.'87	"A Knowledge-based Sales Assistant for Data Communications" in Network IEEE,1987, pp. 1634-1642.
Ferguson et al.'87 b	"Generic Strategies and Representations for Communications Networks Sales"
Findler '79	N.V. Findler (Ed.), Associative Networks: Representation and Use of knowledge by computers, New York: Academic Press, 1979.
Fleischanderl et al. '98	Gerhard Fleischanderl, Gerhard E. Friederich, Alois Haselböck, Herwig Schreiner and Markus Stumptner, Configuring Large Systems Using Generative Constraint Satisfaction, in IEEE Intelligent Systems & their applications, vol. 13, no. 4, July/August 1998.
Fleischanderl'99	Gerhard Fleischanderl, Overview of configurators as effective tools for corporate knowledge management, in Proceedings AAAI'99 Workshop on Configuration, July 1999, Orlando, Florida, USA
Fleming'98	J. Fleming, Web Navigation – Designing the User Experience, sebaastopol CA, O'reilly
Fliegner '88	HOKUSPOKUS -Verwendung terminologischen Wissens in der Analyse von Quantorenskopus und Distributivität. in Hoeppner, W.(ed.), GWAI-88, Springer, Berlin, 1988.
Fogg'99	B.J. Fogg , Persuasive Technologies, in Communication of

the ACM, 42(5), 1999.

Forlizzi'97 — J. Forlizzi, Design for Experience: An Approach to Human-Centered Design, Master's Thesis, School of Design, Carnegie Mellon University, 1997.

Forrester Research'99 — New-car buying races online, January 1999.

Franconi et al.'92 — Enrico Franconi, Bernardo Magnini & Oliviero Stock, Prototypes in a hybrid language with primitive descriptions, in Semantic Networks in artificial intelligence, Fritz Lehmann, ed., Modern applied mathematics and computer science; edited by Erwin Y. Rodin; Volume 24, Pergamon Press, Oxford, England, 543-555, published as a special issue of the journal Computers & Mathematics with Applications 23.

Franconi'90 — Enrico Franconi, The YAK (yet Another Krapfen) manual, IRST, Manual 9003-01, Trento, Italy.

Franconi'93 — Enrico Franconi, A treatment of plurals and plural quantifications based on a theory of collections, Minds and Machine 3, 453-474, special issue on knowledge representation for natural language processing

Frederking and Gehrke '87 — Resolving Anaphoric References in a DRT-based Dialogue system: Part 2: Focus and Taxonomic Inference. WISBER Berich Nr. 17, Siemens AG, München.

Freuder'92 — E. Freuder, Constraint Solving Techniques, Constraint Programming, vol. 131, B. B Mayoh, E. Tyngu and J. Penjaen, Eds. NATO ASI Series F: Computer and system Sciences, Springer Verlag, Berlin.

Frühwirth & Hanschke'93 — Thom Frühwirth and Phillip Hanschke, Terminological reasoning with constraint handling rules, in workshop on Principles and Practice of constraint Programming, Newport, Rohde Island, USA.

Gaskin'99 — James E. Gaskin, A Hitchhiker's Guide to XML, http://www.zdnet.com/intweek/stories/news/0,4164,2270603,00.html

Gemni Consulting'99 — Cars Online Europe 1999, Gemini Consulting.

Gielisch'89 — Praxis Semester Bericht, Forschung & Technologie, DaimlerBenz, Berlin, 1989

Goos & Thakar '96 — Klaus Goos & Sunil Thakar, A Case Based Automobile Sales Advisory system on the web. Presented at KBCS'96, Poster. International conference on knowledge based computer systems, Mumbai, India. Dec. 16-18, 1996.

Greeenbaum and Kyng'91 — J. Greenbaum and M Kyng (eds.), Design at Work: Cooperative Design of Computer Systems. Hillsdale, New Jersey: Lawrence Erlbaum associates.

Grosz & Sidner'86 — Attention, Intentions, and the Structure of discourse. In Computational Linguistics, Vol. 12, No. 3, pp. 175-204

Gruber & Olsen'94 — Thomas R. Gruber and Gregory R. Olsen, An ontology for engineering mathematics, in Proceedings of fourth International Conference on Principles of Knwolege Representation and Reasoning, KR'94, May 1994, Bonn, Germany. Morgan Kaufmann Publishers, Inc. San Mateo,

CA.

Gruber,Tannenbaum & Weber'92 — Thomas R. Gruber, Jay M. Tanenbaum & Jay C. Weber, Toward a knowledge medium for collaborative product development in Proceedings of the second International Conference on Artificial Intelligence in Design, Pittsburgh, June 1992. Kluwer Academic Publishers.

Gruber'91 — Thomas R. Gruber, The role of common ontology in achieving sharable, reusable knowledge bases, in Proceedings KR'91, Cambridge, MA, April 1991. Morgan Kaufmann Publishers, Inc., San mateo CA.

Guha & Bray'97 — R. V. Guha and Tim Bray, Meta Content Framework Using XML, http://www.textuality.com/mcf/NOTE-MCF-XML.html

Guha & Lenat'93 — Ramanathan V. Guha and Douglas B. Lenat, Re:CyCLing paper reviews, Artifical Intelligence 61, 149-174.

Guha & Lenat'94 — Ramanathan V. Guha and Douglas B. Lenat, Enabling agents to work together, communications of the ACM 37, 127-142.

Haag'98 — Albert Haag, Sales Configuration in Business Processes, in IEEE Intelligent Systems & their applications, vol. 13, no. 4, July/August 1998.

Hagel & Armstrong '97 — J. Hagel III and A.G. Armstrong, Net Gain – Expanding Markets Through Virtual Communities, Boston, Massachussets.

Hagel & Singer'99 — J. Hagel III and M. Singer, Net Worth-Shaping Markets when Customers Make the Rules, Boston Massachussets.

Hanschke'93 — Phillip Hanschke, A declarative integration of terminological, constraint-based, data-driven and goal directed reasoning, Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI, Research Report RR-93-46, Kaiserslautern, Germany.

Haselbock`93 — A. Haselbock, Knowledge-Based Configuration and Advanced Constraint Technologies, doctral thesis, Institut für Informationssysteme, Technische Universität Wien, Vienna, 1993.

Hasenfratz et al.'93 — Pierre-Andre Hasenfratz, Philippe Volle & Maurizio De Cecco'93, BACK++ Users's Manual, Non Standard Logics S.A., Paric, France.

Hayes'80 — P.J. Hayes, „The logic of Frames in [mentzing 80], 46-61, 1980.

Hayes'85 — Patrick J. Hayes, Naive physics I: ontology for liquids, in Formal theories of the commonsense world, Jerry R. Hobbs & Robert C. Moore, eds. Ablex, Norwood, NJ.

Hayes-Roth et al.'83 — Frederick Hayes-Roth, Donald a Wareman & Douglas B. Lenat, Building Expert Systems, Addison-Weseley, Reading, MA.

