N° 2009-ENAM-0029

Ecole Nationale Supérieure d'Arts et Métiers
-
Karlsruhe Institut für Technologie (KIT)

Laboratoire de Conception, Fabrication et Commande (LCFC)
Institut für Informationsmanagement im Ingenieurwesen (IMI)
Institut für Technische Mechanik (ITM)

Thesis

under an international cooperation agreement
to obtain the degree of

DOCTEUR de L'ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
AND
DOKTOR der INGENIEURWISSENSCHAFTEN des KARLSRUHE INSTITUT FÜR TECHNOLOGIE

Submitted to

the faculty of mechanical engineering of
the Karlsurhe Institute of Technology (KIT)
and to the Ecole Doctorale Sciences des Métiers de l'Ingénieur (ED 432)
at the Ecole Nationale Supérieure d'Arts et Métiers

DISSERTATION
By
Dipl.-Ing. Armin Azarian

A new modular framework for automatic diagnosis of fault, symptoms and causes applied to the automotive industry

---

Defended on the 25 November 2009 with the following jury:

M. Abdelaziz BOURAS, Professor, LIESP, University Lumière-Lyon II ..... Reviewer
M. Otto IANCU, Professor, University of Applied Sciences, Hochschule Karlsruhe..... Examiner
M. Benoît IUNG, Professor, Faculté des sciences et techniques, University of Nancy I .. Reviewer
M. Markus HOFFMANN, Sector Industry Automation Division, Siemens AG ..... Invited member
M. Patrick MARTIN, Professor, LCFC, Arts et Métiers ParisTech..... Examiner
M. Wolfgang SEEMANN, Professor, ITM, Karslruhe Institut für Technologie (KIT) ..... Examiner
M. Ali SIADAT, Associate Professor, LCFC, Arts et Métiers ParisTech ..... Examiner
M. François VERNADAT, Professor, University Paul Verlaine of Metz ..... Examiner

Arts et Métiers ParisTech, centre de Metz

Arts et Métiers ParisTech (Ecole Nationale Supérieure d'Arts et Métiers) est un Grand Etablissement dépendant du Ministère de l'Enseignement Supérieur et de la Recherche, composé de huit centres :
AIX-EN-PROVENCE ANGERS BORDEAUX CHÂLONS-EN-CHAMPAGNE CLUNY LILLE METZ PARIS

Preface of publisher

The rapid evolution, the diversity, complexity and the reduction of the life cycle of new generations of vehicles result in a higher degree of complexity to locate the cause of failures. Therefore, car manufacturers require methods and computer assisted diagnosis tools for their after sales networks. These new tools should allow controlling the complexity and reliability of current and future automotive systems and must capitalize the knowledge from the return of experience. Over the past few years Siemens developed the product SIDIS Enterprise which combines an information and diagnosis systems. The models for the diagnosis rely on three different information types: the components of the vehicle, the fault codes and the symptoms mentioned by the clients. The diagnosis is based on the acquisition of fault codes from the car's Electronic Control Units, to determine which component is defect. Moreover, the client can describe the symptoms of the car in order to give more information to the system. The diagnosis algorithm uses both sources of information to initialize the diagnosis session and localize the device which may be broken down. The diagnosis is based on expert-rules which are links issued from the perceived symptoms and fault code and pointing towards suspected diagnostic objects. The selection or acquisition of some fault codes or symptoms initiate the expert's rules which propagate the inspection of car entities which are modelled in the diagnostic tree. For the incrimination or discrimination of suspected objects, tests are performed. In general, the selection of symptoms speeds up the localization of the fault by means of the expert rules.
This thesis involves a set of research conducted on SIDIS Enterprise and has the objective to propose innovative computer aided diagnosis tools and methods to facilitate the fault localization and repair. These new methods will logically use simultaneously multiple sources of information: expert knowledge, construction knowledge, the return of experiences of previous diagnosis sessions, but they must also take into account the operational requirements of manufacturers as well as uncertainties.

Acknowledgment

This dissertation results from my work as doctoral candidate at the Siemens Corporation, department for automotive diagnosis in cooperation with the German University of Karlsruhe and the french engineering school Arts et Métiers ParisTech.

Special thanks go to my doctoral advisors from both institutions: Prof. Dr. Wolfgang Seemann and Prof. Dr. Patrick Martin, for their academic support and special encouragement as well as the professional collaboration.

I would like to thank my doctoral co-advisors, first Dr. Ali Siadat, for taking over the advisory opinion and for his excellent academic support and espessialy in the interest my work. I would like to thank my second doctoral advisor: Prof. Dr. Dr.-Ing. Jivka Ovtcharova for her support.

I am very thankfull for the interest of the reviewers: Prof. Benoît Iung and Prof. Abdelazis Bouras in my work and their remarks. Special thanks go to Prof. Otto Iancu who accepted to be the president of the Jury of my thesis.

I also thank the other members of my PhD committee: Prof. François Vernadat for his helpful scientific suggestions. Special thanks go to my colleague Markus Hoffmann for his proffesional collaboration during the project. And of course my thanks go to the Siemens Corporation for the financial support of this work as well as the Region Lorraine and the French-German University.

Special thanks go all the the members of my PhD committee for coming to my vocal examination in Metz and for their scientific remarks, suggestions and future carrer advice in general.

Table of contents

List of figures

List of tables

List of Abbreviations

.NET	Microsoft development platform
AAMA	American Automobile Manufacturers Association
ABS	Antilock Brake System
ACEA	Association des Constructeurs Européens d'Automobiles - European car manufacturers' association
ADAC	Allgemeiner Deutscher Automobilclub - German Automotive Association
AG	Aktiengeselschaft - Corporation
AR	Authorized Repairer
AS	Authoring System
ASAM	Association for Standardization of Automation and Measuring Systems
AT	Associated Term
AUDI	A car manufacturer
BCS	Brake Control System
BMW	A car manufacturer
BoT	Base of Terms
C#	Programming language
CAMS	Computerized Automotive Maintenance System
CBR	Case Base Reasoning
CEA	Commissaire à l'Energie Atomique – French commission of atomic energy
CMS	Content Management System
COM	Component Object Model
DaimlerChrysler	A car manufacturer
DAG	Directed Acyclic Graph
DAT	Deutsche Automobil Treuhand - German automotive market research company
DES	Diagnose Entwicklungs-System - Diagnostic development system
DMS	Dealer Management System
ECU	Electronic Control Unit
EOBD	European On-Board Diagnostics
EPA	Environmental Protection Agency
ESC	Electronic Stability Control
ESP	Electronic Stability Program
EU	European Union
FFL	Fixed Function List
FFP	Fixed Function Parameters
FMEA	Failure Mode and Effect Analysis
GM	A car manufacturer
GPL	General Public License
GT	General Term
GUI	Graphical User Interface
ID	Identifier
IDF	Inverse document frequency
IDS	Integrated Diagnostic System
IR	Independent Repairer / Information retrieval
IS	Information System

IT	Information Technology
LAN	Local Area Network
LRU	Least Replaceable Unit
Mb	Megabyte
MBD	Model Based Diagnostic
NP	Non-deterministic Polynomial time
OBD	On-Board Diagnosis
OCG	Oriented Causal Graph
ODX	Open Diagnostic data eXchange
PC	Personal Computer
PRE-SAFE	SAFEty of PREssure equipment (European Project)
R&D	Research and Development
RBF	Radial Basis Function
RENAULT	A car manufacturer
ROE	Return Of Experience
RPM	Revolutions Per Minute
SC	Short Circuit
SDA	Signal Diagnostic Agent
SIDIS	Service Information and Diagnostic Systems
SOA	Service Oriented Architecture
ST	Specific Term
SY	Synonyms
TF	Term Frequency
TMS	Translation Management System
UML	Universal Modelling Language
USA	United States of America
VDA	Vehicle Diagnostic Agent
VDI	Verein Deutscher Ingenieure – German association of engineers
VHL	Virtual Hierarchy Link
VIN	Vehicle Identification Number
VW	A car manufacturer
Web	World Wide Web
WLAN	Wireless Local Area Network
WS	Workshop System
XML	eXtensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Nomenclature

General notation	
$\| \|$	norm of a vector
$< \mid >$	Scalar product of two vectors
card { }	Cardinal of a set
Cst	A constant
E[ ]	Floor function
Max()	maximum function
Min()	minimum function
o	An offset
o()	Landau notation
tanh()	hyperbolic tangent function
w	A weight coefficient
$\overline{X}$	Average value of a discrete distribution *X*
α	A real number
β	A real number
γ	A real number
Diagnostic	
a_0	Weight of precision factor
a_1	Weight of test cost factor
a_2	Weight of life time factor
a_3	Weight of suspicion rules' partial moment
a_4	Weight of causal links' partial moment
Ab()	Predicate Abnormal
C	A conflict
CBR_i	Weight of a CBR case indexed by *i*
c_i	cost function
CLT	Weight of the life time factor
COMP	Set of components
d(x,y)	distance between two vectors *x* and *y*
Δ	A set of diagnosis
DO	Diagnostic Object
eff	efficiency
f_i	Subset of fault
$f_{SDA,i}$	i^{th} caracteristics of SDA
G	A graph
GC	Weight of a causal link
GDO	Weight for hierarchical links in the diagnostic tree
GS	Symptom weight
GT	Basis weight for explicit suspicion of diagnostic objects
H	A hitting set
h()	an heuristic function
H()	Entropy of a finite discrete probability distribution
H_α ()	Entropy of order α of a finite discrete probability distribution

Heaviside()	Heaviside function
HYP	A set of hypothesis
I_G	Giny impurity factor
K()	Objective function
Level()	Depth of a node in a tree
li	leaf node
LT	Lifetime
NB	Number of "is composed by" relations of a diagnostic object
NP(Ri)	Total number of premises of a Boolean rule Ri
NV	Total number of suspicion links (e.g. of a symptom or a premise)
NV(CBR)	Total number of suspect CBR cases
NV(Ri)	Total number of boolean rules
OBS	A set of observations
P()	Probability
ψ^-	A set of invisible observations
ψ^+	A set of visible observations
PM	Partial moment
PM_{LT}	Partial moment of life time factor
PR	Weight of a premise
R	New rank of suspicion rules
S	Signature vector
s_j	test to execute
SM	Suspicion moment
SM_α	Causal Moment
SM_f	Final Moment
SM_h	Heuristic moment
T	The set of nodes of a tree
Tc	Test cost
Tc_{childs}	The test costs of the children of a DO
W	An explanation set of a diagnostic problem
ODX Exchange System	
C	A case
Q	A query case
σ()	A comparator or similarity function
Information retrieval	
||Sj||	Length of symptom S_j
A	Set of relevant and returned symptoms
B	Set of relevant and non returned symptoms
C	Set of non relevant and non returned symptoms
D	Set of non relevant and returned symptoms
df_t	document frequency for term *t*
dis	distance
idf	Inverse document frequency
L	Acceptability limit
Out_j	Output of the j^{th} arc of the correspondence graph
PO	Popularity matrix
Pr	Precision
q	request vector

Q	set of queries
Re	Recall
$sim()$	Similarity measure
S_j	Level returned by node j in the correspondence graph
tf_{Sj}^{t}	term frequency of term t in symptom Sj
TFQ	Term frequency request vector
Θ	Confidence factor of an associated term or a synonym
$w(t,Sj)$	Weight of term t in symptom Sj
WQ	Weight request vector

A new modular framework for automatic diagnosis of fault, sympotms and causes applied to the automotive industry

DISSERTATION

By

Dipl.-Ing. Armin Azarian

Introduction

The automotive industry is currently facing a global change due to economic climate. One topicl challenge is meeting the specific demands of specific customers. This economic pressure leads to the development of a multitude of variants of the same vehicle model. For example: 550 000 BMW 3-Series' have been sold in 2007, and less than 1% are identical [BMW, 2008] due to the multitude of options. To distinguish their brand value, automotive manufacturers are forced to innovate heavily, especially in the field of safety, comfort, and the integration of information and communication technology. In this context about 90% of innovations are in the field of automotive electronics. Todays electronic systems represent 30% of the car's value, as opposed to 0.5% in the 1980's; and the trend is increasing. This results in more complex vehicles. More and more functionality depends on Electronic Control Units (ECU). Today a middle-class vehicle contains around 30 ECUs, in the luxury class around 70, with around 90Mb of Software monitoring them. According to Tannenberger (R&D director Electric/Electronic of VW) in [Brandl, 2008]
"*Since 6 or 7 years we experience a dramatic accession of electric / electronic systems in the automotive industry. We have painfully experienced that the organization not adequately match these requirements. The increasing relevance of Software in the ECUs could not be taken into the calculation without the support of the corresponding development process nor with the necessary competences. Exploding design cost and significant quality defects were the direct consequences of this excessive demand. […]. A key role is thereby the innovation. 90% of the innovations are based on electronics and software. […] If we take the wiring harness into account, there is barely something in the car which is not controlled by electronic*" [Brandl, 2008].

With automotive electronic systems the software, hardware and above all integration,pose new challenges for the manufacturer. One of the growing constraints is the reliability of vehicles. A study by the center of automotive research in Germany (ADAC) pointed out that in 1998 50.5% of car breakdowns have their origin in electronic failures; and in 2008 the percentage was up to 59.4% [ADAC, 2006]. The complexity of electronic failures and the pressure in terms of brand image leads naturally to the development of computer aided diagnostic tools for the after-sales network. The growing constrains of performance, user satisfaction and the augmented complexity of safety and comfort devices in vehicles drive a critical need for efficient diagnostic systems, to achieve the standards of quality demanded by the consumer. The complexity of engineering devices in the domain of automotive systems has lead to an increased demand for computer-supported behavior, prediction, diagnostic and testing tools.
The overall complexity of repair processes combined with the size of the car park tended to stimulate the demand in the car and service market over the period 1997-2004 [Chieux, Guillaneuf, 2005]. Other factors tended to dampen the demand such as the high reliability of vehicles [DAT, 2007], increased road safety and the wide prevalence of traffic control measures decreased the number of accidents. All these factors led to a changing landscape of workshops where computer assisted diagnostic tools support the technicians. A lot of multi-brand diagnostic tools have been developed for the independent repairer. For example in the European Union (EU), close to 6,000 Independent Repairer (IR) businesses were lost in the group of 12 countries of interest in just 2002-2003. While independent businesses vastly outnumber Authorized Repairers (AR), the ratio of IR to AR has fallen from 7 to 1 to 5 to 1 in the EU [ACEA, 2005]. There are a number of structural and institutional factors that confer competitive advantages on AR. They include [Bozon, 2005]:

- Consumer loyalty
- "Free" extended warranties
- Need for a complete service (e.g. with a new replacement vehicle in case of critical breakdowns)
- Financial advantage [DAT, 2007]
- Privileged access to the technical information of the manufacturer

The main trend is that IR's are losing market share to the AR's, except where they hold some contractual engagement with manufacturer [London Economics, 2006]. A study of the American Automobile Manufacturers Association (AAMA) [AAMA, 1998] suggests that the value of a new car to the after-sales service is around 300$ or an average percentage of 3% of the car's value per annum. Today, the race between global players is accelerating and **service becomes more and more one of the crucial links in the value chain** for car manufacturers. The European market, for the service and repair of motor vehicles, was worth approximately 100 billion Euros in 2004. This represents about one fifth of the market as a whole, and makes the service and repair segment the second largest sub-market in the car sector, after the new car sales sector [London Economics, 2006]. This trend illustrates why car manufacturer assign more and more importance to after-sales and are ready to invest in computer assisted car diagnostic systems for their worldwide network of authorized repairers.

In this context Siemens AG entered the car diagnostics market 20 years ago with their flagship product "Diagnose Entwicklungs-System" (DES) and during the past few years have been developing SIDIS Enterprise; a complete multi-brand car diagnostics solution and information system for manufacturers (detailed description in chapter I section 1.1.4).

The origin of the project of this thesis began with the development of a module in SIDIS Enterprise for off-board diagnostics. Off-board diagnostics aims at repair and, more specifically, replacement of components. In contrast to on-board diagnostics, the task is to localize the fault down to the smallest replaceable unit. Therefore, off-board diagnostics requires a different level of accuracy (granularity) for fault localization. Off-board diagnostics can make use of more observations, using advanced service testers and **human observations**. In order to **increase the quality and accuracy** of the off board diagnosis, a module was developed to research efficiently these human observation for the initialization of the diagnostic session. There are other difficulties in the guided fault finding of defective devices in cars. For example the deviation of fault codes in ECUs. The acquisition of the motor rotation speed from the engine control unit is used in many other subsystems like ABS (Anti-blocking system) and ESP (Electronic Stability Control). If one problem occurs in the engine control unit, 50 - 60 other fault codes occurs which are not related to the cause. Therefore finding the cause of a problem is nearly impossible without diagnostic tools and intelligent algorithms.