Heiderscheit & Skovgaard'99 — Dan Heiderscheit ansd Hans Jorgen Skovgraard, Visualisation of configurations: simplifying configuration user interfaces, in Proceedings AAAI'99 Workshop on Configuration, July 1999, Orlando, Florida, USA

Heinsohn et al.'92	Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel & HansJürgen Profitlich, RAT: representation of actions using terminological logics, Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI, Technical Report, Kaiserslautern, Germany.
Heinsohn et al.'94	Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel & HansJürgen Profitlich, An empirical analysis of terminological representation systems, Artificial Intelligence 68, 367-397.
Heinsohn'91	Jochen Heinsohn, A Hybrid approach for modeling uncertainity in terminological logics, in proceedings of the First European Conference on Symbolic and Quantitative Approaches for Uncertainity, Marseille, France, October 1991, R Kruse & P. Siegel, eds., Springer Lecture Notes, Number 548, Springer Verlag, Marseille, France.
Hinzmann '90	?? Article
Hoffman & Chatterjee'96	D.L. Hoffman and P. Chatterjee, Commercial Scenarios for the web: Opportunities and challenges, http://jcmc.huji.ac.il/vol1/issue3/hoffman.html
Hoffmann & Thakar '91	Achim G. Hoffmann and Sunil Thakar, Acquiring Knowledge by efficient query learning IJCAI-91 Sydney,Australia, August 1991.
Hoffmann & Thakar '91	Achim G. Hoffmann and Sunil Thakar, Overcoming Knowledge Acquisition Bottleneck through Learning via Querries Poster KBCS-90 Pune/India
Hoffmann & Thakar '93	Achim G. Hoffmann and Sunil Thakar, Facilitating Sales Consultation through Case Based Reasoning presented at EWCBR-93, Workshop on Case-Based Reasoning, Nov. 1993,Kaiserslautern, Germany.
Hollunder & Baader'91	Bernhard Hollunder & Franz Baader, Qualifying number restrictions in concept languages, in Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91), Cambridge, MA, April 1991, Morgan Kaufmann Publishers, Inc., San Mateo, CA.
Hollunder '90	Bernhard Hollunder, Hybrid Inferences in KL-ONE-based knowledge representation systems in Proceedings of the 14th German Workshop on Artificial Intelligence (GWAI-90), Eringerfeld, September 1990, Heinz Marburger, ed., Springer-Verlag, Berlin.
Hollunder et al.'90	Bernhard Hollunder, Werner Nutt & Manfred Schmidt-Schauß, Subsumption algorithms for concept description languages, in Proceedings 9th ECAI, Stockholm, August 1990, Luigia Carlucci Aiello, ed., Pitman Publishing, London.
Hollunder et al.'94	Bernhard Hollunder, Armin Laux, HansJürgen Profitlich, & Thomas Trenz, KRIS-manual, Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI, Saarbrücken, Germany.
Hoppe et al.'93	Thomas Hoppe, Carsten Kindermann, J. Joachim Quantz,

	Albrecht Schmiedel & Martin Fischer, „BACK V5: tutorial and manual,“ Technische Universität Berlin, KIT-Report 100, Berlin, Deutschland.
Horacek et al.’88	"From Meaning to Meaning - a Walk Through Wisber" in Hoeppner, W.(ed.), GWAI-88, Springer, Berlin, pp118-129, 1988.
Hudspith’97	S. Hudspith, Utility, Ceremony and Appeal: A Framework for Considering Whole Product Function and User Experience, DRS NEWS (electronic version), Vol. 2, No. 11, November 1997.
Humpherys, Lindberg & Hole’91	Betsy L. Humphreys, Donald A. B. Lindberg and William T. Hole, Assessing and enhancing the value of ULMS knowledge sorces, in proceedings of the fifteenth annual symposium on computer applications in medical care, a conference of the american medical informatics association, washington, DC, November, 1991, Paul D. Clayton, ed. McGraw-Hill, New York.
IDC’00	Project Atlas: A Global Survey of Advanced Web Users and Buyers by Barry Parr
Iocchi and Nardi’97	Luca Iocchi and Daniele Nardi, Information access in the web, in proc. WEBNET-97, 1997.
Jackson & Lefrere ’84	P. Jackson and P. Lefrere, On the application of rule-based techniques to the design of advice giving systems. International Journal of Man-Machine Studies, 20, 1984.
Karpinski’99	R. Karpinski, GM Expands E-Business Scope, in Internetweek 15.03.1999, http://www.techweb.com/wire/story/TWB19990315S0013
Kastner et al.’86	"A Knowledge-Based Consultant for Financial Marketing" The AI Magazine, Winter 1986, pp. 71-79.
Katz’94	Katz, S., et al. Glossary of Software Reuse Terms. Gaithersburg, MD: National Institute of Standards and Technology, 1994.
Kern’90	E. Kern, Der Interaktionsansatz im Investitionsgütermarketing, Berlin 1990.
Kindermann & Quantz’90	Carsten Kindermann and Joachim Quantz, Implementation of the BACK system version 4, Teschnische Universität Berlin, KIT-Report 78, Berlin, Germany.
Kindermann’90	Carsten Kindermann, Class instances in a terminological framework: an experience report, in Proceedings of the 14th German National Workshop on Artificial Intelligence (GWAI-90), Eringerfeld, September 1990, Heinz Marbuger, ed., Springer Verlag, Berlin.
Kindermann’92	Carsten Kindermann, Retraction in terminological knowledge bases, in Proceedings ECAI’92, Vienna, Austria. John Wiley, Chichester, UK.
Klein & Szyperski’98	S. Klein and N. Szyperski, Referenzmodell zum Electronic Commerce: http://www-wi.uni-muenster.de/wi/literatur/refmod/rm-ecinf.html
Kling & Star’98	Rob Kling and Leighg Star, Human Centered Systems in the Perspective of Organization and Social Informatics,

in Computers and Society 28(1) March 1998.

Kobsa'91a — Alfred Kobsa, First Experiences with the SB-ONE knowledge representation workbench in natural-language applications, SIGART Bulletin 2, special issue on implemented knowledge representation and reasoning systems.

Kobsa'91b — Alfred Kobsa, Utilizing Knowledge: the component of the SB-ONE knowledge representation workbench, in Principles of semantic networks: explorations in the representation of knowledge, John F. Sowa, ed., The Morgan Kaufmann series in representation and reasoning, Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Kodratoff et al.'92 — "building a Machine Learning Toolbox" in Enhancing the knowledge Engineering Process, L. Steels and B. Lepape (Editors), pp. 81-108, Elsevier Science Publishers B.V. 1992.

Koehler'94 — Jana Koehler, An Application of Terminological Logics to Case-based Reasoning. In proc. KR'94

Körner'99 — V. Körner, Kurzkonzept Customer Profiling, http://www.ecc.ch/customer_profiling/kurzkonzept_cp.pdf

Kortüm'93 — Gerd Kortüm, How to compute 1+1? A proposal for the integration of external functions and computed roles into BACK, Technische Universität Berlin, KIT-Report 103, Berlin, Germany.