There are a lot of problems that need to be addressed for a computer assisted car diagnostics system. The following list outlines the major difficulties:

- **Variant problem**: In the automotive industry each vehicle consists of a number of variant subsystems and elements [Struss, 2003]. Only small parts of vehicles are strictly identical [Gigon et al., 2009]. A remedy to this problem could be to offer compositional modeling by the use of generic component models in the diagnostics system (for example from a structural description).

- **Phenomena from different physical domains**: The interaction of components from different physical domains (e.g. electrical, electronic, hydraulic, pneumatic, and mechanical) determines the overall behavior of a subsystem. Creating a model of components that can be used in a common framework where the diagnostic reasoning can be applied to the overall phenomena is one of the main problems in Model Based Diagnostics (MBD). The designing of a component oriented model library with reusable and aggregate models could answer this problem.

- **Dynamic subsystems**: Dynamic systems have internal states that depend on previous inputs and failures which may only be visible in a subset of the operating modes of the vehicle (e.g.: full acceleration at low temperatures) or they may only be visible at the transition of the operating modes of subsystems. In general the identification of this sort of failure needs complete quantitative models that the automotive industry cannot afford to develop. An answer to resolve these types of failures could be to combine heuristics and qualitative reasoning in a diagnostic engine.

- **Limited measurability**: Very few sensors are available in cars and the manufacturers try to keep this number minimal for obvious cost reasons. To compensate this limited sensorial visibility the use of models with fault modes can help the fault localization process (exoneration of components through refutation of all faults they can present). Fault models also approximately solve the problems caused by dynamic subsystems (by adjoining appropriate knowledge to the models).

- **Time of a diagnostic session**: This constitutes the main criteria against which to evaluate the efficiency of a diagnostic system. While the generation of suspects for the algorithm to process must have computational and memory requirements acceptable norms for present day computing hardware. One option might be the the pre-compiation of the most probable failures in a case base. With this method the solution retrieval should achieve short session times with minimal computational and memory use in the majority of cases.

- **Models for diagnosis**: This is one of the primary non-functional requirements: the cost to establish a model for the computer-based diagnosis. Each new piece of information has a cost (temperature pressure intervals, nominal values, tolerance, uncertainties…). It is clear that redundant information has to be avoided as well as incomplete information which can't be used to ensure result. In some cases the feedback loop can be used to capture the behavior of a device. Nevertheless, various formalisms have to be examined for the behavior represented. This has to be investigated in order to provide a deeper visibility into the fault and its various dynamic manifestations. The trend today in the industry is to minimize the cost related to the development of models [AAMA, 1999].

Following the trend of the last 5 years, the next major evolution in car manufacture will be in the areas of safety (e.g. ESP, ABS) and communication technologies. On one hand, safety equipment tends to be active and monitor in real-time the driving parameters. The active cruise control is one example of safety equipment which interferes with the vehicle's dynamic. In active safety equipment one innovation is "Car 2 Car communication" which will be promising in the future. Based on Wireless Local Area Network (WLAN) technologies the car can communicate with other cars within a distance of 300 meters. This is the idea behind the SAFEty of PREssure (PRE-SAFE) project [VDI, 2003].
Another trend is the X-By-Wire technology. Currently this technology is under development but shows promising advantages over mechanical and hydraulic connection based on cost, lower weight and the smaller physical dimension. The system architecture leads to close networked electronic devices and more interfaces. A lot of questions like the testability, safety, diagnostic ability, liability have to be addressed with this technology before it can go mainstream.

The aim of this dissertation is both to study theoretical concepts and to create a new diagnostic off-board methodology bridging theory and practice through analysis of the related business environment. Resulting from this approach this thesis will proceed in proposing a new practical engineering solution for the automotive industry.
Performing the diagnostic task (off-board) will be achieved by optimal information management (propagation and correlation among measurements and diagnostic model variables), including knowledge discovery and inductive learning. The latter being employed to complete the diagnostic models with the statistics provided via feedback protocols.
In the scope of this dissertation, the research was confronted with the following issues:

- **Modeling**: having to provide model libraries that cover an entire application domain and the important features of the components as well as the dependencies.
- **Diagnostic strategies**: covering different goals and conditions (for instance, off-board fault identification, fault localization based on customer complaints for component replacement).
- **Methodology**: providing criteria and guidelines for the choice of modeling formalisms and strategies (for example preparing the diagnostic data for a new system without any past experience) alongside those for the architecture of diagnostic systems.
- **Artificial intelligence**: providing criteria and metrics for the choice of automatic model completion or inferences and pattern recognition. Providing algorithms that simulate and

combine different types of reasoning. Proceeding with the automatic treatment of natural language from human observations.

In this context, this dissertation was targeted at the design of a complete diagnostic system which allows cost-effective fault localization, with tighter targets around performance and precision, while also providing either simplified or automatic data management. The proposed global framework for the diagnosis is based on the architecture of SIDIS Enterprise. Regarding the aforementioned requirements the following objectives have to be considered in the development of a new framework:

- **Objective 1**: Reducing the time of a diagnostic session.

One goal for improving of the diagnostic methodology is to provide more detailed and accurate information at the time the tests are performed. Often Tests incriminate or discriminate many vehicle components that are suspects, but the test coverage and reliability are never 100%. User effort with the software should also be limited, and if the diagnostic methodology allows it, the resulting candidates should be explained to the users on the screen. The coverage should also be adequate despite the number of physical domains implicated in vehicles.

- **Objective 2**: Improve the quality of off-board diagnostics.

An investigation of automatic learning and knowledge discovery methods may provide algorithms that could address this issue. A central feedback server could collect the protocols of AR and capitalize on their evolving experience.

- **Objective 3**: Reduce the cost of model development.

This objective is clearly identified by car manufacturers, that the cost of post-development models for diagnostics should be minimized.

This dissertation proposes an operational framework for automotive off-board diagnosis.

- Chapter I section 2 details SIDIS Enterprise and the knowledge structure and networks this entails. In particular the data update management and the data editing processes for diagnostic related knowledge. Other aspects like the current diagnostic algorithm in SIDIS Enterprise and the protocol engine will be examined briefly with these elements being investigated to a deeper level later in this dissertation.

- Chapter II looks at automatic data completion in the knowledge structures of SIDIS Enterprise. On the one hand it details the management of qualitative description of failures. Qualitative descriptions reflect the nature of available observations, e.g. "engine does not start in winter ". In some cases it appears not to be relevant to reason in terms of the actual values of quantities. Rather, it can be sufficient to reason in terms of (qualitative) deviations from nominal values only. In addition to distinctions expressed in the quantity space of variables, it also turned out necessary to distinguish the different orders of magnitude of the relevant physical quantities. However, due to the amount of qualitative failure descriptions called perceived symptoms (around 1700) technicians neglect the selection of these observations at the initialization of the diagnosis session. But these symptoms are a useful source of information and may become more important with the emergence of X-by-Wire technology. In order to initialize the diagnosis session with more information, a module has been developed to retrieve the perceived symptoms automatically and thereby speed up the diagnostic session [Azarian, Siadat, 2008].
  This chapter also examines intelligent data management of ECU description files. One of the most important issues with vehicle diagnostics is to deal with the enormous variety of ECU variants and the vast array of different vehicle configurations (15.000 ECUs and around 800 vehicles variant). Despite the complexity inherent in processing this information, there is

currently no option other than the manual import of the data, in its entirety, into the SIDIS Enterprise database by its authors. Considering the potential time and cost savings, a method to automate this import of vehicle description data based on the new ASAM-ODX standard is also developed in this dissertation [Cremmel et al., 2008].

- Chapter III proposes a new diagnostic strategy which is still based on inference rules. One main drawback of inference rules is that they are difficult to debug and it is difficult for a human operator to keep an overview of the whole system. As detailed earlier tests incriminate or discriminate many vehicle components that are suspects, but the test coverage of all possible failures modes is never 100%.
  User effort with the software should also be limited, and if the diagnostic methodology allows it, the generate candidates should be explained to the users on the screen. The coverage should also be adequate despite the number of physical domains implicated in vehicles.
  Another area of investigation in this chapter, one that might yield a strong operating advantage is that of the extendibility of expert systems and optimization methods with soft constraints (e.g. On time factor and uncertainty).
  In accordance with the objective of cost reduction for the post-development of models, this chapter also investigates methods and tools for the analysis of feedback from workshops. For a car manufacturer like BMW, who have 2500 workshops in Europe [IP/06/302, 2006], the protocols of diagnostic sessions can reach 10,000 per day. This therefore represents a valuable source of information. The investigation in this area explores the inductive learning of parameters that speed up the diagnostic process, such as test cost or component lifetime [Gigon et al., 2009].

- Chapter IV presents a **framework for the diagnosis** which on one hand allows **bridging between the model based diagnostic world and traditional expert system**. At the same time, it makes use of the knowledge from previous diagnostic sessions produced by the after sales network with the help of inductive trees and Bayesian networks. This combination of knowledge makes it possible to increase the quality of a diagnostic session, to reduce the number of unnecessary tests and to auto-complete the missing information and parameters allowing a reduction of the cost of database population. This framework will be examined, checked and validated through 3 real use cases.

- Finally a conclusion contains the summary of the essential results obtained in the dissertation, along with an outlook on how this new framework might be extended. Similarly a view is given on further changes of basic parameters in the automotive after-sale industry and on future trends in automotive electronics. Following on from this related variants and possible trends are outlined as well as future challenges. The chapter and dissertation work ends with a general conclusion on the design of diagnosis.

Chapter I: Computer aided diagnostic tools and improvement requests

1 Introduction

Across the life cycle of a vehicle, manufacturers are confronted by several important challenges. Examples of these might be; demand for innovation, product and process complexity and the management of several types of risk. Additionally, quality requirements are increasing alongside legal restrictions concerning emissions and media pressure in this highly competitive market. With automotive electronic components in particular, car manufacturers find themselves in competition to provide the best quality to the customer and to ensure the functionality of highly complex electronic systems and components.

1.1 Current situation

In recent years there has been a marked increase in vehicle complexity with the growing 'intelligence' of technology playing the major part. At the same time customer expectations on quality have not lowered requiring the manufacturer to meet the same quality goal with a significantly more complex product. This problem is compounded with shorter product lifecycles and engineering time phases making reliability and maintenance even more difficult.
This section looks at the electronic technology in today's vehicles and the need for computer aided diagnostic systems to speed up vehicle maintenance. Afterwards some diagnostic tools are presented alongside SIDIS Enterprise.

1.1.1 Automotive electronic technology

In the industry at the moment the basic electronic architecture consists of a voltage source, sensor actuators and electronic control units (ECUs) containing software and hardware; as demonstrated in Figure I-1 [Bauer, 2003]. The ECU plays a central role in the vehicle architecture and typically consists of a standard micro-controller, specific signal processing and a controller integrated circuit as well as the necessary packaging, power supply, and power electronic components [Bortolazzi, 1996]. Sensors and actuators build the interface between the vehicle with its complex brake, drive and chassis functions and the mostly digital ECUs as processing units. Sensors convert a physical or chemical value into an electrical value possibly over non electrical intermediate stages [Bauer, 2003].

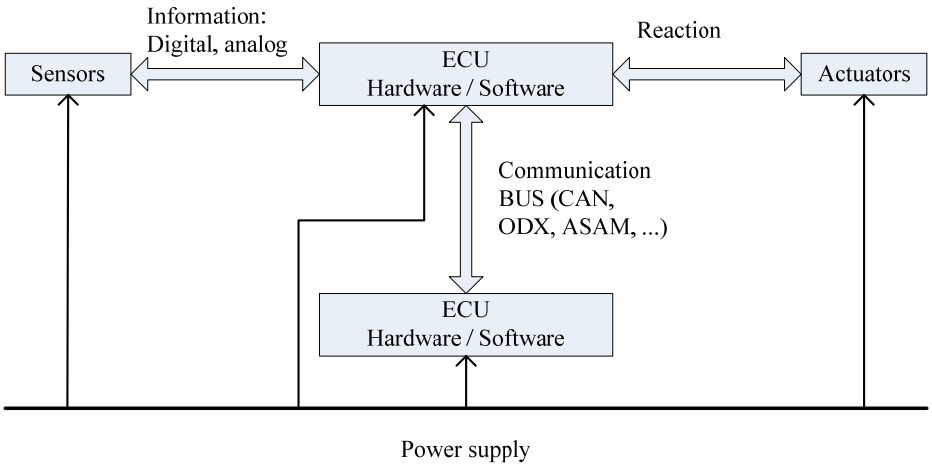

Figure I-1: Automotive electronic

As seen in Figure I-1, actuators provide the interface between the electronic signal processing (information processing) in an ECU and the related reaction. These signal converters combined with amplifier elements use the physical conversion principles between different energy forms, i.e. electric, mechanical, fluid and thermal [Bauer, 2003]. Examples of actuators are the electric motors for seats, power windows and electronic supported servo steering. The coherent operation, considering functionality, safety and comfort, of the high number of sub-systems in a vehicle can only be achieved through highly developed control systems [Braess, Steiffert, 2003].
A good example for illustrating the domain of vehicular elictronics is the brake control system (BCS).
A vehicle will progress based on the forces applied to it, these are a combination of the forces directed by the intentions of the driver and environmental forces. Examples of driver directed forces might be:

- Angle of wheels based on angle of steering wheel.
- Drive power from transmission based on position of pedals/gearstick.
- Braking resistance based on position of brake pedal.

Examples of environmental forces might be:

- Angular momentum of vehicle based on mass and rotation.
- Friction of wheals.
- Air resistance.

Where sudden changes occur in environmental forces (eg patch of ice), or the intentions of the driver exceed the physical boundaries of the system (eg excessive braking in a turn) there can be a mismatch in the actual behavior of the vehicle as dictated by the forces applied and the desired behavior of the vehicle as indicated by the driver. BCS seeks to address this behavioural gap though focusing the braking fores on each individual wheel.
To make these interventions possible an extended sensor network needs to identify the driver intentions by measuring things like steering angle and pedal position. Micromechanical torsion rates are taken into account in monitoring elements such as brake pedal position of wheel rotation speeds. The ECU for this system is also connected to the engine control unit and the transmission.
Via this extended sensor network the external forces applying to the vehicle can be evaluated along with the intentions of the driver. Where these exceed defined thresholds the BCS ECU can take corrective action by regulation of valves at each wheal changing individual braking pressures and thus changing the movement and resistance forces applying to the vehicle. Drive moment can also be altered via the connections to the transmission and engine control unit. In this manner can optimal vehical stability be realized [Liebemann et al., 2004].

The following components and parameters are involved in the brake control system:

- 8 driving parameters are monitored: wheel rotation speed, micromechanical torsion, steering angle, driving and break pedal position.
- 2 ECUs communicate with the BCS: Engine and transmission control units.
- 4 actuators are regulated: the valve regulating the breaking pressure at each wheel.

This example outlines briefly the dependencies between the automotive electronic subsystems for what is now standard safety equipment and the recognition of the driver's intentions from simultaneous sensors readings.

1.1.2 Automotive electronic complexity

In recent years electrical and electronic system components and accessories in automobiles have steadily increased, both in complexity and quantity. The need for innovations in comfort, convenience, entertainment, safety, security, communication and environmental concerns requires improved electronic systems such as BCS, ABS, and ESP for example. The past four decades have witnessed an exponential increase in the number, and sophistication, of electronic systems in vehicles. In 1977, the value of electronics systems and silicon components such as transistors microprocessors and diodes in motor vehicles averaged $110 per vehicle, while in 2001 it had increased to $1800 per vehicle [Leen, Heffernan, 2002]. Today's top of the range vehicle may have more than 4km of wiring and around 70 ECUs, see Figure I-2. Reducing this wiring mass through virtual network technologies in vehicles brings an explosion of dependencies between components.