Kosba et al.'94 — A. Kobsa, D. Müller, and A. Nill, KN-AHS: An adaptive hypertext client of the user modeling system BGP-MS, in proceedings of the fourth international conference on user modeling, Hyannis, MA.

Kotler & Bliemel'98 — P. Kotler, F. Bliemel, Marketing Management, 9. Auflage, Stuttgart.

Kotler & Bliemel'99 — Marketing Management, Analyse, Planung, Umsetzung und Steuerung. 9. Auflage, Stuttgart.

Kramer'93 — J. Kramer, Philosophie des Verkaufens: ein situativer Ansatz, Wiesbaden

Kumar'95 — http://blackcat.brynmawr.edu/~dkumar/UGAI/planning.html .

Lander'97 — Richard Lander, The Search Foe Metadata, http://pdbeam.uwaterloo.ca/~rlander/Metadata/metadata.html

Lassila'97 — Ora Lassila, Introduction to RDF Metadata, http://www.w3.org/TR/NOTE-rdf-simple-intro-971113.html

Lebergott'93 — Stanley Lebergott, Pursuing Happiness: American Consumers in the twentieth century, Princeton University Press, Princeton, 1993.

Lenat & Feigenbaum'91 — Douglas B. Lenat & Edward A. Feigenbaum, On the threshold of knowledge, Artificial Intelligence 47.

Lenat & Guha'90 — Douglas B. Lenat & Ramanathan V. Guha, Building large Knowledge-based systems: representation and inference in

	the Cyc Project, Addison-Weseley Publishing Company, Inc., Reading, Massachusetts.
Lenzirini & Schaerf '91a	Maurizio Lenzerini & Andrea Schaerf, Concept languages as query languages, in Proceedings Ninth National Conference on Artificial Intelligence, July 1991, AAAI Press/MIT Press, Menlo Park, CA.
Lenzirini & Schaerf '91b	Maurizio Lenzerini & Andrea Schaerf, Querying concept-based knowledge bases, in International Workshop on Processing Declarative Knowledge (PDK'91), Kaiserslautern, Germany, July 1991, Harold Boley & Michael M. Richter, eds., Lecture notes in artificial intelligence; subseries of lecture notes in computer science; edited by Jörg Siekmann; volume 567, Springer-Verlag, Berlin.
Lenzirini et al.'90	Maurizio Lenzerini, Daniele Nardi & Maria Simi, Inheritance hierarchies in knowlege representation and programming languages, John Wiley & Sons, Chichester, UK.
Levesque & Brachman'85	Hector J. Levesque & Ronald J. Brachman, A fundamental tradeoff in knowledge representation and reasoning, in Readings in knowledge representation, ronald J. Brachman & Hector J. Levesque, eds., Morgan Kaufmann Publishers, Inc. Los Altos, Ca.
Levesque & Brachman'87	Hector J. Levesque & Ronald J. Brachman, Expressiveness and tractibility in knowledge representation and reasoning, Computational Intelligence 3.
Levesque'84a	Hector J. Levesque, The Logic of incomplete knowledge bases, in On conceptual modelling: perspectives from Artificial Intelligence, Databases and Programming Languages, Michael L. Brodie, John Mylopoulos & Joachim W. Schmidt, eds. Topics in information systems;Springer Verlag, New York, NY.
Levesque'84b	Hector J. Levesque, Foundations of a functional approach to knowledge representation, Artificial Intelligence 23.
Lindberg & Humpherys 1990	Donald A. B. Lindberg and Betsy L. Humphreys, The UMLS knowledge sources: tools for building better user interfaces, IEEE computer society press, Los Alamitos, CA.
Liver & Gantenbein'99	Beat Liver and Dieter Gantenbein, Value chains and Mass customization, in Proceedings AAAI'99 Workshop on Configuration, July 1999, Orlando, Florida, USA
Luck et al.'86	Kai von Luck, Bernhard Nebel, christof Peltason & Albrecht Schmiedel, BACK to consistency and incompleteness, in Proceedings of the 9th German Workshop on Artificial Intelligence (GWAI-85), Dassel/Solling, September, 1985 Herbert Stoyan, ed., Springer Verlag, Berlin.
Luck et al.'87	Kai von Luck, Bernhard Nebel, christof Peltason & Albrecht Schmiedel, The anatomy of the BACK system. Technische Universität Berlin, KIT-Report 41, Berlin, Germany

Luck'86	Kai von Luck, Semantic networks with number restricted roles or another story about Clyde, in GWAI-86 and Österreichische Artificial-Intelligence Tagung., Ottenstein/Niederösterreich, September 1986, Claus-Rainer Rollinger & Werner Horn, eds. Springer Verlag, Berlin.
Luck'89	Kai von Luck, Repräsentation assertionalen Wissens im BACK-System: Eine fallstudie, Technische Universität Berlin, KIT-Report 72, Berlin, Germany.
Ludwig et al '98	Bernd Ludwig, Günther Görz, Heinrich Niemann, Combining Expression and Content in Domains for Dialog Managers, in proc. 1998 International Workshop on Description Logics (DL'98) IRST, Povo -- Trento, Italy, June 6 - 8, 1998.
MacGregor & Brill'92	Robert M. MacGregor and David Brill, Recognition algorithms for the LOOM classifier, in Proceedings Tenth National Conference on Artificial Intelligence, July 1992, San Jose, CA, AAAI Press/MIT Press, Menlo Park, CA.
MacGregor '91a	Robert M. MacGregor, Inside the LOOM description Classifier, SIGART Bulletin 2, special issue on implemented knowledge representation and reasoning systems.
MacGregor '91b	Robert M. MacGregor, The evolving technology of classification-based representation systems, in Principles of semantic networks: explorations in the representation of knowledge, John F. Sowa, ed., The Morgan Kaufmann series in representation and reasoning, Morgan Kaufmann Publishers, Inc., San Mateo, CA.
MacGregor '92	Robert M. MacGregor, Representing reified relations in LOOM, in Symposium:Propositional Knowledge Representation, Stanford University, March 1992, AAAI spring Symposium series, Stanford, CA.
MacGregor'88	Robert M. MacGregor, „A deductive pattern matcher", in proceedings of the seventh national conferencew on Artificial intelligence. pp 403-408, 1988.
Maes'94	P. Maes, Agents that reduce work and information overload, Communications of the ACM, 37(7), 1994.
Mahling & Craven'95	Dirk E. Mahling and Noel Craven, From Office Automation to Intelligent Workflow systems, in IEEE Expert/Intelligent Systems & their Applications Vol. 10, No.3, June 1995.
Margolin'97	V. Margolin, Getting to know the user, Design Studies, Vol. 18 No. 3, July 1997.
Mars'94	Nicolaas J. I. Mars, An ontology of measurement, in Proceedings of the ECAI'94 Workshop Comparison of Implemented Ontologies, Amsterdam, the Netherlands, August 1994, Nicolaas J. I. Mars ed.
Matieson et al'99	K. Mathieson, M. Bhargava and M. Tanniru, Web-Based Consumer Decision Tools: Motivations and Constraints. Working Paper 1999-04, school of business administration Oakland University. http://www.sba.oakland.edu/workingpapers1999/1999-