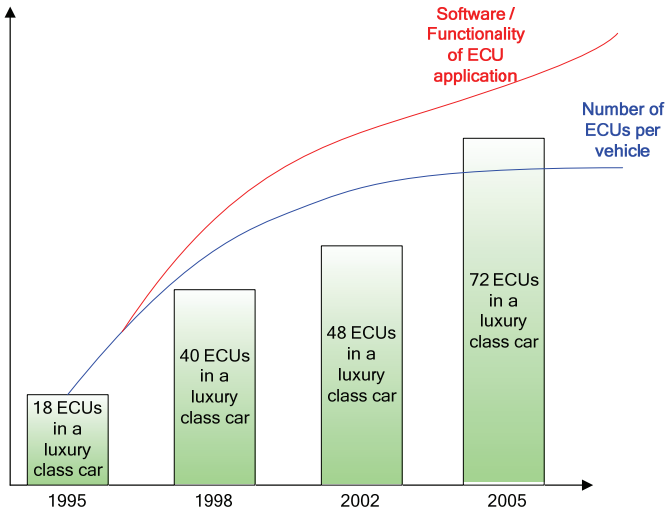

Figure I-2: ECUs in cars [DaimlerChrysler, 2004]

Vehicles, in many ways, are becoming more and more like personal computers, creating the potential to think more in terms of hosting plug and play devices. Today, nearly 30% of the total vehicle cost is in the automotive electronics. Further the introduction of multimedia and telematics to vehicles provides connectivity and entertainment for the customer and implies a huge market for automotive companies. The car is becoming an integral part of automotive and telecommunication services, giving access to individual traffic and navigation information, breakdown assistance, automatic emergency calls and traffic control. The car today contains sophisticated IT systems and the use of semiconductors and sensors continues to grow dramatically.

The functionality of electronic components is becoming more complex at an accelerating rate. Nowadays about one third of development costs in the automotive industry are spent on electronic developments, and this continues to increase [Weber, Weisbrod, 2002]. More importantly though the functionality and software content per ECU will increase at an even faster rate than the number of ECUs per vehicle. These trends point towards the number and the value of automotive electronics systems growing at a rate of 10% per year [Broy, 2003]. Another tendency that confirms the growing part that electronic play in vehicle is the size of the microcontroller market. According to a report from [Frost, Sullivan, 2008], the automotive microcontroller market should grow at an average of 13% between 2006 and 2010 because of the increasing demand for safer and more comfortable vehicles, in turn due to safety and comfort becoming more and more essential.

1.1.3 Automotive electronic reliability

A study by the center of automotive research in Germany (ADAC) [ADAC, 2005] pointed out that in 1998 50.5% of car breakdowns have their origin in electronic failures; and in 2004 the percentage increased to 59.4% (for vehicles between 3-5 years old, cf. Figure I-3).

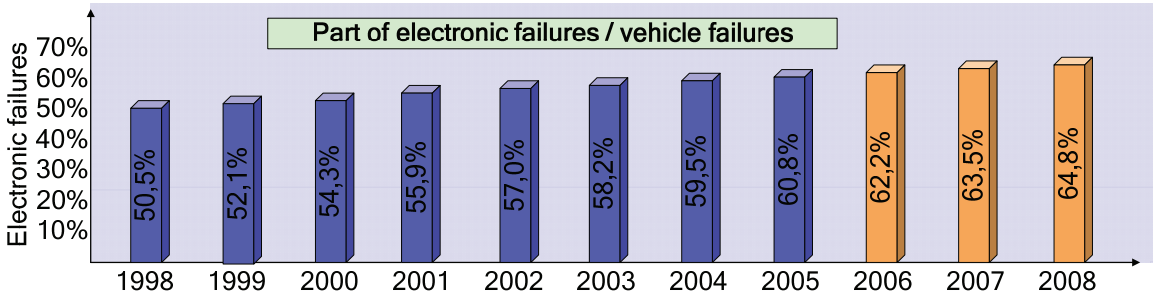

Figure I-3: Statistics of electronics failures [ADAC, 2005]

To distinguish the brand value, automotive manufacturers have to spend their efforts in innovation especially in the field of safety, comfort as well as the integration of information and communication technology (see chapter I section 1.1.2). In this context about 90% of innovations are in the field of automotive electronics. Each subsystem has become more efficient, accurately managing more parameters. However, more parameters increase the complexity of the system due to the high number of possible links between components and thus increase the risk of breakdowns. According to Tannenberger (R&D director Electric/Electronic of VW) in [Brandl, 2008]: "*Since 6 or 7 years we experience a dramatic accession of electric / electronic systems in the automotive industry […] The increasing relevance of Software in the ECUs could not be taken into the calculation without the support of the corresponding development process nor with the necessary competences. Exploding design cost and significant quality defects were the direct consequences*".

These quality defects are causing significant damage to the brand image. For example October the 3rd in 2004 a driver reported falling victim to a defect of the speed regulation system. He became stuck at a speed of around 200km/h for a distance of 150km with a Renault Velsatis [Liberation, 2004]. The problem was published across all Medias and for the customer it incriminates the whole brand. Therefore manufacturer after spending so much in innovation now also needs to take the after-sales service into account. Thus this trend of increasing electronic defects also has an impact on the landscape of workshops in the after sales service.

1.1.4 Automotive after-sales

As previously mentioned, the European market, for the service and repair of motor vehicles, was worth approximately 100 billion Euros in 2004. This represents about one fifth of the market as a whole, and makes the service and repair segment the second largest sub-market in the car sector, after the new car sales sector [London Economics, 2006].
For example in the European Union (EU), close to 6,000 Independent Repairer (IR) businesses were lost in the group of 12 countries of interest in just 2002-2003. While independent businesses vastly outnumber Authorized Repairers (AR), the ratio of IR to AR has fallen from 7 to 1 to 5 to 1 in the EU [ACEA, 2005]. There are a number of structural and institutional factors that confer competitive advantages on AR. They include [Bozon, 2005]:

- Consumer loyalty
- "Free" extended warranties
- Need for a complete service (e.g. with a new replacement vehicle in case of critical breakdowns)
- Financial advantage [DAT, 2007]
- Privileged access to the technical information of the manufacturer

The main trend is that IR's are losing market share to the AR's, except where they hold some contractual engagement with manufacturer [London Economics, 2006]. A study of the American Automobile Manufacturers Association (AAMA) [AAMA, 1998] suggests that the value of a new car to the after-sales service is around 300$ or an average percentage of 3% of the car's value per annum.

Through the initial period of the automobile industry diagnosis was performed manually. From there measurement become the standard means for diagnosis in the workshop as the tools to allow this became available. For example the stroboscope was introduced to determine ignition time in the middle of the 20th century. By the 1960's exhaust measurements became a common method of diagnosis. However, it was not until the first legislative regulations regarding On-Board Dagnostics (OBD) introduced by California Air resource Board and later by the federal Environmental Protection Agency (EPA) that modern off board diagnostic tools started to take shape. In 1990, the Environmental Protection Agency in the USA estimated that 60% of the total tailpipe hydro-carbon emissions from light-duty vehicles originated from 20% of vehicles with seriously malfunctioning emission control systems. Terefore it was important that such faults were detected early thus allowing repairs to be made as quickly as possible. [Kien, Nielsen, 2005]

Actually the vehicles are equipped with very sophisticated electronic devices and the networking of ECUs changed the occupation of technicians in the workshops. For example the motor rotation speed parameter is monitored by the motor management ECU but also by the ABS and ESP functionalities, and a faulty reading of this value causes around 60 fault codes in different ECUs. The overall complexity of repair processes combined with the increasing size of the car park tend to stimulate the demand in the car and service market over the period 1997-2004 [Chieux, Guillaneuf, 2005]. Other factors tend to dampen the demand such as the increasing reliability of vehicles [DAT, 2007], increased road safety and wide prevalence of traffic control measures decreased the number of accidents. All these factors lead to a changing landscape of workshops where computer assisted diagnosis tools support the technicians from simple maintenance to complex repair operation where evaluation and interpretation of failure codes play a central role.

An example of an early computer assisted diagnosis tool (middle 80's) in the automotive domain is the General Motors CAMS (Computerized Automotive Maintenance System) [William, 2003]. Although the system discussed here is essentially obsolete; it is at least representative of the level of diagnosis. The GM-CAMS was capable of detecting, analyzing and isolating faults in GM's vehicle equipped with digital engine control system. This system (commonly called technician's terminal) has now modern equivalent which are still relying on expert systems technology. [William, 2003]. Other commercial products like the XR25 for Renault were launched in 1985 that include a vast majority of vehicle. Peugeot and Citroen equip their networks with the platforms TEP92 and the ELIT in 1989 which became later Diag2000 and Lexia in 1998 (and later Peugeot Planet 2000 and Lexia-3 in 2003).

This trend illustrates that car manufacturers accord more and more importance to the part after-sales in the value chain and are ready to invest in computer assisted tools for the car diagnosis for their worldwide network of authorized repairers. As an example of the importance of after-sale service in the value chain of automotive manufacturers is Toyota with the model Lexus. Despite the few electronic devices and comfort equipment of this model, Toyota entered the car luxury market with this model. The reason of this success is that the car is sold with an important service contract which engages the manufacturer to do all the revision, repairs etc.
In this context Siemens AG had entered the car diagnosis market 20 years ago with their flagship product DES and since few years they are developing SIDIS Enterprise. This last product is a complete multi-brand car diagnosis solution for manufacturers. The main features of SIDIS Enterprise, which will be deeper investigated in this dissertation, are presented in section I.2; other features and functional modules are reported in appendix A section 1.1.

2 SIDIS Enterprise: system overview

The following section introduces the constitution and basic concepts of SIDIS Enterprise. This part is focused on the software architecture and usual user scenarios for car manufacturer. This chapter outlines the following list of topics:

- Overview and basic concepts: This sub-section describes the technology and software architecture of SIDIS Enterprise as well as usual deployment scenarios or car manufacturers.
- Data structure: This sub-section details the relevant data related to the diagnosis knowledge. The publication process which allows updating the vehicle data will also be examined.
- Workshop modules: This part is dedicated to the functional module used in the workshops. Here only the modules for the guided fault finding are presented including the diagnosis algorithm of SIDIS Enterprise.

The last section concludes with a synthesis of SIDIS Enterprise capabilities and gives and discusses the main drawbacks.

2.1 Architecture and main concepts

SIDIS Enterprise is a modular system [Ripplinger, 2006] divided in two essential basic components:

- The Authoring System (AS): allows an efficient editing, update, versioning of the whole vehicle data and service documents for the workshops.
- The Workshop system (WS) is used by mechanics in the workshop for vehicle diagnosis and to query and program ECUs; the complete service documentation can also be called up by context on the workshop system.

All system components operate on the basis of a common information model as depicted in Figure I-4. Both components have a client/server architecture, which means that for example authors can work in parallel via different clients and edit the same databases; the WS can also operate in client/server mode which is dedicated for large workshops, were a central server updates his database from the manufacturers and transmits all necessary information to the clients which can realize the diagnosis on cars as depicted in Figure I-4.

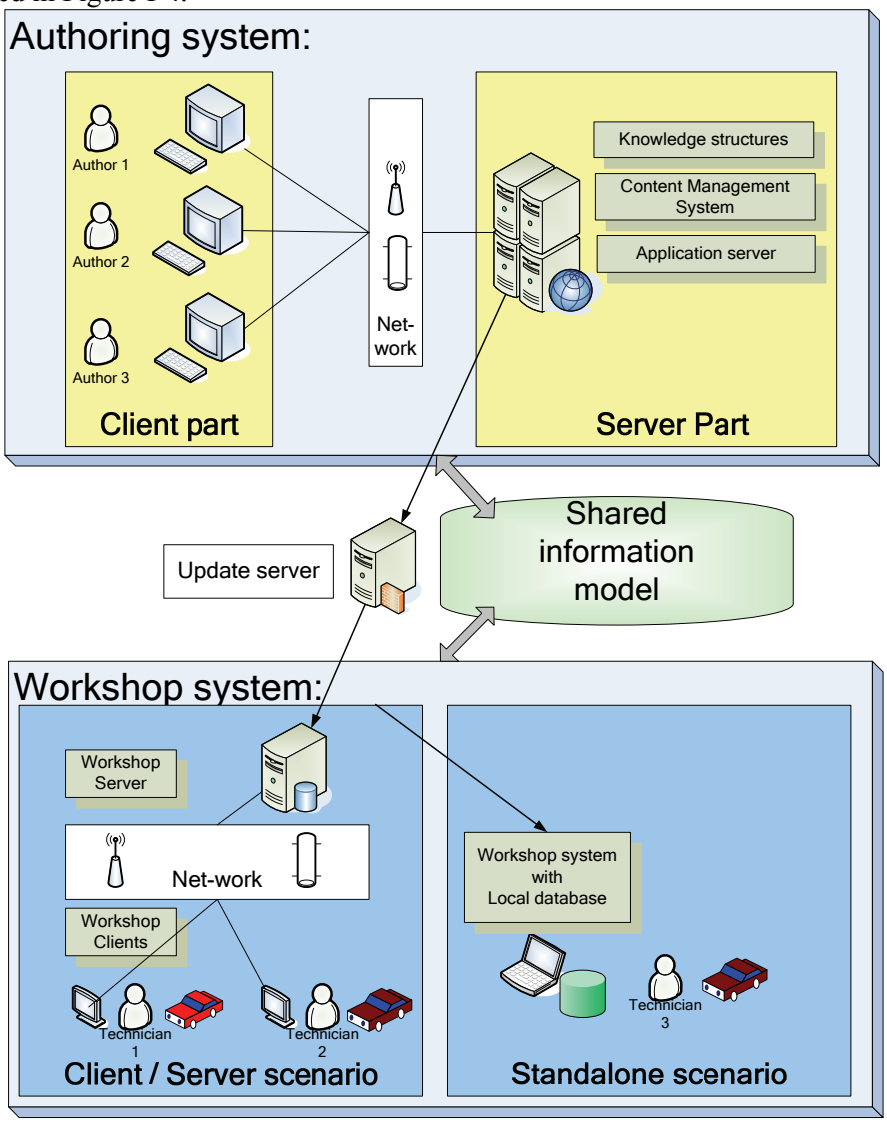

Figure I-4: SIDIS Enterprise main components overview

The SIDIS Enterprise Workshop system forms the core of a modern, future-orientated workshop information system. Its aim is to adapt the flow of work in the workshop to suit each task using a modular, sub-divisible system. To do so, SIDIS Enterprise provides a large number of independent function modules which are linked to individual work stations as a factor of workflow need and the application involved. This flexibility facilitates fast adaptation to new boundary conditions without costly implementation work, even after the task has begun. In this way, SIDIS Enterprise represents a platform for future expansion, with each new functional module building on the already existing modules.

The compilation and maintenance of the necessary vehicle data is a significant factor when it comes to operating such a system. That is why the SIDIS Enterprise Authoring system contains comprehensive options for efficient information gathering:
It supports distributed and simultaneous tasks at multiple locations using a shared database. The usual rights and privileges concept ensures that only authorized authors can make changes to the data assigned to them.

- The system is object-orientated and provides fast access to all structure, content and language levels. Information modules are user-friendly and immediately recognizable thanks to graphic interfaces arranged in a hierarchical tree structure.
- A linkage (Appendix A section 1.2) concept supports the reuse of structures and content. This also allows service documents to be associated with vehicle data.
- The flexible publication (Appendix A section 1.3) mechanism allows targeted publication and transmission of relevant data to the workshop system. Changes to the previous version are taken into account here in order to keep the amount of data to be transferred to a minimum.
- The translation of vehicle data into other languages is supported by XML-based export and import functions (e.g. for the connection with an external translation management system).
- The information model in the authoring system is sent to the workshop systems by an efficient distribution and updating mechanism (Appendix A section 1.3). This can also ensure that individual workshops receive only the parts of the database relevant to them.

The modular and service-orientated architecture of SIDIS Enterprise has, means it is possible to create a wide range of deployment scenarios. Application areas range from a standalone work station where a workshop server (dedicated PC in the workshop) is used, to access via an Internet portal. In all instances, the best means of distribution of software modules is from the central authoring system. The SIDIS Enterprise workshop information system consists of standard components and/or applications which are used by different customers without any change to their core functionality. These components are constantly updated and refined to that they always represent the current state of the art and knowledge. SIDIS Enterprise provides basic functions and system services within the functional modules. The associated user interface is specific to each manufacturer. This process allows precise matching of the user guide and interface appearance to the requirements of the manufacturer.

2.2 Information model

The SIDIS Enterprise information model is constructed and administered on the basis of an integrated Content Management Systems (CMS). It consists in a hierarchical classification of the information in tree structures:

- The term “information” here means all diagnostic and service data in general.
- Each node represents a structural and/or information building block. These can be linked to texts, images, ECUs or a single title.

The information model decides which nodes can be created. The appearance and behavior of a node can be configured differently. Links can be configured between individual nodes and trees. Links have different meanings, e.g.:

- To illustrate reference and relationships between vehicle components
- To refer to additional information, e.g. to a schematic of a controller, etc.
- Link between a device and its associated documentation

Each node can have configurable meta-information. For example, the relevant area of a vehicle (e.g. engine) and the particular type (e.g. repair manual) can reference a service document. The user can search for information based on these criteria within the workshop system. A typical search query could be “Display all repair manuals for Engine”.
This information model can be calculated, supplemented and expanded via a graphic interface, see Figure I-5.

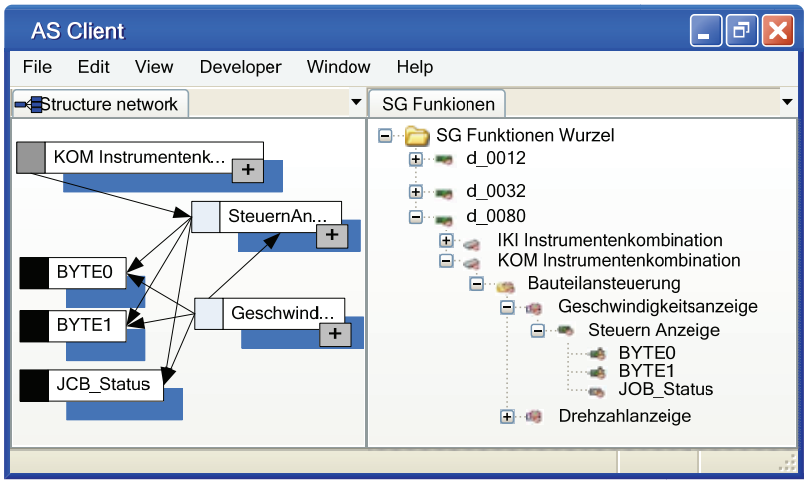

Figure I-5: Example of the GUI of a client of the AS

The high degree of flexibility derives from several technical factors:

- All content and data are stored in a structured pattern.
- Classes are used to define the structure of the pattern. Because of their class affiliation, all nodes and links are assigned predetermined structural characteristics, which are also inherited.
- Nodes and links can be supplied with attributes and metadata values.
- The overall structure of the information forms a hierarchy. This allows values from superior objects to be transferred to subordinate objects alike the inheritance mechanisms in object oriented programming.

Basically, all information in the CMS can be derived from two general classes: the BaseLinkClass and the BaseNodeClass. Figure I-6 gives an example of a possible information model for the implementation of diagnostic relative knowledge structures. The information model of these structures can be changed depending on the specific constraints of a manufacturer, for example if specific meta-data has to be associated to a node (e.g. the failure rate for a diagnostic object). All the information are derived from both classes depicted in the top of Figure I-6, then specific classes are derived from them. Here, the class suspicion link is an intermediate model that generalizes the class suspicion from fault codes and suspicion from perceived symptoms (not depicted). The attribute “source” of these suspicion link classes is then overridden depending on the origin of the link.

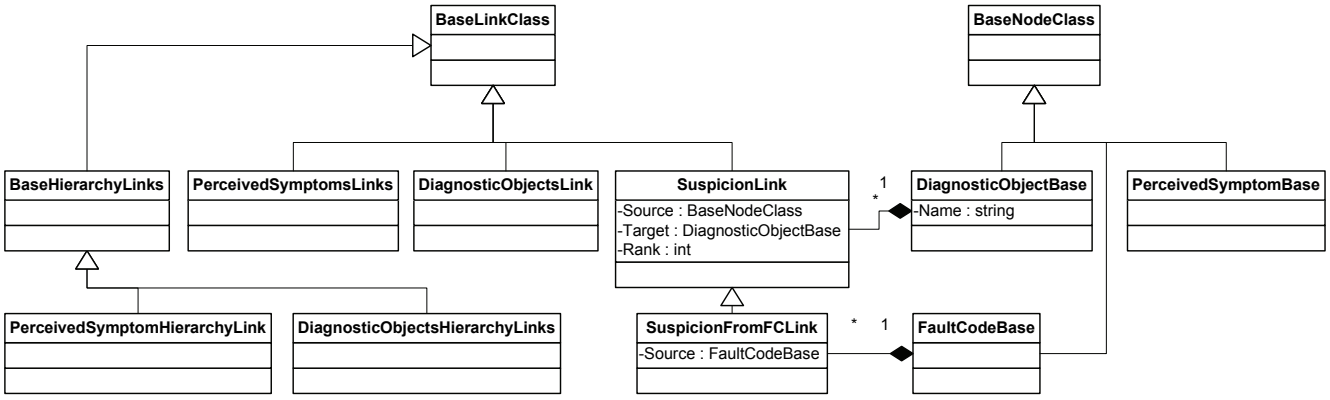

Figure I-6: Example of an information model for diagnostic knowledge structures

The classes depicted in Figure I-6 will be detailed in chapter I section 2.5.3, which is concerned with the description of the diagnosis related knowledge structures.

2.3 Technology and interfaces:

SIDIS Enterprise is based on standardized technologies which are sure to be used in the future. This means that the basic system architecture is aligned to current technical standards and integrated into existing IT infrastructures. The Microsoft .NET platform allows a hardware independent running. Open interfaces through XML play also a central role here, because this format is free of charge.

- All system components were developed on a uniform .NET basis.
- The service orientated architecture (SOA) allows a high degree of modularization, flexible distribution and cost-optimized maintenance of the software components involved.
- Communication with centralized server components is via http / Web services.
- Information can be imported and exported via XML-based interfaces and, if necessary, formatted using XSLT style sheets and represented graphically (e.g. diagnostics reports).

As an open system, SIDIS Enterprise uses standardized external interfaces. These are based for preference on XML and/or Web services. Within the Workshop system, Web services can be addressed normally via service adapters. This allows all required method calls to external systems to be logged transparently. The adapters are to be constructed to suit individual manufacturers and to encapsulate the features via a standard interface. This way they decouple the specific external system from the standard SIDIS Enterprise components used. The connection mechanism supports parameterized data transfer in both directions; both synchronous and asynchronous method calls are possible.
The authoring system supports data export and import using an XML file interface, for integrating a translation management system (TMS), for example. Other external systems which can be integrated to suit specific manufacturers include:

- DMS (Dealer Management System)
- TMS (Translation Management System)

2.4 Deployment and usual operating scenarios

Modules can be distributed to separate hardware under appropriate conditions and introduced into the structure there. This makes it possible to distribute the modules independently of the system landscape in the workshop, for example, to devices and servers or combined on a device for standalone operation. System distribution can be orientated to the system and hardware platform and then occurs as a factor of the infrastructure and performance of the hardware. For example, devices which, because of their hardware resources, are only usable as a client can also be deployed in combination with a workshop server.
The SIDIS Enterprise workshop system distribution scenarios are diverse and can be preconfigured in the authoring system. Some examples of this application are shown in the following scenarios:

- Standalone operating mode: An end device equipped with appropriate hardware can be operated in standalone mode with full functionality. Update and online communications can be used directly from the device via network or Internet connections. This scenario also provides the range of functions needed for the mobile service.
- Workshop server operating mode: A workshop server is a central, fixed server at workshop level. It is connected to the relevant end devices via a LAN or local wireless network. A workshop server increases the range of application types. For example, the structural logic and data-content modules are installed on the workshop server. The automatic update service and online communications are also available there. The exchange of task information and connections to third-party systems (e.g. DMS) are also possible. An end device which can also be operated as a standalone device allows a change of operating mode, with or without server, as for example in the parking lot or during a test drive. The data are synchronized via the existing network connection to the workshop. Figure I-7 provides an overview of the usual scenarios described above.

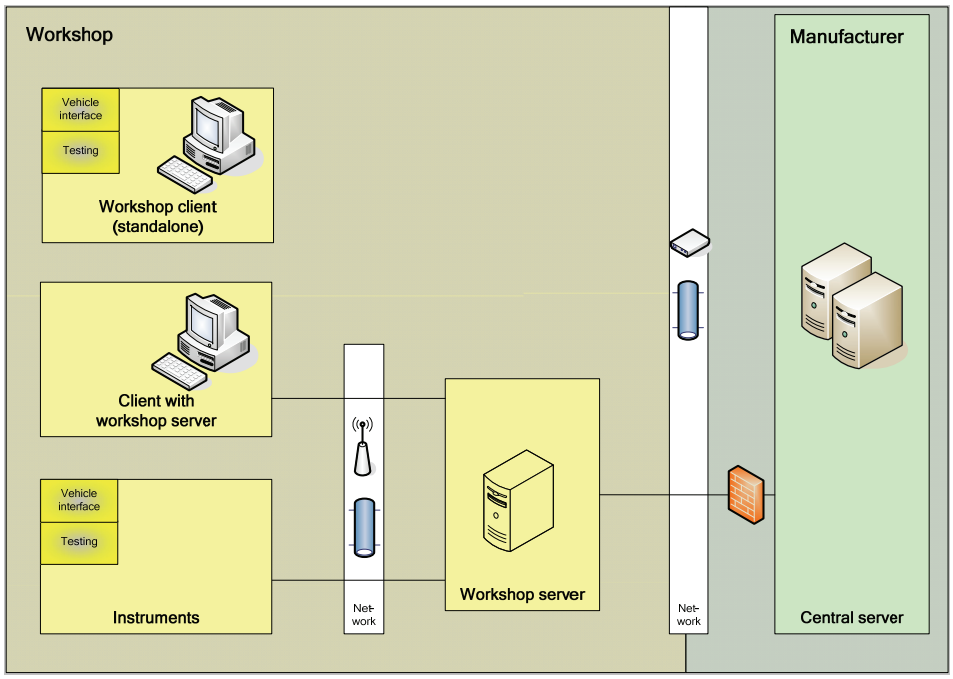

Figure I-7: Scenario overview

2.5 Data edition and structure editors

2.5.1 Overview of the data generation process

The SIDIS Enterprise authoring system facilitates the efficient generation, maintenance and re-versioning of all vehicle data and service documentation for the workshop. Figure I-8 provides an overview of the basic procedure for data generation and use:

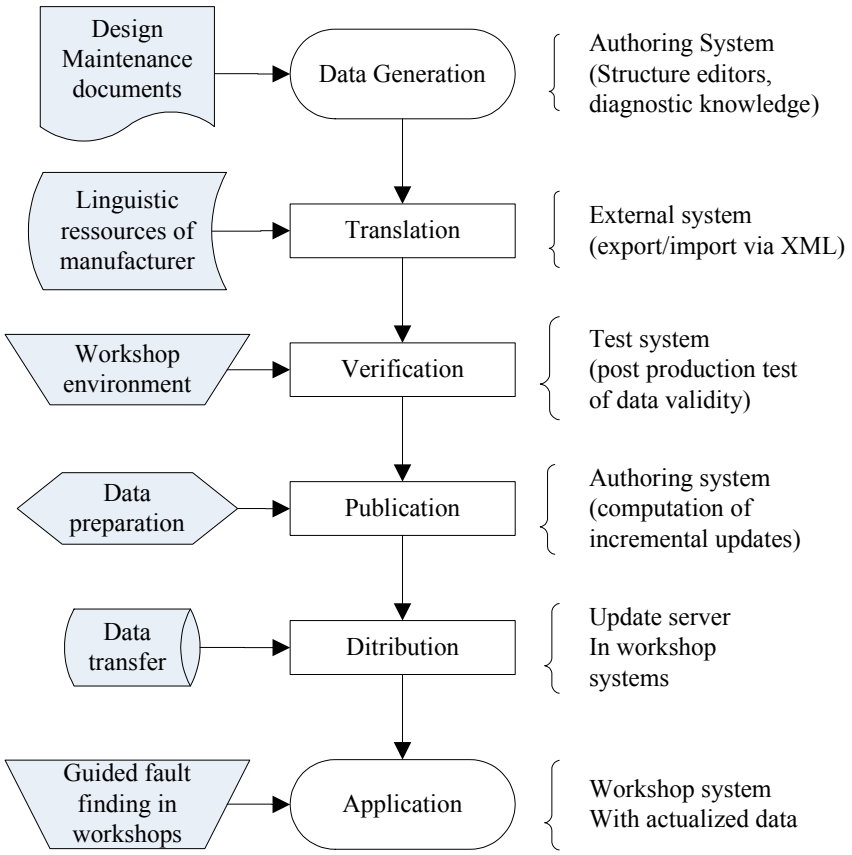

Figure I-8: Main steps in data generation

The diagnostic data generation is the most important activity with the application in the workshops, which includes the following activities:

- Describing the complete vehicle series. This activity involves the following steps (cf. chapter I section 2.5.2 to 2.5.4): specifying all equipment used in all vehicles, structuring and setting parameters for the diagnostic objects. Here, the authors need the design documents of vehicle to create and edit the model for the diagnosis.
- The vehicle variants (characteristics, vehicle configuration, variant rules, etc.) are identified and limited (cf. chapter I section 2.5.2 and 2.5.3): specifying the basic features of the vehicle and assigning appropriate diagnostic objects, generating vehicle configurations, generating the required variant rules.
- On the basis of the vehicle description: Modeling the faults and their causes: logical connection between individual diagnostic symptoms / perceived symptoms and their possible causes (cf. chapter I section 2.5.3 and 2.5.4): establishing the fault code for the ECU, plus its meaning in natural language, establishing the perceived symptoms, establishing the expert rules for the diagnostic objects, creating suspected links between fault codes, symptoms and diagnostic objects.
- Expanding system functions using tests (cf. chapter I section 2.5.3): generating tests for ECUs, functional and repair tests for diagnostic objects, for calibrating sensors and actors, as well as supporting vehicle identification, attaching test to diagnostic objects.

A screenshot of the different editors is depicted in Figure I-9.

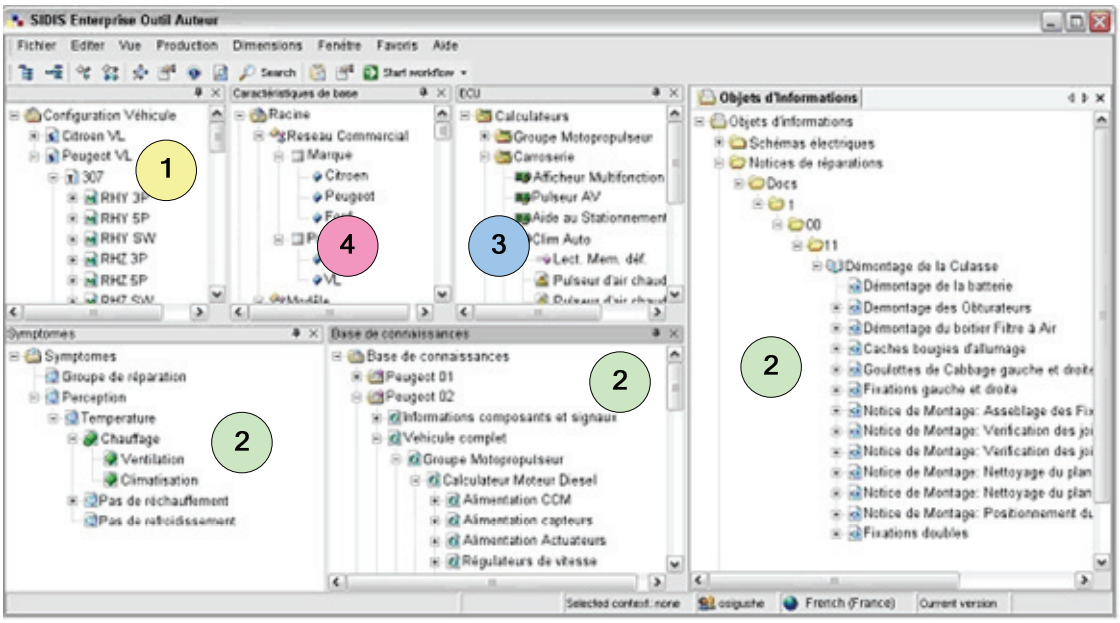

Figure I-9: Screenshot of the authoring system with different editors

This process emphasis the following list of structure editors:

- Editors handling the vehicle's variant (indexed by 1 in Figure I-9):
  - Characteristics master data:
    Listing the different vehicle's characteristics like model, engine, etc.
  - Equipment master data:
    Containing all the optional equipment of cars that have no ECU communication.
  - Vehicle configuration
    Used to resolve the variant in the workshop system.
  - Variant rules:
    Permits to establish the validity of diagnostic object, tests, documents, etc.
- Editors related to diagnosis knowledge (index by 2 in Figure I-9):
  - Diagnostic objects:
    Component model of the car, with tests and service documentation.
  - ECUs:
    The editor with all the electronic control units and fault codes.
  - Perceived symptoms:
    The qualitative description of failures.
  - Empirical rules:
    This intermediate editor allows the formulation of complex expert rules.
  - Test editor:
    This editor allows to create tests using standard function from a toolbox.
- Editors related to electronic devices (indexed by 3 in Figure I-9):
  - Fault memory fault text master data:
    This tree associates a text in natural language for the fault code in order to make them interpretable.
  - ECU job tree:
    The ECU Jobs tree is meant to describe the jobs (services) of specific ECUs.
  - ECU function tree
    The ECU Functions tree describes the ECU jobs on an abstracted (functional) level.
- Publication editor (indexed by 4 in Figure I-9):

This editor permits to create and update package of the data. As seen in Figure I-8, the data have to be validated before being sent as update to all the workshops. The usual scenario is to use a workshop client as post-production test for the validation of the new edited data from authors. The publication job consists in the collection of all required information (trees, sub-trees, documents, etc…) and to

create a package of files with this information. This package is then sent to an update server where the workshop system automatically checks for new updates.