	04/cdt_motivations.html
Mays et al.'91	Eric Mays, Robert Dionne and Robert Weida, K-Rep system overview, SIGART Bulletin 2, special issue on implemented knowledge representaion and reasoning systems.
Mays et al.'91	Eric Mays, Sitaram Lanka, Robert Dionne and Robert Weida, A persistent store for large shared knowledge bases, IEEE Transaction on Knowledge and Data Enginnering 3.
Mays'87	Eric Mays, Organizing knowledge in a complex financial domain, IEEE Expert 2, 61-70.
McDermott '82	John McDermott, R1: A Rule-Based Configurer of Computersystems. In Artificial Intelligence 19, 1982.
McDermott '88	John McDermott,, Preliminary Steps Toward a Taxonomy of Problem Solving Methods, in Automating Knowledge Acquisition for Expert Systems, Sandra Marcus ed., Kluwer Academic, Boston, 1988.
McGuiness & Wright'98	Deborah L. McGuinness and Jon R. Wright, An industrial-Strength Description-Logics-Based Configurator Platform, in IEEE Intelligent Systems & their applications, vol. 13, no. 4, July/August 1998.
McGuinness & Borgida'95	Deborah L. McGuinness and Alexander Borgida, Explaining subsumption in description logics, in proc. IJCAI'95, Morgan Kaufmann, 1995.
Mcguinness and Wright'98	An industrial strength description logics-based configuration platform, in ieee special issue on configuration
Mcguinness and Wright'98b	Conceptual Modeling for Configuration: A description Logic-based Approach. AT&T Research Report
Mcguinness et al.'95	Description logic in practice: A classic application. In proceedings of the international conference on artificial intelligence, Montreal, Canada.
Mcguinness'91	Deborah L. McGuiness, The CLASSIC knowledge representation system: implementation, applications and beyond, in Proceedings of the International Workshop on Terminological Logics, Schloß Dagstuhl, Germany. KIT-Report 89, Technische Universität Berlin, Germany.
McKendree & Carroll '86	J. McKendree and J. M. Carroll, Advising Roles of a Computer Consultant, in Marilyn Mantei and Peter Orbeton (Ed.), Human factors in Computing Systems: CHI'86 Conference Proceedings, ACM/SIGCHI, 1986.
Mefert'98	H. Meffert, Marketing: Grundlagen marktorientierter Unternehmensführung konzepte – Instrumente – Praxisbeispiele, 8. Auflage, Gabler, 1998.
Mentzing'80	D. Mentzing (Ed.), Frame Conceptions and Text Understanding, Berlin: de Gruyter, 1980.
Minsky '75	M. Minsky (Ed.), „A Framework for Representing Knowledge", in [Winston 75], 211-277.
Mittal & Fraymann'89	S. Mittal and F. Frayman, Towards a generic model of configuration tasks in proceedings 11th IJCAI, Detriot,

1989.

Mittal and Falkenhainer'90 — S. Mittal and B. Falkenhainer, Dynamic Constraint Satisfaction Problems, in proceedings of 8th national conference on artificial intelligence, AAAI Press, 1990.

Mittal and Arya'86 — "A Knowledge-Based Framework for Design Process" in Proceedings of the Fifth National Conf. on Artificial Intelligence, pp. 856-865, vol. 2, August 1986.

Moore'85 — Robert C. Moore, Semantical considerations on nonmonotonic logic, Artificial Intelltgence 25.

Morales'90 — "The Machine Learning Toolbox database" Deliverable 5.8, Machine Learning Toolbox ESPRIT project P2154, 1990.

Morik'89 — Katharina Morik, Sloppy Modelling, in Knowledge Representation and Organisation in Machine Learning, K. Morik ed., Springer Verlag, 1989.

Moser'83 — M. G. Moser, An overview of NIKL, the new implementation of KL-ONE, Bolt, Beranek and Newman, Inc., Technical Report 5421.

Murray-Smith & Thakar '93 — Roderick Murray-smith and Sunil Thakar, Case Storage and Adaptation using RBF Neural Nets, KI-93, Workshop on Machine Learning, Sept. 1993, Berlin, Germany.

Murray-Smith & Thakar' 93 — Roderick Murray-smith and Sunil Thakar, Combining case based reasoning with neural networks, AAAI-93, Workshop on AI in services and support: Bridging the Gap between Research and Application, July 1993, Washington D.C., USA.

Mylopulos'86 — John Mylopulos, On knowledge base management systems, in On knowledge base management systems: integrating Artificial Intelligence and Database technologies, Topics in Information systems edited by Michael L. Brodie, John Mylopulos and Joachim W Schmidt, Springer Verlag, New York, NY.

Nebel & Peltason'90 — Bernhard Nebel and Christof Peltason, Terminological reasoning and information management, Teschnische Universität Berlin, KIT-Report 85, 1990, Berlin, Germany.

Nebel & von Luck'87 — Bernhard Nebel and Kai von Luck, Issues of integration and balancing in hybrid knowledge representation systems, in Proceedings of the 11th German Workshop on Artificial Intelligence (GWAI-87), Geske, September 1987, Katharina Morik, ed., Springer-Verlag, Berlin.

Nebel'88 — Bernhard Nebel, Computational complexity of Terminological reasoning in BACK, Artificial Intelligence 34.

Nebel'90a — Bernhard Nebel, Terminological reasoning is inherently intractable, Artificial Intelligence 43..

Nebel'90b — Bernhard Nebel, Reasoning and revision in hybrid representation system, lecture notes in Artificial Intelligence; subseries of lecture notes in computer science, edited by Jörg Siekmann; volume 422, Spring-Verlag, Berlin.

Nebel'91 — Bernhard Nebel, Terminological Cycles: semantics and

computational properties, in principles of semantic networks: expolorations in the representation of knowledge, John F. Sowa, ed., The Morgan Kaufmann series in representation and reasoning, 1991, Morgan Kaufmann Publishers, Inc. San Mateo, CA.

Neches et al.'91 — Robert Neches, Richard E. Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Sentor & william R. Swartout, Enabling technology for knowledge sharing, AI Magazine 12, 1991.

Neighbors'84 — J.M. Neighbors. The draco approach to constructing software from reusable components. IEEE Transactions of Software Engineering, SE-10(5), September 1984.

Neighbours '80 — J.M. Neighbors. Software construction using components. Technical Report 160, Department of Information and Computer Sciences,University of California, Irvine, 1980.