The interesting thing in the publication process is that during the collection of new information, some pre-calculation can be made and distributed to the workshops. This option was used for the retrieval of qualitative failure description in chapter II section 2.

An overview of the main editors and the links between them is given in Figure I-10 :

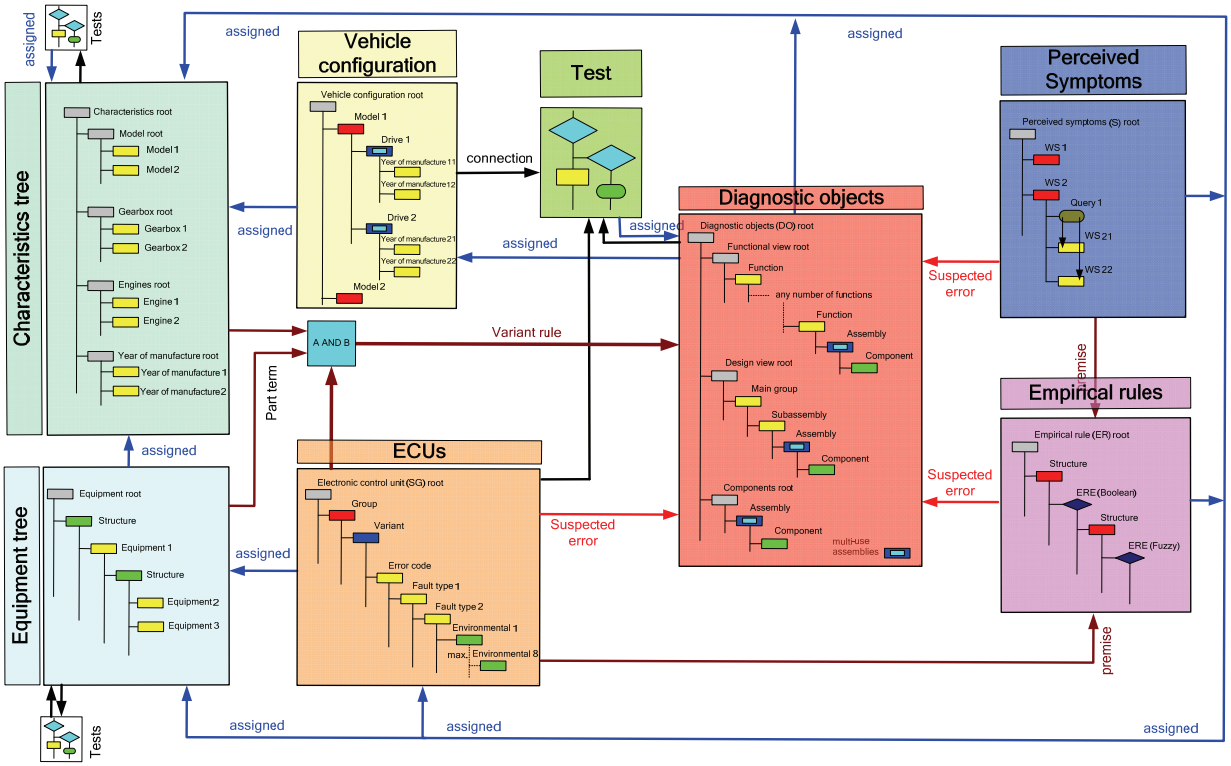

Figure I-10: Overview of main editors and links between data [Ripplinger, 2006]

2.5.2 Editors resolving the vehicle variants

The following subsections present briefly the different editors and their link to handle the vehicle variants. These editors are fundamental for the vehicle identification step (cf. Appendix A section 1.1) and afterwards for the diagnosis. These editors define the validity of diagnosis data, electronic configuration, tests, documentation and equipment for a given vehicle.

- **Characteristic tree**: The characteristics master data editor contains the basic characteristics tree. The roots for characteristic types followed by the actual characteristic nodes can appear under the characteristic root. The information is usually subdivided into the following categories: model, gearbox, engine, year of manufacture. These nodes are linked with Diagnostic objects, documents, tests, vehicle configuration, variant rule, equipment.
- **Equipment tree**: The equipment master data editor is used to create objects for optional vehicle components derived from the characteristics of the vehicle configuration. The associated tree contains random hierarchies of structural nodes under a root for guidance and actual non-communication dependent equipment such as sliding roofs, tow bars, etc. This tree can be linked to characteristics, diagnostic objects, documents, tests and variant rules.
- **Vehicle configuration tree**: Vehicle configurations are used in the workshop to identify the vehicle. The possible combinations of characteristics given in the vehicle configuration are either presented to the mechanic for selection (manual identification) or decided

automatically by linked tests using the vehicle communications system (automatic identification). The identified vehicle characteristics are used to control procedures and to select data attributed to characteristics (diagnostic objects, test, documents, etc.). In this way, the data displayed in the workshop system is filtered accurately to match the vehicle being diagnosed. The linked information with the vehicle configuration tree are characteristics, tests and documents (the linking principle is the same as for the trees in the previous sections).

- **Variant rules tree**: Variant rules can be assigned to diagnostic objects, tests or documents and thereby establish their validity as depicted in Figure I-10. If they are linked to diagnostic objects, they automatically influence the associated tests and documents. However, tests and documents can also be linked to a variant rule independently of the diagnostic object. This is always necessary if multiple tests or similar document types are linked to a diagnostic object. Variant rules contain a logical expression. This in turn consists of equipment, characteristics or ECU data linked by the operators AND, OR and NOT, and single brackets. Characteristics can also have usual prefixes like: <, >, <>, >=, <= (eg. For rules like "All vehicles construction year > 2002").

2.5.3 Editors related to diagnosis knowledge

There are 5 editors which are the core of the diagnosis engine. The first one: the diagnostic object editor is a model of the vehicle which is explored during the diagnosis (cf. chapter I section 2.6). The 2nd and 3rd ones are symptoms whereby direct observations called perceived symptoms and self-diagnosis symptoms obtained through the acquisition of fault codes from the ECUs. These ones are linked to diagnosis objects which mean that the targeted component is declared as suspected if the symptom is active. The 4th editor is the empirical rule tree, which permits to formulate more complicated suspicion rules with the help of Boolean or fuzzy logic. For example: a targeted diagnostic object of an empirical rule is declared as suspected if the Boolean expression is evaluated to true. The last editor is the test editor. This one permits to create tests with the help of a toolbox, which contains blocks of functions like vehicle communication, function dialogues, etc… The basic knowledge structures: observations and components are depicted in Figure I-11. The subsequent list details each editor:

- **Diagnostic object**: This tree is the core of the diagnosis system. Under the root (Figure I-11) are three further roots for the tree sections functional, design, and components views. The tree section for the functional view contains functional groups and functions, as well as assemblies and components (these can also be defined in the component or design view tree section and can be used multiple times from the functional view tree section by means of links). The tree section for the design view contains nodes for main and sub-assemblies, as well as assemblies and components (these can also be defined in the component or functional view tree section and used multiple times from the design view tree section by means of links). The tree section for components contains nodes for assemblies and components. The functional and design modeling of vehicles and components in a hierarchy provides different views (i.e. different access paths in the selection of functions and components) of the diagnostic objects in the workshop system.

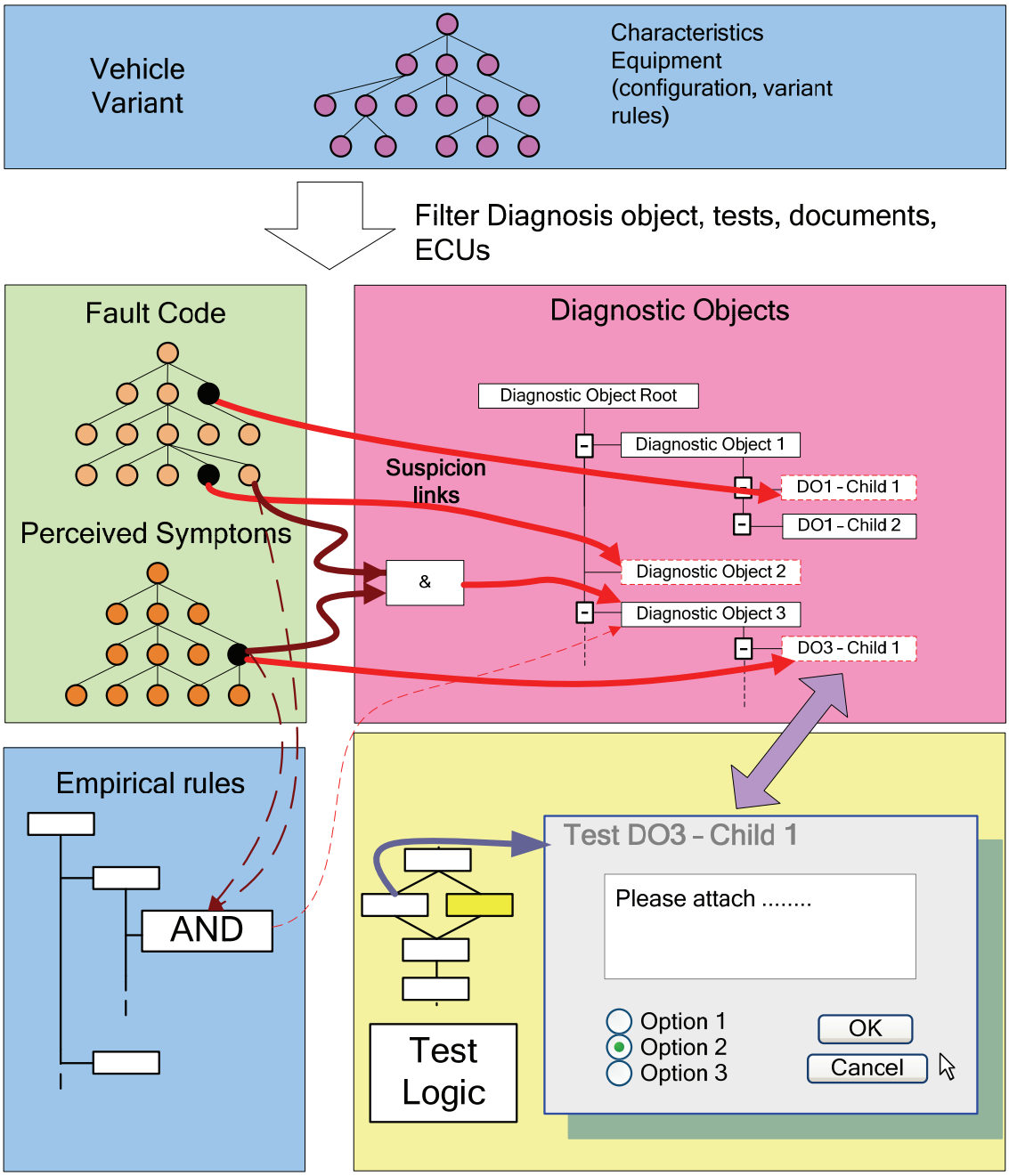

Figure I-11: SIDIS Enterprise diagnostic principle

- **Perceived symptoms**: they represent qualitative descriptions and reflect available observations, e.g. "engine does not start in winter ". Because in some cases it appears not to be relevant to reason in terms of the actual values or quantities. Rather, it can be sufficient to reason in terms of (qualitative) deviations from nominal values only by distinguishing different orders of magnitude. These symptoms can also contain a modal condition of the apparition of a failure like “engine bucks after starts during cold stage only”.
- **ECU tree**: This tree contains the ECU and the tree for fault codes contents with suspected relationships. Under one root, the tree contains two hierarchy levels for ECU groups and variants, with fault memory nodes under that and manually entered suspect relationships. The fault memory nodes can also form a hierarchy. It consists of fault codes and various types of faults as depicted in Figure I-11. The ECU objects perform the vehicle system test in the workshop system. They are identified one by one and their fault memories are read.

- **Empirical rule tree**: The empirical rules tree is an editor for complex suspicion links with premises and suspected relationships targeting diagnostic objects. Under one root there is a random hierarchy of structural nodes and leaf nodes for either Boolean or fuzzy empirical rules (see Figure I-11 for a simple AND rule from a fault code and a symptom).
  - **Boolean**: The Boolean empirical rules contain a logical expression of links (premises) to perceived symptoms and/or fault memory objects (ECUs). The diagnostic object is suspected if the logical expression composed by the premises is evaluated to true
  - **Fuzzy**: The fuzzy empirical rules contain links to perceived symptoms and/or fault memories with a particular weighting.
- **Test editor**: Test are always used where fine-grain intelligent structures have to be created. This is mainly the case with the procedural portion of the diagnostics, but also with programming of ECUs, for example. Tests allow the system to adapt flexibly to boundary conditions and requirements which arise, for example, from continuing developments in vehicle technology. A test consists of a structure in the form of a step-by-step chain. A graphics editor in the authoring system can be used to create a separate interface for each process step in the tests. Individual steps can contain logic in the form of library functions or scripts. Functions can for example be: vehicle communication with ECUs through ASAM, ODX, external interfaces (with some standard GUI controls like buttons, textbox…), or access particular data in the session context. One great advantage in the editing of tests in SIDIS Enterprise is that the vehicle communication is integrated through a DSC (Diagnostic System Component) adaptor. In other words, the DSC adapter is responsible for converting the convenient abstract calls from the DSC into concrete context-dependent API calls. To do this, the DSC adapter uses the master data stored in the CMS and the context data to perform the corresponding DC call. The DSC adapter needs to be configured specifically for each manufacturer depending of the ECUs to be addressed.

2.5.4 Editors related to vehicle's electronic devices:

The last parts of editors that are presented are related to the vehicle's electronic devices. There are 4 involved editors; the ECU tree had already been described in chapter I section 2.5.3

- **Memory fault text tree**: The concepts of fault codes and fault code texts are clearly separated, as the texts belonging to a fault code are defined separately in the ECU Memory fault text tree. The advantage of such an information structure is that the texts shall be defined only once in the Memory fault text tree and then referred to by the ECU variants located in the ECU tree. Due to the big number of texts to administrate, structure nodes can be inserted in the tree to order the texts. However, multiple definitions of the fault code texts can be stated all the same. As the text objects contained in the Memory fault text tree are supposed to describe the objects contained in the ECU tree, each node type in the ECU tree has its counterpart in the Memory fault text tree. However, the Memory fault text tree is definitely designed to store a unique version of the texts as it features sub-root nodes for each type of information object (fault codes, fault modes and environment data). This feature does not prevent from defining environment condition texts that are specific to one trouble code.
- **ECU job tree**: The ECU jobs tree is meant to describe the jobs (services) of specific ECUs in a diagnostic oriented way, which means that only the jobs interesting for a diagnostic session are imported in this tree. Following the same idea, only the interesting response parameters are to be found in this tree. There is no prescribed structure to order the jobs in this tree, which means the author has to choose the best structure on his own so as avoid redundant job definitions.
- **ECU function tree**: The ECU functions tree describes the ECU jobs on an abstracted (functional) level. It reuses the jobs located in the ECU jobs tree and puts them in relationship with specific ECU-variants. Note that the ECU-variants in the ECU Functions tree can be related to one physical ECU in the ECU tree or stand for a virtual ECU. As the information from the ECU functions tree is to be presented on the garage client display, functional nodes are introduced between the ECU-variant and the job nodes, allowing to group similar jobs

together on the one hand and giving the job an explicit name and fixed function type on the other hand.

2.6 Guided fault finding

The functional modules in the SIDIS Enterprise workshop system are designed to have minimum or no dependency on each other. The functional relationship and thus the configuration of the workshop system result from the definition of sequentially executed functional modules known as a SIDIS procedure. These can be edited from in the authoring systems. One standard procedure which will be partially detailed in this section is the guided fault finding procedure also called guided diagnosis. Figure I-12 shows the different steps of the guided fault finding procedure:

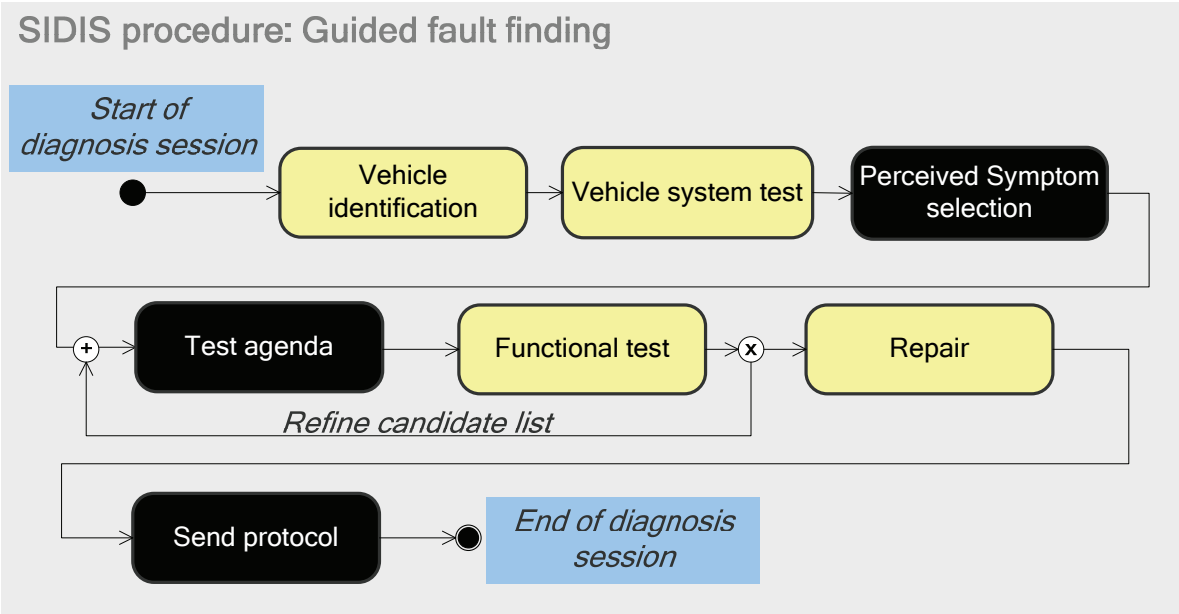

Figure I-12: Guided fault finding procedure

During guided fault-finding, the technician in the workshop is guided from a particular fault scenario to the vehicle component(s) that are responsible for the fault scenario. If necessary, a choice is made between several suspect vehicle components after further testing. The module with a black background color in Figure I-12 will be examined deeper in this dissertation.
The guided diagnosis procedure follows the subsequent steps:

- **Vehicle identification**: determines the characteristics of the vehicle to be serviced. Using these vehicle characteristics, SIDIS Enterprise can provide the mechanic with the available information in the context of the identified vehicle: It is possible to select diagnostic and repair structures to suit the current particular vehicle (eg. to select tests, documents that suit this vehicle). The vehicle characteristics are normally determined automatically by the reading of the VIN (Vehicle Identification Number), but can also be entered manually by the mechanic.
- **Vehicle system test**: All installed ECUs of the vehicle are interrogated systematically during vehicle system testing. Interrogation relies on earlier vehicle identification because the theoretical installed status of the ECUs can only be known after the vehicle has been identified. Sequential interrogation of individual ECUs determines: which ECUs are actually present in the vehicle and able to be accessed using the vehicle's communication system; which ECUs have fault memory entries; and which ECU versions are involved.
- **Perceived symptom selection**: This functional module shows the user the possible symptoms from the perceived symptom tree relevant to the vehicle in question. It is only possible to select those symptoms which have at least one valid suspicion in relation to the vehicle. The

user can select any symptoms from the symptom tree, and these are then added to the list of perceived symptoms.

- **Test agenda**: After the perceived symptoms and self-diagnosis symptoms (see chapter I section 2.6.2) have been collected, it is possible to draw up the test schedule. This is done in the SIDIS Enterprise "Test schedule" functional module. Due to the important number of suspected candidates functional tests have to be executed to incriminate or discriminate candidates. The technician has a list of the 7 most suspected objects were he can choose which part of the vehicle he can investigate deeper until the defect component is found and repaired.
- **Send protocol**: The feedback system integrated into SIDIS Enterprise allows the transfer of user and system information to the vehicle manufacturer. This information can be used to optimize the system. Most of the information is compiled automatically in the course of a diagnosis session (e.g. completed tests), other information comes from supportive input by the user (e.g. successful repairs, installed parts), and yet more comes from the user's own experience (suggestions for improvement, comments). The protocols can be entirely configured into the authoring system.

2.6.1 Perceived Symptom selection

This functional module constitutes a very important step in the initialization of the diagnosis sessions, because the more symptoms are selected, the better the information to perform the diagnosis and the more time is saved. The problem at the moment is that the selection of the perceived symptom is done by the technician in the workshop and the user interface presents the symptom tree (filtered, depending of the vehicle). It is particularly fastidious to navigate through this tree without experience and no idea how the information is structured. For instance for certain car manufacturers they can be around 1700 symptoms and the tree can have 4-5 levels, and as specified in chapter I section 1.1.3, this amount of information is going to increase. If a comparison is made of diagnosis sessions with the same broken elements with and without the selection of relevant symptoms the performance difference can reach 5 tests in the **best cases** which means a **time saving of 10 to 30 minutes** (whereby the time of tests is difficult to estimate due to the strong context dependant value). A user-friendly interface in this step of the guided diagnosis session which maps the relevant symptoms from a request in natural language as depicted in Figure I-13 could help to increase the diagnosis performance.

The nature of human is to have intellectual capacity deployed over millennia via the so called natural languages. Today, the language is present in the core of all information systems: oral when it comes to the phone, written for Emails, files for the web and for knowledge based systems. However, the competition between businesses and organizations leads to the question of the usage of "natural language" in information systems as a source of potential productivity. The diversity of the forms but also in the meaning rises to a great challenge: to ensure that the language faculty was extended into machines and software. This is a technological challenge which aims to provide concrete answers to the growing management of texts, documents, e-mails, sound and video productions. How to provide this convergence concerning the language that everyone expected to need in a wide spectrum of tasks like: find, organize, analyze, diffuse, reproduce, check and summarize? This dissertation will examine certain aspects of the difficulties of research in natural language in written form for a practical use and an operational application within the module of the search of car symptoms by queries in natural languages. This study will address a number of problems which deals with the nature of the languages.

Firstly this report will present the difficulties (linked to natural language) and to the extension of the interpretation of the language by machines. Then we will address the performance indicators of a document retrieval system with the notions of precision and noise of a search engine (cf. chapter II section 2). The third part (cf. chapter II section 2.1) will be devoted to the State-of-the-art in retrieval systems which is followed by the study of some developed solutions (cf. chapter II section 2.3) and the examination of their behaviour with a test base composed by multilingual queries. In Appendix A section 2.1, the details are given about the implemented language resources in the developed solutions.

Chapter II section 2.3.3 will conclude this study by proposing a few possible orientations for the improvements of the research module [Azarian et al., 2007] as for example the enhancement by a feedback (cf. chapter III section 3.1.3).

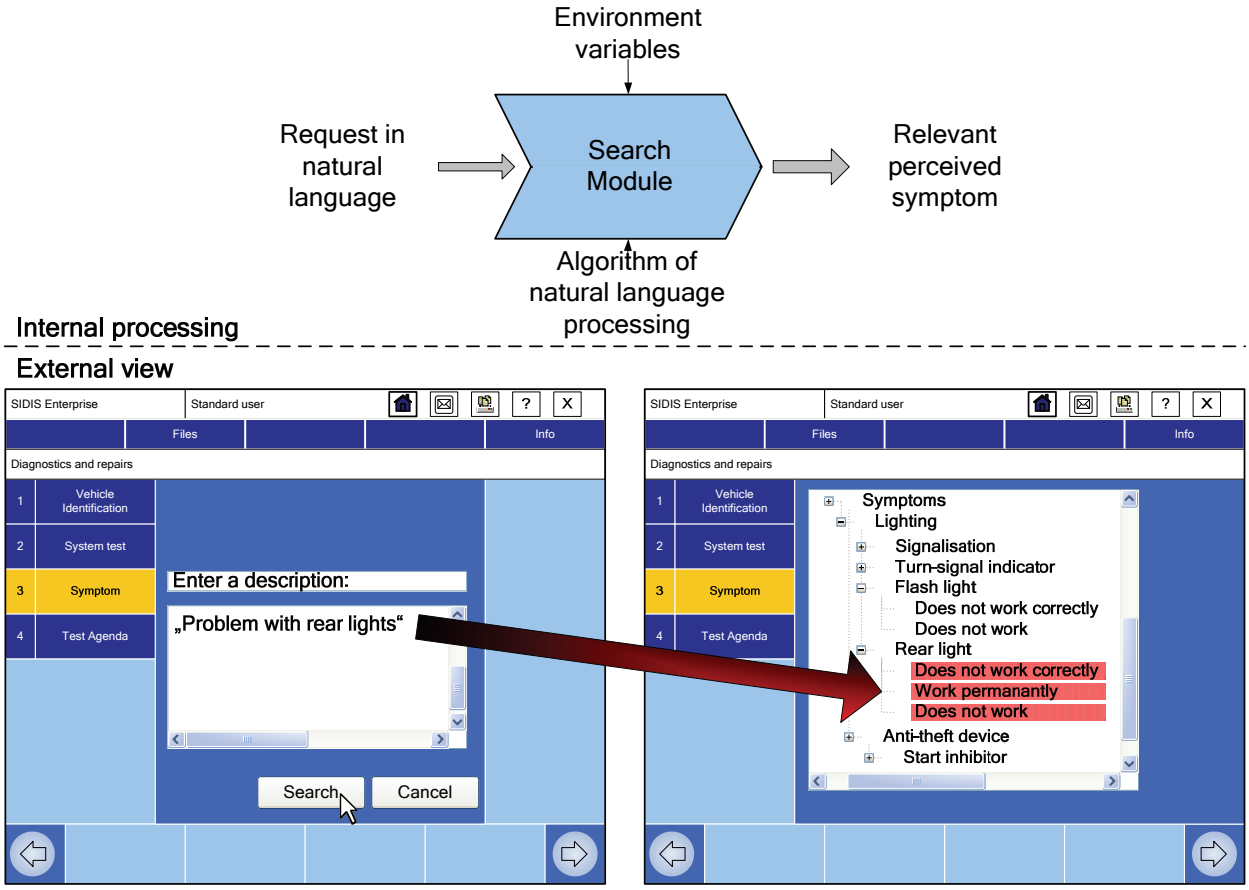

Figure I-13: Mapping from request to symptom tree with GUI implementation example

2.6.2 Diagnosis algorithm

The diagnosis algorithm of SIDIS Enterprise begins with the computing of a prime test agenda. This prime test agenda is a list of the 7 most suspect components based on the evaluation of the suspicion links and empirical rules. Afterwards the user selects a component or a group of components to investigate in this prime test agenda.

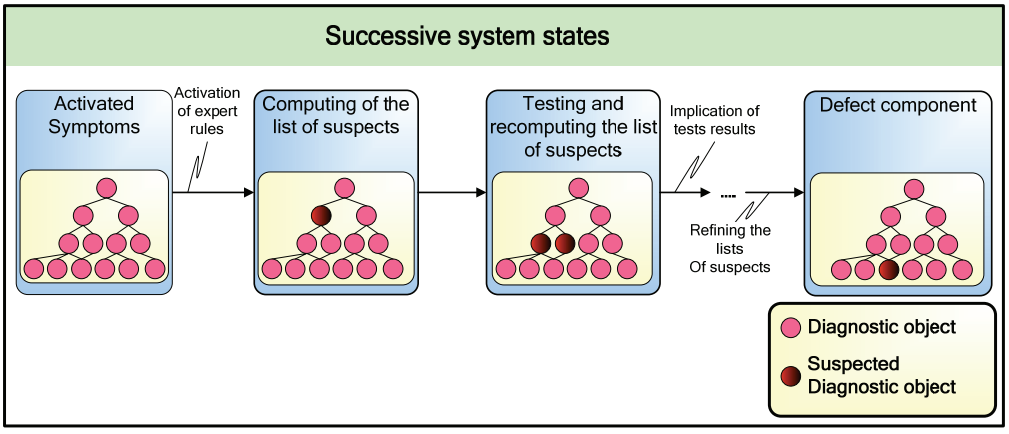

Figure I-14: System states during a diagnostic session

The diagnostic algorithm of SIDIS Enterprise takes into account the attributes listed in Table I-1 associated to nodes and links:

Attribute	Notation	Default Value
Symptom weight	*GS*	100%
Diagnostic object weight	*GR*	100%
Rule rank (If rules are hierarchies)	*RSL*	1
Test cost of a node	*Tc*	none
Weight of hierarchical link	*GDO*	100%
Weight of the test confidence	*GT*	100%

Table I-1: Attributes list and default values

The partial moments are computed for each appropriate suspicion relation "suspect object" depending on the number of suspicion links from the symptom called *k* for each symptom Si, given in (Eq. I-1):

$$PM_n = GS\frac{(k+1) - RSL_n}{\sum_{i=1}^{k} RSL_i}$$

(Eq. I-1)

With:

- *GS*: the weight of the symptom
- *PM*: the partial moment or partial suspicion degree induced by a rule
- *k*: the number of rules from the symptoms
- *n*: index of the considered suspicion rule
- *RSL*: rank of the considered suspicion rules

The evaluation of empirical rules is slightly different and depends of the number of the premises and the type of rule. A complete description is given in appendix A section 1.4. They are not detailed in this section because they are rarely used.
Once the suspicion moments denoted SM, given by the sum of all partial moments of the diagnostic objects have been evaluated, the objects are ranked by a heuristic metric *h* given in (Eq. I-2):

$$h = \frac{SM_x}{Tc_x}$$

(Eq. I-2)

With:

- *SM*: the total suspicious moment (or the sum of all partial moments)
- *Tc*: the test cost for the precised element given in index

The technician can select to perform a test on one of the 7 most suspected objects. 3 usual scenarios can occur depending on the test-result:

- Test result "NotOK": implicit "NotOK" declaration of all child nodes of the diagnostic object.
- Test result "OK": implicit discrimination of the child nodes of the diagnostic object
- Test result "?OK": the test returns the result "OK" but the test confidence was not equal to 100%. The child objects are set to "OK".

The result of the partial moment depending of the test results is given in the following list:

- "NotOK" declaration: Computation of the partial moments for each hierarchical relationship in the diagnostic tree (type "Is composed by[*j*]" relationship), depending on the total number of "is composed by" relations called *NB*(DOx) is given in (Eq. I-3):

$$PM(DOx, DOx\_Child[i]) = GDO.\frac{(NB(DOx)+1-i)}{\sum_{k=1}^{k=NB(DOx)} k}$$

(Eq. I-3)

- "OK" declaration: All child objects of DOx, which corresponding test giving the results OK are declared as OK (excepted if the confidence of the Test is <100%, see point 3). The OK state of a node can only be changed if the same test is repeated or another test explicitly declares that the node state has to been changed to "NotOK".
- "?OK" declaration: If the test result is "OK", but the test confidence is less than 100%, the explicit suspicion due to the list of suspected objects DO[*i*]. Evaluation of the partial moments for each diagnostic object in the list of suspects is given by (Eq. I-4):

$$PM(DO[i]) = GT$$

(Eq. I-4)

The diagnosis algorithm of SIDIS Enterprise explores recursively the diagnosis object tree after the ignition of the expert rules which depends on the perceived symptoms and fault codes. The weights of symptoms, the rank of the rules and the weight of the hierarchy links play a central role in the ordering of the test agenda if they are optimized. But in practice authors do not commit themselves to edit the weight parameter and therefore the test schedule is not optimal. This phenomena is illustrated by the abstract example (issue from an air conditioning subsystem of a car) depicted in Figure I-15.