Nonnenman'86 — Uwe Nonnenmann: Wissensgesteurte Lösungen des Konfigurationsproblems. Diplomarbeit, FB Informatik, TU Berlin, 1986.

Norman & Drapers '86 — D. A. Norman and S. W. Drapers (eds.) User Centered System Design: New Perspectives on Human Computer Interaction. Hillsdale, NJ: Lawerence Erlbaum Associates, Inc. 1986

Norman'93 — Donald A. Norman, Things That Make Us Smart: Defending the Age of the Machine, Addison Weley, New York, 19

Nua Internet Survey'98 — Nua Internet Survey 1998.

Nutt & Patel-Schneider'94 — Useful defaults in description logics, in Proceedings of the international workshop on Description Logics, Bonn, Germany May 1994, Franz Baader, Maurizio Lenzerini, Werner Nutt & Peter F. Patel-Schneider eds. DFKI, Kaiserslautern/Saarbrücken, Germany.

O'Keefe & McEachern'98 — R. O'Keefe and T. McEachern, Web-based customer Decision Support Systems. Communication of the ACM vol. 41, No. 3 1998.

Oertel & Perersohn'96 — Wolfgang Oertel and Uwe Petersohn, Hybrid Knowledge Organization within an Object Framework, in proceedings KI-96 Workshop WRKP, 1996, DFKI, Kaiserslautern, Germany.

Orsvärn & Axling'99 — Klas Orsvärn and Tomas Axling, The Tacton View of Configuration Tasks and Engines, in Proceedings AAAI'99 Workshop on Configuration, July 1999, Orlando, Florida, USA

Owsnicki-Klewe'88 — Configuration as a consistency maintenance task. Proc. GWAI-88.

Padgham & Lambrix'94 — Lin Padgham and Patrick Lambrix, A Framework for part-of hierarchies in terminological logics, in Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning, KR'94, May 1994, Bonn, Germany Jon Doyle, Erik Sandewall & Pietro Torasso, eds. Morgan Kaufmann Publishers, Inc. San

Franscisco, CA.

Padgham & Zhang'93 — Lin Padgham and Tingting Zhang, A terminological logic with defaults: a definition and an application, in Proceedings of the 13th International Joint Conference on Artificial Intelligence, Chambery, France, August 1993, Ruzena Bajcsy ed., Morgan Kaufmann Publishers, Inc. San Mateo, CA.

Padgham'94 — Lin Padgham, Systems vs. Theory vs.....: KR&R research methodologies, in Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning, KR'94, May 1994, Bonn, Germany Jon Doyle, Erik Sandewall & Pietro Torasso, eds. Morgan Kaufmann Publishers, Inc. San Franscisco, CA.

Patel-Schneider & Swartout'93 — P.F. Patel-Schneidern and William R. Swartout, Description-Logic knowledge representation system specification, Available from AI principles Research Department, AT&T Bell Labs, Murray Hill, NJ.

Patel-Schneider '84 — P.F. Patel-Schneider, Small can be beautiful in knowledge representation, in Proceedings IEEE workshop on Principles of Knowledge-Based Systems, Denver, CO. December 1984. IEEE Computer Society.

Patel-Schneider '87 — P.F. Patel-Schneider, Decidable, logic-based knowledge representation, Palo Alto Research, Technical Report No. 56, Palo Alto, CA.

Patel-Schneider '89 — P.F. Patel-Schneider, „Undecidability of subsumption in NIKL. Artificial intelligence, 39(2):263-272, 1989

Patel-Schneider '89 — P.F. Patel-Schneider, A four valued semantics for terminological logics. Artificial Intelligence, 38 (3): 319-351, 1989.

Patel-Schneider et al.'91 — P.F. Patel-Schneider, Deborah L. McGuinness, Ronald J. Brachman, Lori Alperin Resnick and Alexander Borgida, The CLASSIC knowledge Representation system: guiding principles and implementation rationale, SIGART Bulletin 2, special issue on implemented knowledge representation and reasoning systems.

Patmore & Renner'97 — A.B. Patmore and D. H. Renner, Closer to the Customer, Closer to the Goal: Customer Relationship Management Demands a Process-centered Approach, http://www.ac.com/overview/Outlook/over_2nov97.html

Pazzani&Billsus'97 — M. Pazzani and D. Billsus, Learning and revising user profiles: The identification of interesting web sites. Machine Learning, 27, 1997.

Peltason et al.,'87 — C. Peltason, K.v. Luck, B. Nebel and A. Schmiedel, The Users guide to the BACK-System, KIT-Report 42, Technische Universität Berlin 1987.

Peltason et al.,'89 — C. Peltason, A. Schmiedel, C. Kindermann, J. Quantz, The BACK system revisited, KIT-Report 75, Technische Universität Berlin, 1989.

Peltason et al.,'91 — C. Peltason, Kai von Luck & Carsten Kindermann, Proceedings of the Terminological Logic Users Workshop,

	Berlin, Germany, October 1991, Teschnische Universität Berlin, KIT-Report 95, Berlin, Germany.
Peltason'89	C. Peltason, Wissensrepräsentation für Entwurfsysteme: Die Behandlung von Klassifikation und Taxonomie.- Dissertation, FB Informatik, TU Berlin. 1989.
Peltason'91	C. Peltason, „The BACK System: An overview", in proceedings of the SIGART bulletin, vol. 2(3), pp. 114-119, 1991.
Puls'91	H. Puls, Entwicklung eines Modules zur Konsistenz überwachung für eine wissensakquisitionskomponente zum Einsatz innerhalb eines Beratungssystems für den Vertrieb von Fahrzeugen, Diplomarbeit FB Allgemeine Informatik, TFH Berlin, 1991.
Peppers & Rogers'93	D. Peppers and M. Rogers, The One to One Future, building relationship one customer at a time, New York.
Peppers & Rogers'97	D. Peppers and M. Rogers, Enterprise One to One, Tools for Competing in the Interactive Age, New York.
Perkins & Martin'86	D. Perkins and F. Fragile, Knowledge and Neglected Strategies in Novice Programmers. In E. Soloway & s. Iyengar (eds.),Empirical Studies of Programmers, Norwood, NJ: Ablex, 1986.
Phol'98	W. Pohl, Logic-Based Representation and Reasoning for User Modeling Shell Systems. Number 188 in Dissertation zur Künstlichen Intelligenz (DISKI), infix, st. Augustin.
Piller'98	F. Piller, Kundenindividuelle Massenproduktion:Mass Customization, http://www.wifak.uni-wuerzburg.de/mc/wasist.htm
Piller'98	F.T. Piller, Kundenindividuelle Massenproduktion: Die Wettbewerbsstrategie der Zukunft, Hanser.
Pine & Gilmore'98	Welcome to the Experience Economy, Harvard Business Review, July-August 1998.
Pine and Gilmore'99	B. J. Pine III and J. H. Gilmore, The Experience Economy: Work is Theatre & Every Business a Stage, Harvard Business school Press, Boston, Massachussetts, 1999.
Pine'93	B. J. Pine II, Mass Customization, Boston.
Poesio'87	Temporal Reasoning in a Hybrid System, WISBER Bericht Nr 21, Universität Hamburg.
Poesio'88a	The QUARK Reference Manual. WISBER Memo.
Poesio'88b	Dialog-Oriented A-Boxing. WISBER Bericht, Universität Hamburg.
Pollack et al. '82	M. E. Pollack, J. Hirschberg and B. Webber, User participation in the reasoning processes of expert systems, in Proceedings of the National Conference on Artificial Intelligence, 1982.
Puls'91	H. Puls, Entwicklung eines Modules zur Konsistenz überwachung für eine Wissensakquisitionskomponente zum Einsatz innerhalb eines Beratungssystems für den Vertrieb von Fahrzeugen, Diplomarbeit, FB Allgemeine Informatik, Technische Fachhochschule, Berlin, 1991.

Quantz & Royer'92	J. Joachim Quantz and Veronique Royer, A Preference semantics for defaults in terminological logics, in Proceedings of the Third International Conference on Knowledge Representaion and Reasoning (KR'92), Cambridge, MA, October 1992, Bernhard Nebel, Charles Rich and William R. Swartout eds., Morgan Kaufmann Publishers, Inc. San Mateo, CA.
Quantz & Schmitz'94	J. Joachim Quantz and Birte Schmitz, Knowledge-based disambiguation for machine translation, Minds and Machines 4, 1994.
Quantz et al.'94	J. Joachim Quantz, Birte Schmitz and Uwe Küssner, Using description logics for disambiguation in natural language processing, in Proceedings of the international workshop on Description Logics, Bonn, Germany May 1994, Franz Baader, Maurizio Lenzerini, Werner Nutt & Peter F. Patel-Schneider eds. DFKI, Kaiserslautern/Saarbrücken, Germany.
Quantz'92a	J. Joachim Quantz, How to fit generalized quatifiers into terminological logics, in Proceedings 10th European Conference on Artificial Intelligence, August 1992, Vienna, Austria, Bernd Neumann, ed., John Wiley, Chichester, UK.
Quantz'92b	J. Joachim Quantz, Semantische Repräsentation anaphorischer Bezüge in Terminologischen Logiken, Technische Universität berlin, KIT-Report 96, Berlin, Germany.
Quantz'93	An overview over Terminolgical Logics, working paper, TU-Berlin.
Quantz'94	An HPSG parser based on description logics, in Proceedings COLING'94.
Quantz'90	J. Quantz, Modeling and Reasoning with Defined Roles in BACK, KIT Report 84, Teschnische Universität Berlin, 1990.]
Quillian'68	M.R. Quillian, „Semantic Memory", in „Semantic information processing„ M. Minsky (Ed.), Cambridge (Mass.): MIT Press, 1968.
Rahmer & Voss'96	J. Rahmer and A. Voss, Case-Based Reasoning in the configuration of Telecooperation Systems, Configuration - Papers from the fall 1996 symp. Tech report FS-96-03, AAAI Press, 1996.
Rahmer et al.'99	Jörg Rahmer, Andreas Böhm, Heinz-Jürgen Müller Stefan Uellner, A Discussion of Internet Configuration Systems, proceedings AAAI'99 Workshop on Configuration , Sixteenth National Conference on Artificial Intelligence -- AAAI'99, July 18 - 22, 1999, Orlando, Florida.
Rahmer et al.'99	Jörg Rahmer, Andreas Böhm, Heinz-jürgen Müller, Stefan Uellner, A discussion of internet configuration systems, in Proceedings AAAI'99 Workshop on Configuration, July 1999, Orlando, Florida, USA
Ramachandran &	Knowledge Acquisition for configuration tasks: The

Gil'99	EXPECT Approach, in Proceedings AAAI'99 Workshop on Configuration, July 1999, Orlando, Florida, USA
Rasmussen'86	J. Rasmussen, Information Processing and Human Machine Interaction: An Approach to Cognitive Engineering. New York: North-Holland, 1986.
Rassmussen'86	J. Rassmussen, Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering. New York: Nort-Holland, 1986.
Reagle & Ackermann'99	Beyond Concern: understanding Net Users Attitudes about Online Privacy. AT&T Labs-Research Technical Report TR99.4.3, http://www.research.att.com/library/trs/TRs/99/99.4/99.4.3/report.htm
Reiter'80	Raymond Reiter, A logic for default reasoning, Artificial Intelligence 13, 1980.
Resnick et al.'87	D. E. Resnick, Mitchell C. M. and T Govindraj, An embedded computer-based training system for rihno robot operators. IEEE Control systems magazine, May 1987.
Resnick et al.'93	Lori Alperin Resnick, Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider and Kevin C. Zalondek, CLASSIC description and reference manual for the common lisp implementation: version 2.2, AT&T Bell Labs. Murray Hill, NJ.
Resnick et al.'94	Lori Alperin Resnick, Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider and Kevin C. Zalondek, CLASSIC description and reference manual for the common lisp implementation: version 2.2.1.1, AT&T Bell Labs. Murray Hill, NJ.
RFC 722	[RFC 722]
Rhea'92	D. Rhea, A new Perspective on Design: Focusing on Customer Experience, Design Management Journal, Fall 1992.
Rice'92	R.E. Rice, Task Analyzability, Use of New Media and Effectiveness: A Multi-Site Exploration of Media Richness, in Organisational Science, Vol. 3 No. 4, November 1992.
Rice'92	R.E. Rice, Task Analyzability, Use of New Media and Effectiveness: A Multi-Site Exploration of Media Richness, in Organisational Science, Vol. 3 No. 4, November 1992.
Roberts'92	Don D. Roberts, The existential graphs, in Semantic networks in artificial intelligence, Fritz Lehmann, ed., Modern Applied mathematics and computer science; edited by Y. Rodin; Volume 24, Pergamon Press, Oxford, England.
Robinson 1972	Definition, Oxford University Press, London, England, first edition: 1954
Rogers and Belloti'97	Y. Rogers and V. Belloti, Grounding blue-sky research: How can ethnography help? Interactions, May - June 1997.
Roland Berger'99sept.	Autokauf im Internet, Roland Berger Forschungs-Institut,

München, September 1999.

Roth et al.'87 — E Roth, K. Bennett and D. D. Woods, Human Interaction with an 'Intelligent' Machine, in International Journal of Man-Machine Studies, 27, 1987.

Rubash et al.'87 — "The use of an "Expert" to train salespeople" in Journal of Personal Selling & Sales Management, Vol. VII, (August 1987), pp. 49-55.

Sabin & Freuder '96 — D. Sabin and E. Freuder, Configuration as Composite Constraint Sarisfaction, Configuration-papers from the 1996 fall symposium, Tech Report FS-96-03, AAAI Press, 1996.