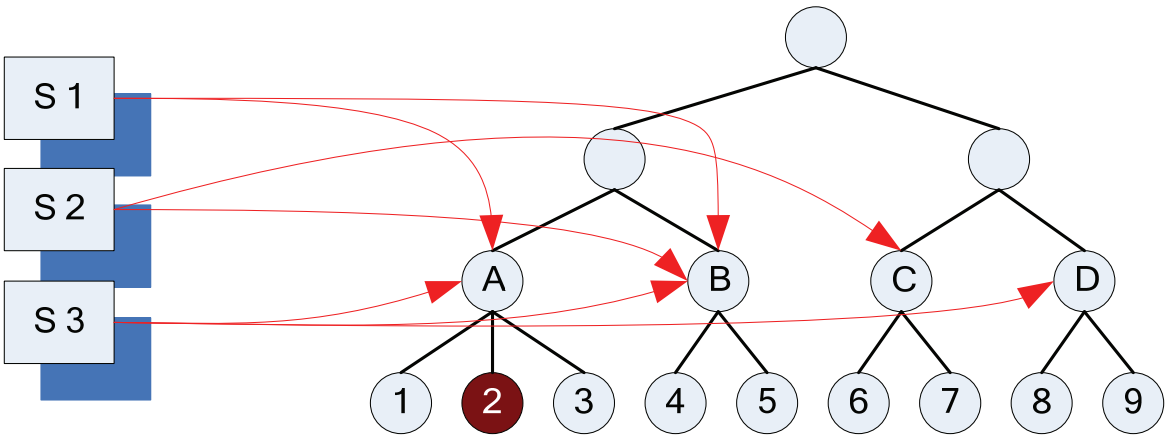

Figure I-15: Example of a restricted model

- Example:

In this example we suppose that the entire weight coefficients (and the test confidence) are set to their default value (100%) and the component corresponding to node 2 is broken. We suppose that symptoms S1 and S2 are active at the beginning of the diagnosis session and that the test cost (of the relevant node) are known and given in Table I-2. During this example we suppose that the technician always tests the first ranked object in the test agenda.

Associated Node	Test Cost (minutes)
Node A	5
Node B	3
Node C	2
Node 1	2
Node 2	5

Node 3	7

Table I-2: Test costs

Due to the expert rules from S1 and S2, nodes given in Table I-3 are suspected.

Node	A	B	C
SM	66	133	66

Table I-3: Total suspicion

Clearly, node B has the highest suspicion, because of 2 active suspicion links pointing to the object. The associated test of node B will be ranked at the first position in the test agenda and then the test return the result OK. Node B and its child will be discriminated. Then node C will be ranked first in the second test agenda, because it had a lower test cost than node A. In the third test agenda node A will be ranked first and logically return the result NotOk. Nodes 1, 2 and 3 will have an implicit suspicion degree of 50, 33 and 16. Node 1 will appear in the 4th test agenda and then finally the broken object (node 2) will be ranked first in the 5th test agenda. As seen in this example due to the fact that there is no hierarchy between the rules, 3 tests have been performed in vain. If the symptom S2 had a weight of 50% and if the rule from symptom S1 to node A had the maximal priority then the first test agenda would be: Node A (*SM*=66), node B (*SM*=50) and node C (*SM*=16), and just with this both parameters 2 unnecessary test can be avoided and significantly reduce the time of the diagnosis session as well as user effort.

2.6.3 Protocol engine

The feedback system integrated into SIDIS Enterprise allows the transfer of user and system information to the vehicle manufacturer. The main advantage of SIDIS Enterprise is that the protocols are fully configurable and that this information can be used to optimize the system. Moreover thanks to the CMS system, all protocols are language independent and just the unique object identifier is written in the protocol. As seen in the example before, the expensive aspect of vehicle repairs is in the fault localization step. More tests which means more time is needed to solve the problem. Actually there is no software part of SIDIS Enterprise dedicated to the evaluation of the feedback protocols of diagnosis sessions despite the fact that feedback of a manufacturer network represents a valuable source of knowledge. With 2500 workshops in Europe for a manufacturer like BMW there could be at least 10.000 protocols a day in Europe. This experience based knowledge could be shared with other workshops. The feedback system aims to:

- Reducing time of diagnosis sessions
- Avoiding unnecessary component replacement
- Compiling empirical knowledge from the mechanic
- Extending the vehicle model by other expert rules
- Optimize the automatic symptom selection
- A course for continuous improvement:
- Recording diagnostic errors and software problems
- Reporting suggestions and complaints

In order to collect all the necessary data, the workshop system is equipped with a protocol engine which logs all the data of a diagnosis session using a reflection mechanism. A typical protocol of a diagnosis session contains:

- Basic vehicle configuration (e.g. type, series, year of manufacture)
- Vehicle details equipment and km
- System tests (faulty ECUs)
- Selected perceived symptoms
- Guided diagnostic steps (prime test agenda, selected test by user in agenda, second test agenda…)
- Faulty components, open documents

- Optional: e.g. improvement suggestion

One of the most expensive parts of the protocols is the evaluation of the feedback data. In chapter III, we explore this area. By focusing on the crucial parameters which can optimize the diagnosis inductive learning techniques for the extraction of data and how values of parameters can be induced and automatically update the database for all workshops.

2.7 Outlook on SIDIS Enterprise

2.7.1 Synthesis

As presented in the previous sections (cf. chapter I section 2.1to 2.6) SIDIS Enterprise is a complete car diagnosis solution for manufacturers after sales networks where manufacturer and workshops operate through a common information model. One strong advantage of SIDIS Enterprise is that the versioning of diagnostic objects permits to re-use sub-trees for other vehicles, which makes it possible to re-use components or groups of particular devices. The usual operating scenario for the data editing process permits to fix several user groups dedicated to particular tasks but the overall complexity of the model makes it difficult to have an overview. For example for the diagnosis data it is difficult to anticipate how a test agenda will be if a new suspicion link is edited or if weights are changed. In a special implementation for a car manufacturer this leads to the development of high priority suspicion links, which if active rank the target diagnostic object at he first position. This particular exception is due to the reason that the overall behavior of the algorithm is difficult to anticipate. Moreover, authors do not commit themselves to edit a particular weight coefficient which had a strong incidence on the test agenda as seen in the abstract example in chapter I section 2.6.2. Other parameters like the cost of the tests are usually unknown and the tests are published without their relevant parameters. This lack of information leads often to time consuming diagnosis sessions.
In the workshop system there is also the problem of the qualitative failure description which makes it difficult for technician to find and navigate through the symptom tree (cf. chapter I section 2.6.1) which in the most use-cases leads to the ignoring of this step in the guided fault finding procedure, whereby perceived symptoms are a useful source of information and can help to better discriminate the components. Another problem is the interpretation of the results in the diagnosis. The technician just has a successive list of test agendas and can select which group to investigate deeper, but how the diagnosis computes this result e.g. from past experience, from causal dependencies, is not explained. The way the test agenda is generated should be explained to the technician to make the algorithm transparent and the results interpretable.
Another drawback of SIDIS Enterprise is that there is no feedback system that permits to evaluate past results and to learn or correct the initial data in the authoring system. As conclusion, SIDIS Enterprise fits well in the actual automotive after-sale network but due to time constraints the manufacturer or authors do not use the full potential of the software. In the scope of this dissertation, the main drawbacks of SIDIS Enterprise are discussed in chapter I section 2.7.2 and the current state of the art of diagnosis techniques as well as automatic learning methods are reported in chapter I section 3, chapter III section 1 and 3.

2.7.2 Motivation

One major point of this dissertation is to improve the diagnosis algorithm so that it fits with future requirements. As discussed previously, the new electronic area in the automotive industry with several multiplexed networks and X-by-wire technology (already available for some vehicles), which eliminates mechanical or hydraulic connections between elements and replaces them with wires (for example for brakes, steering) highly increases the dependencies and communication between components. A brief overview of the possibilities of X-by-Wire is given in [Trevett, 2002] chapter I, section 2.6.1. Finally, according to a press kit published in 2006 by the French Atomic Energy Commission (CEA) [CEA-2006], the need of efficient diagnostic algorithms will be more and more demanding because of all the security systems on board of vehicles. Several difficulties occur with

these coming technologies: perceived symptoms will become more and more important as well as the part "natural language failure description" (cf. chapter I section 2.6.1). In order to prolong the "natural language" faculty in the workshop system, this dissertation reports about a module, which allows to take this aspect into account to save time during the diagnosis. Finally, this module will lead to improve the diagnosis algorithm and the input of a diagnosis session. To do this, the following questions have to be answered:

- How can natural language be interpreted by a machine?
- How can an algorithm judge of the relevance of perceived symptoms by a given request?
- How can unknown terms be interpreted or learned?
- How to minimize the maintenance cost of specific electronic resources like dictionaries, thesaurus, etc. ?

This dissertation proposes answers to each of the above questions in chapter II section 2, with the development of a correspondence graph based on neural networks [Azarian et al. 2007].

Another aspect of this dissertation is to reduce the work of the authors and in particular when handling ECUs data. When confronted with an increasing importance of vehicle's electronics, it will be useful to import automatically the relevant information from ECUs description files called ODX files. However, at the time being, the possibilities offered by the Authoring System to structure and abstract the large amounts of heterogeneous information the author is confronted to are rather restricted. This implies that one central difficulty to be faced in this task is to find intelligent possibilities for modeling and managing the authors' knowledge so as to find a concrete substitute for tacit knowledge, personal competences and skills of experienced employees. For this the following ambitions are faced:

- Reduce and simplify the authors work
- Support new ECUs functionalities
- Eliminate data redundancies
- Simplify the test edition.

The difficulties and the development of an abstract intermediary native language to model the ECUs data are reported in chapter II section 3.

One major key point in this dissertation is also to improve the diagnosis algorithm. Since some functions need the same information (e.g. speed measurement used for the speed display, cruise control and all less visible functions in which speed is a parameter such as stability control (ESC)), they have in common a certain number of components and therefore of symptoms as illustrated in the example below depicted in Figure I-16. The speed sensor is supposed faulty. If several symptoms are active, the system may indicate the best possible cause of the breakdown at the first rank as symptoms indicate the same object (consequently the *SM* increases). In this case, both the power supply and the speed sensor are suspected by all the symptoms and they have the same *PM* whereas the steering wheel sensor and the ABS would be suspected only once and therefore have a smaller *PM*. Furthermore, if the ABS was defect, it would certainly involve the suspicion of other sensors and consequently this will increase theirs suspicion moments (denoted *SM*).

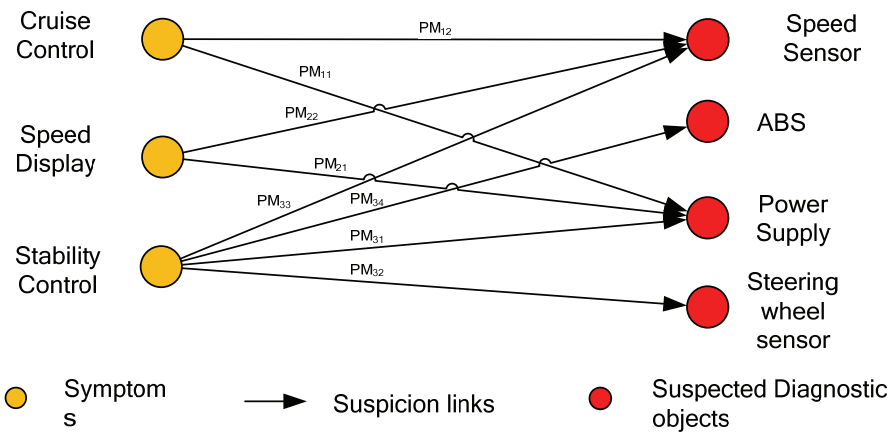

Figure I-16: Diagnostic principle

But what if the power supply is ranked first as the test consists in checking the input voltage whereas for the speed sensor, the test is far more difficult and this leads to a lower ranking? The mechanic will logically test first the power supply and only then the speed sensor. However as we supposed above, the faulty element is the sensor and therefore the mechanic wasted time by checking the power supply. This situation occurs very frequently for real-world cases, and therefore there is a strong improvement request of the current diagnosis algorithm.
The expert systems have a major weakness as outlined by [Smith, Kandel, 1993] and in [Jarmulak et al., 1997]. The practical nature has to do with the complexity of the rules describing the positional relationship between elements in the image. Rules can become problematic and difficult to debug, especially where these involve discussion with expert. Though the problem of rule complexity does not itself prevent the construction of an expert system, it none the less makes the maintenance of such a system difficult. This problem occurs in particular in the medical domain where heuristic diagnostic systems are used with uncertainties on rules, for example the CARPAT Project has classified 43 tumors with around 65000 factors, if a new one is discovered it will be difficult to relate certainty factors to the old one [Schroeder, 1998]. Torasso et al. [Torasso, Console, 1989] express that this approach proved very useful for "the development of systems with good diagnostic performances, in limited domains". He also outlines the importance of the hierarchy "The refinement of the process allows a precise solution to be reached from the abstraction generated in the heuristic match phase […]". Schroeder's version of the hierarchy is more critical (dramatic) "it is for example crucial to fully use the interval of possible certainties to get the best discrimination between diagnoses" [Schroeder 1998]. The classification hierarchy of diagnostic hypotheses and heuristic reasoning corresponds to searching in the frame system for a frame which can be instantiated to actual data. Another powerful aspect of these methods is the extendibility [Davis, 1989] of models with other parameters. For example if the times of the tests are taken into account, the formalization of the problem allows it to find a classical optimization problem where user, effort, convergence criteria on fault localization and time can be traded [Buchaman, Shortliffe, 1985]. Chapter I section 3 reports and discussed about the state of the art in diagnosis methodologies and Chapter III section 2 proposes two new approaches based on deeper investigation of current methodologies that are adaptable to SIDIS Enterprise.

The last point that is finally examined in this dissertation to complete a new framework for the diagnosis is a feedback module to evaluate the current parameters in order to optimize the diagnosis sessions. The tasks that have to be performed by the module are to:

- Optimize the test ranking for given inputs,
- Learn and calibrate the test cost, coverage and uncertainty
- Update the thesaurus for the research of qualitative fault description
- Calibrate the weight for fault codes and perceived symptoms
- Discover and propose new suspicion links or empirical rules to authors.

The learning and knowledge discovering methods are reported in chapter III section 3. A deeper investigation of the reported methodologies and experimentation follows in chapter III sections 3.1.1 and 3.1.2.

3 Current diagnosis strategies

This section presents the current strategies and algorithms in the diagnosis field with a strong focus on the methods that fit within SIDIS Enterprise data structures (without major changes). The first subsections (3.1) report about a well spread methodology called case base reasoning, whereby here only an operative approach applied to car diagnosis is reported.
A heuristic method is an easy way for engineers to represent a diagnostic strategy. This method relies on the inference mechanism expressed by "if…then…else" rules between symptoms and components. Forward and backward chaining via fault symptom trees or approximate reasoning with fuzzy logic leads to the generation of a list of suspects where the suspicion degree depends on the weight of the applied rules. In simple cases where the symptoms directly indicate the faults by a few logical rules, the fault diagnosis is proportionaly simplified. Other parameters can be added in the heuristic method group like; failure rate of components, priority of certain rules in the generation of the list of suspects.

Under the strategies that can be adapted or combined to expert systems there are decision trees which are presented in chapter I section 3.2 with specific metrics like efficiency.
The last subsequent section of this chapter is dedicated to a particular field of model based diagnostics. Model based diagnosis has emerged over the last 20 years [Schroeder, 1998]. This sections reports about ths subfield of model based diagnosis. The general idea is that the model predicts what should be happening in the observed device, and thus discrepancies between the model of the device and the observed device can be used to detect problems and to decide what might be causing these problems. The idea of model based diagnostics [Robin, 2003] arose with the associative formalization of failure knowledge in the works of [Peng, Reggia, 1990] and with the logical formalization of the failure knowledge by [Console, Torasso, 1991]. There are many different methods in the field of model based methods but most of them rely on the diagnosis of [Kleer, Reiter, 1992]. The questions that occur in these methods are the level of representation of the components, the simplification hypothesis from components to models (e.g. models include the fault behavior or not) and at least the representation language (logical, object oriented, networks, etc.). Due to the fact that model based diagnosis increase really significantly the cost of models [Cornelius, 2004], the investigation is focused on the area of causal networks.