Sabin & Weigel'98 — Daniel Sabin & Rainer Weigel, Product Configuration Frameworks - A Survey, in IEEE Intelligent Systems & their applications, vol. 13, no. 4, July/August 1998.

Sabin and Freuder'97 — Daniel Sabin & Rainer Weigel, Modeling and Solving Configuration Tasks with ILOG solver, in proceedings of third international users meeting, ILOG optimization suite, ILOG, mountain view, California, 1997.

Sachs'95 — P. Sachs, Transforming Work: collaboration, Learning and Design. Communications of the ACM, 38(9).

Schallenmüller'99 — S. Schallenmüller, Multiple Customer Touch Points, http://www-wi.uni-muenster.de/wi/lehre/as/ss99/skripte/8-crm.pdf

Schallenmüller'99 — S. Schallenmüller, Multiple Customer Touch Points, http://www-wi.uni-muenster.de/wi/lehre/as/ss99/skripte/8-crm.pdf

Schank'90 — Schank, R. A. (1990). Tell Me a Story: A New Look at Real and Artificial Memory. New York: Charles Scribner's Sons.

Schild'91 — Klaus Schild, A correspondance theory for terminological logics: Preliminary report, in the proceedings of 12th International Joint conference on Artificial Intelligence, Sydney, Australia. August 1991, John Mylopoulos and Ray Reiter, eds. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Schmitz, Quant' 93 — Birte Schmitz & J. Joachim Quantz, Defaults in Machine Translation, Technische Universität Berlin, KIT-Report 106, Berlin, Germany

Schmolze '85 — James G. Schmolze, The language and Semantics of NIKL. Bolt Meranek and Newman Inc., Cambridge.

Schneider & Lederbogen'99 — B. Schneider and K. Lederbogen, Navigationskonzepte für Internet-Anwendungen, Information Management 1, 1999.

Schneider & Lederbogen'99 — B. Schneider and K. Lederbogen, Navigationskonzepte für Internet-Anwendungen, Information Management 1, 1999.

Schoch'69 — R. Schoch, der Verkaufsvorgang als sozialer Interaktionsprozess, Winterthur.

Schröder et al.'96 — Carsten Schröder, Ralf Möller and Carsten Lutz, A Partial Logical Reconstruction of PLAKON/KONWERK, in proceedings KI-96 Workshop WRKP, 1996, DFKI, Kaiserslautern, Germany.

Schumacher et al. '99 — Jürgen Schumacher, Wolfgang Wilke and Barry Smyth,

	Domain Independent Adaptations using Configuration Techniques, in Proceedings AAAI'99 Workshop on Configuration, July 1999, Orlando, Florida, USA
Searls and Norton'90	D. Searls and L. Norton, Logic Based configuration with a semantic network, Journal Logic Programming, vol. 8, nos 1-2, Jan.-Mar. 1990.
Seils'94	Oliver Seils, Modelierung einer Omnibus-Baureihe mit Hilfe eines Wissensrepräsentationssystems und Abschätzung seiner Leistungsfähighkeit, Diplomarbeit, WS 1993/1994, Technische Fach Hochschule Berlin.
Selfridge '91	Peter G. Selfridge, Knowledge Representation support for a sofware information system, in Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications, Miami Beach, FL, Feb. 1991.
Sellution White Paper	*Sellution White Paper*
Selz & Klein'98	The Changing Landscape of Auto Distribution, in proc. Of the 31 HICSS, Los Alamitos, CA. IEEE, 1998, Vol. VI.
Selz & Klein'98	The Changing Landscape of Auto Distribution, in proc. Of the 31 HICSS, Los Alamitos, CA. IEEE, 1998, Vol. VI.
Sharples et al'97	M. Sharples, N. Jeffery, D. Teather, B. Teather and G. duBoulay, A Socio-Cognitive Engineering Approach to the development of a Knowledge-based Training system for Neuroradiology, in proceedings of the World Conference on Artificial Intelligence in Education (AI-ED'97) , 1997.
Sharples et al'99]	M. Sharples, N. Jeffery, J.B.H. duBoulay, D. Teather, B. Teather and G.H. duBoulay, Socio-Cognitive Engineering: A Methodology for the Design of Human-Centred Computer Systems, in proccedings of the Human Centered Process conference (HCP'99), Brest, France, 1999.
Sharples'95	Mike Sharples, An Introduction to Human Computer Interaction, In Handbook of Perception and Cognition, Vol. 14:Computational Pscychology and Artificial Intelligence, Academic Press, 1995.
Shedroff'94	N. Shedroff, Information Interaction Design: A Unified Field Theory of Design, URL: http://www.nathan.com/thought/unified.html
Sleeman et al. 91	"Specification of Consultant-1" Deliverable 5.3, Machine Learning Toolbox ESPRIT project P2154, 1991.
Speel & Patel-Schneider '94b	Piet-Hein Speel and Peter F. Patel-Schneider, A whole-part extension for description logics, in the Proceedings of the ECAI'94 Workshop on Parts and Wholes: Conceptual Part Whole Relations and Formal Mereology, Amsterdam, The Netherelands, August 1994, Nicola Guarino, Simone Pribbenow & Laure Vieu eds.
Speel & Patel-Schneider'94a	Piet-Hein Speel and Peter F. Patel-Schneider, CLASSIC extended with whole-part extension for description logics, in Proceedings of the international workshop on Description Logics, Bonn, Germany May 1994, Franz Baader, Maurizio Lenzerini, Werner Nutt & Peter F. Patel-Schneider eds.

DFKI, Kaiserslautern/Saarbrücken, Germany.

Speel et al. 1991 — Piet-Hein Speel, Nicolaas J.I. Mars, Paul E. van der Vet, A knowledge based approach to semi-automatic indexing, in proceeedings of the workshop on Language and Information Processing, October 1991, Washington DC.

Speel'95 — Piet-Hein Speel, Selecting Knowledge Representation Systems, Thesis universiteit Twente Enschede 1995. ISBN: 90-9008086-4.

Strauss'99

Suchman'87 — L. A. Suchman, Plans and Situated actions: The problem of human-machine communication, Cambridge, England, cambridge University Press, 1987.

Swartout and Neches'86 — William R. Swartaut and Robert Neches, The shifting terminological space: an impediment to evolvability, in Proceedings Fifth National Conference on Artificial Intelligence, Philadelphia, PA, August 1986, Morgan Kaufmann Publishers, Los Altos, CA.

Tamura et al.'87/88 — "Knowledge-Based System Provides sales Support to Timber Products Brokers" in The journal of Computer Information Systems, Winter 1987/1988, pp. 24-27.

Tank & Thakar'90 — A Phase model for customer requirements analysis in the Proceedings of 2nd. German National Workshop Knowledge Engineering in Berlin FRG. 1990.

Teege '94 — Gunnar Teege, Using Description Logics in Intelligent Tutoring Systems, in proc. Int. Workshop on Description Logics (DL94), 1994.

Thakar '89 — Configuration: State of the Art Daimler-Benz as internal research report, 1989.Research & Technology, DaimlerBenz, Berelin, 1989.

Thakar '91 — Modeling "Sales Advisory" Domain in BACK and Some Problems Faced International Terminological Logics Users Workshop, Berlin, October 1991

Thakar '94 — Artificial Intelligence & Systems Engineering: A Case Study in Knowledge Modeling. Presented at AAAI'94 Workshop on Systems Engineering & AI, Seatle, Washington, USA. July 31st-4th August 1994.

Thakar '99 — Industrial experiences in empirical evaluation of domain modeling approaches . in Proceedings of the IJCAI'99, Workshop on Empirical Artificial Intelligence, Stockholm, Sweden , July 1999.

Timm & Rosewitz'98

Timmermans '99 — Patric Timmermans, The business Challenge of configuration, in Proceedings AAAI'99 Workshop on Configuration, July 1999, Orlando, Florida, USA

Timmers '98 — P. Timmers, Business Models for Electronic Markets, in international Journal for Electronic Markets, 8(2), 1998.

Tong'87 — Christopher Tong, Toward an Engineering Science of Knowledge-Based Design, in Artificial Intelligence in Engineering, Vol.2, No. 3. 1987.

Top & Akkermans'94 — Jan Tops and Hans Akkermans, Task and ontologies in

	engineering modelling, in International journal of Human-Computer studies, 1994.
Tsang 93	E. Tsank, Foundations of Constraint Satisfaction, Academic Press, London, 1993.
Wache & Kamp'96	Holger Wache and Gerd Kamp, Using Description Logic for Configuration Problems, in proceedings KI-96 Workshop WRKP, 1996, DFKI, Kaiserslautern, Germany.
Warner'99	F. Warner, Online auto dealers drive competition, in Wall Street Journal, http://www.zdnet.com/zdnn/stories/news/0,4586,2211336,00.html
Warner'99	F. Warner, Online auto dealers drive competition, in Wall Street Journal, http://www.zdnet.com/zdnn/stories/news/0,4586,2211336,00.html
Weiz'81	B. A. Weitz, Effectiveness in Sales Interactions: A Contingency Framework, in journal of Marketing Research, Winter 19981.
Weld'99	Daniel S. Weld, Recent Advances in AI Planning, AI Magazine 1999.
Werner Kohl '90	"Mit ExTel 1:0 in Führung" in Zeitschrift für Post und Telekommunikation, Jahr 1990, Heft 3, seiten 38-45.
Wilensky et al.'84	"Talking to UNIX in English: An Overview of UC" in Communications of the ACM, June 1984, Volume 27, Number 6, pp. 574-593.
Wixelbaum'93a	Elia S. Wixelbaum, C-CLASSIC Reference Manual, AT&T Bell Labs, Murray Hills, NJ.
Wixelbaum'93b	Elia S. Wixelbaum, C-CLASSIC 1.4 release notes, AT&T Bell Labs, Murray Hills, NJ.
Wolter & Scholz'96	Olaf Wolter and Uwe Scholz, The Necessity of Using Semantic Models for Configuration, in proceedings KI-96 Workshop WRKP, 1996, DFKI, Kaiserslautern, Germany.
Woods & Hollnagel'87	D: D. Woods and E. Hollnagel, Mapping Cognitive Demands in Complex Problem Solving Worlds. International Journal of Man-Machine Studies, 26, 1987 (special issue on Knowledge Acquisition for Knowledge Based Systems).
Woods & Roth'88	D. D. Woods and E. M. Roth, Aiding Human Performance II: From Cognitive Analysis to Support Systems, Le Travail Humain, 51(2), 1988
Woods & Roth'88	D. D. Woods and E. M. Roth, Cognitive Systems Engineering, In Handbook of Human-Computer Interaction, M. Helander(ed.), Elsevier Science Publishers B. V. (north-Holland), 1988.
Woods and Schmolze'92	William A. Woods and James G. Schmolze, The KL-ONE Family, in Semantic networks in artificial intelligence, Fritz Lehmann, ed., Modern Applied mathematics and computer science; edited by Y. Rodin; Volume 24, Pergamon Press, Oxford, England.
Woods et al. 88	D. D. Woods, E. M. Roth and K. B. Bennett, Explorations

in joint Human-Machine Cognitive Systems, in W. Zachary, S. Robertson & J Blach (eds.), Cognition, Computing and Cooperation. Norwood, NJ:Ablex Publishing, 1988.

Woods'75 — William A. Woods, What's in a Link, in : D. Bobrow, A collins (eds.), Representation and Understanding, Academic Press, New York (N.Y.) 1975 35-82.

Woods'83 — William A. Woods, What's important about knowledge representation?, IEEE Computer 16, 1983.

Woods'85 — William A. Woods, Understanding subsumption and taxonomy: a framework for progress, in Principles of semantic networks: explorations in the representations of knowledge, John F. Sowa, ed., The Morgan Kaufman series in representation and reasoning, Morgan Kaufmann Publishers, Inc., San Mateo, CA.

Wotruba'81 — T.R.Wotruba, Sales Management, Santa Monica.

WP5, 1990 — "Overview of Consultant-0", Deliverable 5.2a, Machine Learning Toolbox ESPRIT project P2154, 1990.

Wright et al. '93 — "A Knowledge-Based Configurator that supports Sales, Engineering, and Manufacturing at AT&T Network Systems", in: The proceedings of innovative applications of AI, Washington DC. 1993.

Wright et al. '95 — Conceptual Modeling Using Knowledge Representation: Configurator Applications, Proc. Artificial Intelligence in Distributed information Networks workshop, IJCAI-95, 1995.

Yen & Bonissone'90 — John Yen and P. Bonissone, Extending term subsumption systems for uncertainty management, in proceedings of the Sixth Conference on Uncertainty in Artrificial Intelligence, Cambridge, Ma.

Yen et al.'89 — Using terminologcal models to enhance the rule based paradigm, in: Proc. Second International Symposium on AI, October, 1989.

Yen'91 — John Yen, Generalizing term subsumption languages for fuzzy logic, in Proceedings of the 12th IJCAI, Sydney, Australia, August 1991, John Mylopoulus & ray Reiter eds. Morgan Kaufmann Publishers, Inc., Sam Mateo, Ca.

Yu & Skovgaard'98 — Bei Yu and Hans Jorgen Skovgaard, A configuration tool to increase product competitiveness, in IEEE Intelligent Systems & their applications, vol. 13, no. 4, July/August 1998.

Zeller'96 — Andreas Zeller, Software Configuration with Feature Logic, in proceedings KI-96 Workshop WRKP, 1996, DFKI, Kaiserslautern, Germany

Zisner & Henneman'88 — K. Zisner and R.L. Henneman, Development and evaluation of a model of human performance in large scale system. IEEE Transactions on Systems, Man and Cybernetics, 1988.

Its kind of fun to do the impossible

Walt Disney