3.1 Discrimination of errors by cases

3.1.1 Introduction to case based reasoning

Case-Based Reasoning (CBR) is a way of using past solutions to a particular kind of problem to solve similar new problems. CBR has been enjoying considerable attention over the past few years. CBR may be considered as being analogous to human experts solving a problem by employing their relevant past experience as far as possible. If the new problem has some novel aspects, then the solution to the new problem is added to their expertise. It is particularly suited to application areas where the problems may not easily be decomposed or where the general principles involved are not well understood, but where there is a library of past experience which can be employed.
The four main steps of the CBR engine are:

- Obtain symptoms of the new diagnostic problem (from the user or from another system).
- Match the symptoms with the descriptions of previous problems (often referred to as cases), retrieving previous problems with similar symptoms.
- Perform tests to distinguish between the previous problems that match the symptoms. This might be done by investigating the best matching case first or by ordering the matching cases by frequency of occurrence. In order to accomplish this task, some information about how to perform fault localization and/or fault diagnosis needs to be added to the case-base or to the diagnostic system.
- If the new problem was different from any previously known problem, it should be added to the case-base. In practical diagnostic systems, this is usually not done automatically. Adding cases is more likely to be done during a maintenance phase, where an expert verifies the cases before they are added to the case-base database.

The delicate part of a reasoning engine based on cases is to define which information describes a case and how to compare two cases. One interesting system for the comparison applied to the automotive field is proposed by [Wen et al., 2003]. The architecture is based on a system of Agents of Diagnostic Signals (ADS) responsible for the diagnostic of a given signal (or Signal diagnostic Agents, SDA). These agents make a report of their diagnostic to a Vehicle Diagnostic Agent (VDA) responsible for the analysis of different diagnostics with a case base reasoning.
The case base reasoning by the VDA agent can be realized in two different strategies: the first one consists of the results given by the SDA (cf. chapter I section 3.1.2) and the second is based on the analysis of the characteristics of the information segments given by the SDAs (chapter I section 3.1.3).

3.1.2 Reasoning from global diagnostics

Each agent SDA gives one of the following four results after the analysis of its signal:

- Good (G) if the signal gathered by the agent is correct
- Wrong (W) if the signal gathered by the agent is incorrect
- Undefined (U) if the agent cannot deliver a diagnostic
- No Disposal (ND) if the gathered signal is unknown from the agent

The state of the system composed by *n* agents SDAs can consequently been described by a signature vector $S = (S_i)$, where the *i*-th coordinate gives the diagnostic of the *i*-th SDA agent. In order to have the possibility to compare the signature SD of the current cases with the *k* signatures S_L^k of the cases of the library, a distance *D* is defined which compares each component separately with another distance *d*. The distance *d* must satisfy the following conditions:

- The distance between two identical states is equal to zero :
  $d(G,G) = d(W,W) = d(U,U) = d(ND,ND) = 0$
- The maximal distance must be reached for the difference between the states "Good" and "Wrong":
  $d(U,W) < d(G,W)$
  $d(U,G) < d(G,W)$
  $d(ND,W) < d(G,W)$
  $d(ND,G) < d(G,W)$
- The state "Undefined" is nearer to the state "Wrong" than "Good"
  $d(U,W) < d(U,G)$
- The state "No Disposal" is nearer to the states "Good" and "Wrong" than "Undefined"
  $d(ND,W) < d(U,W)$
  $d(ND,G) < d(U,G)$
- The states "No disposal" is nearer to the state "Undefined" than the state "Good"
  $d(ND,U) < d(ND,G)$

Considering these constraints, a table of distances can be built as for example in Table I-4:

d	*G*	*W*	*U*	*ND*
G	0,0	1,0	0,7	0,6
W	1,0	0,0	0,3	0,2
U	0,7	0,3	0,0	0,5
ND	0,6	0,2	0,5	0,0

Table I-4: Distance between the different states

With this reasoning, some hypothesis had been made concerning the distances to the states *ND*. Another reasoning for the distance table construction could have been to ignore these states in the distance calculation between two signatures and then to normalize the distance by the ratio n_s / n_{ND}, where *n* represents the dimension of the signature vector *S* and n_{ND} the number of *ND* states that occurred in *S*.

3.1.3 Diagnostic from wrong signal segments

Another strategy for the resolution of the diagnostic problem consists in the extraction of certain characteristics of the wrong segments of the signals with the techniques of the wavelet (allowing to decompose a signal with the help of basic functions called wavelets) and compare them with the ones stored in the library. The distance $d(S_{SDA},S_s)$ between the information S_{SDA} given by an SDA agent and

the information S_s stored is defined as being smallest between to wrong segments of each information, cf. (Eq. I-5):

$$d(S_{SDA}, S_S) = \min_{B_{SDA} \in S_{SDA}; B_S \in S_S} d(B_{SDA}, B_S)$$

(Eq. I-5)

The distance between two wrong segments B is calculated as an average of the discrepancies between their n_c characteristics f_i as given in (Eq. I-6):

$$d(B_{SDA}, B_S) = \frac{1}{n_c} \sum_{i=1}^{n_c} \frac{\left| f_{SDA,i} - f_{S,i} \right|}{\left| f_{SDA,i} \right| + \left| f_{S,i} \right|}$$

(Eq. I-6)

3.1.4 Comparison of both methods

Both previously presented methods (chapter I section 3.1.1 and 3.1.2) are compared with a test base of 13 vehicles with symptoms which had not been stored in the library. A convergence with a similar known case is realized by a case based reasoning. With the considerations that a strategy returns correct result if the similar case retrieved in the library encompasses the same causes as the real causes of the problem. The method reasoning of global diagnostics delivered better results for the performed tests. A strong advantage of these methods is that they cover a wide spectrum of phenomena, because the physical interactions between components are not interpreted. The strategy consists only in the calculation of the distance between a vector or an information segment and the ones stored in the library. This data processing allows to deliver results in a short time interval, the operations on vectors (also with a large dimension) can be done within a very short time with today's hardware performance. Moreover, this is the reason why the case base reasoning can be found within a lot of maintenance software of complex systems as reported by the IDS project for the diagnostics of aircrafts by CBR (where the symptoms are sent when the aircraft is in use).
Nevertheless, both strategies are complicated to implement in SIDIS Enterprise, also if a superposition between VDA and symptoms could be envisaged as well as a compilation of the suspicious rules in a SDA base. The edition of the rules would be done during the compilation step of the update process or directly in the signature vectors by the authors. This last solution changes a lot the nature of the work of the expert. Moreover the second drawback is the combinatory explosion of the dimension of signature vectors: with a base of 1200 symptoms and 1500 fault codes, the dimensions of the signature vectors reach 3000 composants. Finally the signatures are specific to each vehicle and the re-usability of signature segments is not assured for variants of a given vehicle.

In summary, according to [Sandkar, Simon, 2004] the following strengths and weaknesses are recognized for CBR:

- CBR proposes a solution to problems quickly, without having to derive the solution from scratch over and over again
- CBR works well in domains that are not completely understood
- CBR can provide evaluation of the quality of solutions
- Cases are useful in interpreting open-ended or ill-ended problems.
- Remembering previous cases is useful in the warning of potential problems and in avoiding past mistakes
- CBR allows focus on the important features of the problem that led to previous successes or failures

But CBR also presents a series of weakness among the literature that can be resumed with two distinct points:

- Relying on previous cases without expert validation may result in incorrect solutions
- Retrievals of inappropriate cases may result in costly errors and inefficiencies
- For a new system, where failures are unknown it is difficult to populate the CB from scratch.

In brief; case-based reasoning is an excellent way of building diagnostic systems where there is a wide range of possible problems. The CBR cover a large domain, but the algorithm is difficult to implement where there are numerous attributes. Efficient diagnosis also demands investigation of possible failures in a particular order, and at least the CBR needs relevant information about past diagnosis experience. It is not a suitable solution for new types of devices where manufacturers have no ROE about the possible causes and effects of failures [Baumeister, Seipel, 2002].

3.2 Decision trees and AO* algorithm

3.2.1 Overview

A decision tree is a predictive model, that is, a mapping from observations about an item to conclusions about its target value. In these tree structures, leaves represent classifications and branches represent conjunctions of features or failure modes for the diagnosis that lead to those classifications. For the construction of the tree or the construction of decision nodes or tests: the Gini rule splits off a single group of a size as large as possible, whereas the entropy and towing rules find multiple groups comprising as close to half the samples as possible. Both algorithms proceed recursively down the tree until stopping criteria are met. Decision trees had become very popular in the last 10 years; they are used in a wide range of applications as: optimization of an investment portfolio, operation management, decision support or net present value calculation.

3.2.2 Principle

Once the candidates have been generated, the sequence of the tests (diagnostic tree) needs to be found regarding the following two constraints:

- Be able to isolate without ambiguity each fault of the system.
- Have a minimal average cost of diagnosis.

A diagnostic tree is a schedule of tests s_j. It is composed of OR nodes, corresponding to sub-sets of faults f_i that remain to be discriminated. The second type of nodes in the diagnostic tree are AND nodes, corresponding to tests s_j to execute. Figure I-17 is an example of a diagnostic tree that contains both types of nodes.

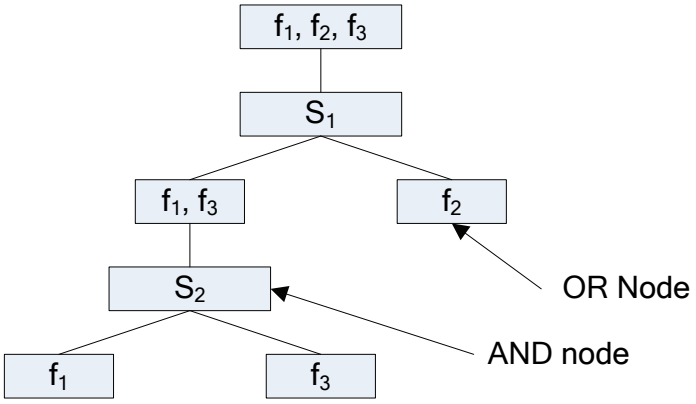

Figure I-17: Structure of a diagnostic tree

In a diagnostic tree, there are 3 manners to regroup a subset of faults [Milde, Hotz, 2001], [Midle et al., 1999]:

- **Grouping by symptoms**: a sub-set of faults corresponds to a specified symptom.
- **Grouping by physical structure**: Faults of a sub-set concern the same component.

- **Minimization of the average cost of a diagnostic:** The AO* algorithm based on the affiliation of AND and OR nodes in the diagnosis tree selects a schedule of tests that aims to minimize the average costs of the diagnostic tree *T*.

The objective function called $K(T)$ is given in (Eq. I-7):

$$K(T) = \sum_{i=1}^{n_l} \left(P(l_i) . \sum_{j=1}^{n_s} d_{ij} c_j \right)$$

(Eq. I-7)

This cost takes the occurrence probability $P(l_i)$ of each leaf l_i (sum of the occurrence probabilities p_i of the faults f_i contained in the leaf) and the cost c_j of the test s_j into account. The boolean operator d_{ij} is equal to 1 if the test s_j conducts to the leaf l_i and 0 else [Olive et al., 2004].

The construction of a tree begins with an OR node as root of the tree, containing the set of possible faults. The tree is then built by adding one by one the available tests and theirs modalities. A certain state being reached, the expandable OR nodes in the tree (a node is expandable if it does not contain isolated faults) are evaluated by a heuristic function *h* and the node *N* for which $h(N)$ is maximal is expanded by the addition of all available AND nodes [Olive et al., 2004]. These AND nodes are then expanded by their OR nodes (the modalities). After the evaluation of the cost of each AND node, the one that remains in the tree (or which will be selected) is the one that had the lowest costs. The cost of the j^{th} AND node is given in (Eq. I-8):

$$c_{ET} = c_j + \sum_{k=1}^{n_M^2} h(N_{OR}^k)$$

(Eq. I-8)

were c_j represents the cost of the test and $h(N_{OR}^k)$ the heuristic evaluation of the k^{th} child OR node.

In order to obtain an optimal diagnostic tree, it is necessary that the heuristic *h* defining the costs of a OR leaf is admissible. This means that the cost $c_{OR} = h(N_{OR})$ attributed to an OR node needs to be systematically inferior to the minimal necessary costs that permit to obtain the leaves with isolated faults from this node. The function *h* can be defined following the method presented in [Milde, Hotz, 2001], which consist in the evaluation of the cost of an impossible diagnostic sub-tree in adherence with the following conditions:

- The cost of the tests is inferior to these of the optimal diagnostic tree.
- All tests are exclusive.
- The capacity of discrimination of the test is superior to those of the optimal diagnostic tree.

The lower cost of the tests is obtained by affecting to the first test the cost of the less expensive remaining test, and for the next level: by affecting the cost of the second less expensive test and so on. In order to assure a higher discrimination capability of the tests in the impossible tree, the remaining AND-test that gives the maximal number (noted *k*max) of OR leaves is considered. It is the allowed for each test to generate *k*max OR leaves. With the notation n_f as the number of faults in the OR-leaf that is evaluated, the number of levels *n* in the impossible diagnostic tree is then given by (Eq. I-9).

$$k\max^n = n_f$$

(Eq. I-9)

The depth of this tree is then consequently leastwise *n*min given in (Eq. I-10) with *E* the floor function.

$$n\min = E\left[\log_{k\max}(n_f)\right]$$

(Eq. I-10)

A lower bound of the necessary cost to isolate all faults from this OR node is c_{OR} given in (Eq. I-11).

$$c_{OR}(n_f) = \sum_{j=1}^{E[\log_{k\max}(n_f)]} c(s_j)$$

(Eq. I-11)

Where the costs $c(s_j)$ of the tests s_j are scheduled in ascending order.

Once the test is joined to the tree, the cost of the tree is updated. The algorithm stops when there are no more expandable OR nodes.

3.2.3 Outlook

The execution speed of the AO* algorithm strongly depends of the cardinality of the set of the remaining available tests because each of the OR leaves has to be evaluated by the heuristic *h*, which needs computational resources. In order to optimize the generation of diagnostic trees by the AO* algorithm, it is elemental to pay attention on means to reduce the number of available common tests. For the optimization of the AO* algorithm, instead of using the set of tests, it is possible to use a sub-set of tests with a cardinality *n*max selected in the set of so far not used tests, in relation with one of the following 3 criteria:

- their costs c_j
- their number of modalities n_j^M
- their efficiency $eff_j = \frac{\log(n_i^j)}{c_j}$

The most absorbing criteria is the efficiency, because it combines the two others. This partitioning is far more efficient than a construction based on the entropy or the Gini coefficient.

According to [Ao, 2008] and [Breiman et al., 1984], decision trees present the following advantages:

- They are simple to understand and interpret.
- They have value even with little hard data.
- They use a "white box" model.
- They can be combined with other decision techniques.
- They are computational cheap.
- There are alternatives to information gain for splitting nodes (e.g. the efficiency of the tests).

But it is possible to get in trouble with over-fitting. This problem can be overcome with Bayesian decision trees or by chi-square testing [Buntine, 1992]. Another problem in the construction of the tree is the test set error, which consists on hiding some data away (due to forward thinking) and once the data is learned the tree predicts too well these data. This explains the discrepancies between future predictions of the tree and observations [Moore, 2008].

The decision tree realizes a differential diagnosis with tests that are mutually exclusive to lead to the failure mode. The main advantage relies in their construction and tests ranking which can be optimized on certain criteria e.g. test costs, test efficiency, or an heuristic metric of admissibility (for example a combination between costs and information gain). In chapter III, a proposed approach for the diagnosis was developed with a function based on the discrimination power of suspicion rules for the weights affectation of symptoms. Compared to the default weight values for the symptoms, this function leads to better results.

3.3 Model based diagnosis

3.3.1 Overview

Model based methods need an interpretation engine to understand and predict the behavior of the model. This interpreter can become very complex; for example to predict the behavior of fluid parameters in a hydraulic circuit the evolution of the parameters and the physical laws of the phenomena that occur are needed as background knowledge. There are several variants of model types that can be built. [Price, 1999] separates them in the following categories which are practice oriented:

- Simple dependency models: they express the causality relation between components.
- State based models: these models describe the possible states of the systems.
- Component based models with or without fault simulation: the models derive from control theory and are very useful for continuous monitoring of processes.

This dissertation is focused on the first type of model, all that is expressed is the fact that the correct operation of one component depends on the operation of a number of others. In essence, what is formed is a tree of dependencies, with the most important components (often the components where the symptoms are observable) at the top of the tree. The core of these models relies on the causality relationship. The fact A → B must be understood as "A might cause B", which is slightly different from a logical implication relation. Here the abductive reasoning is privilege. A simple example of a causality probability network is depicted in Figure I-18.

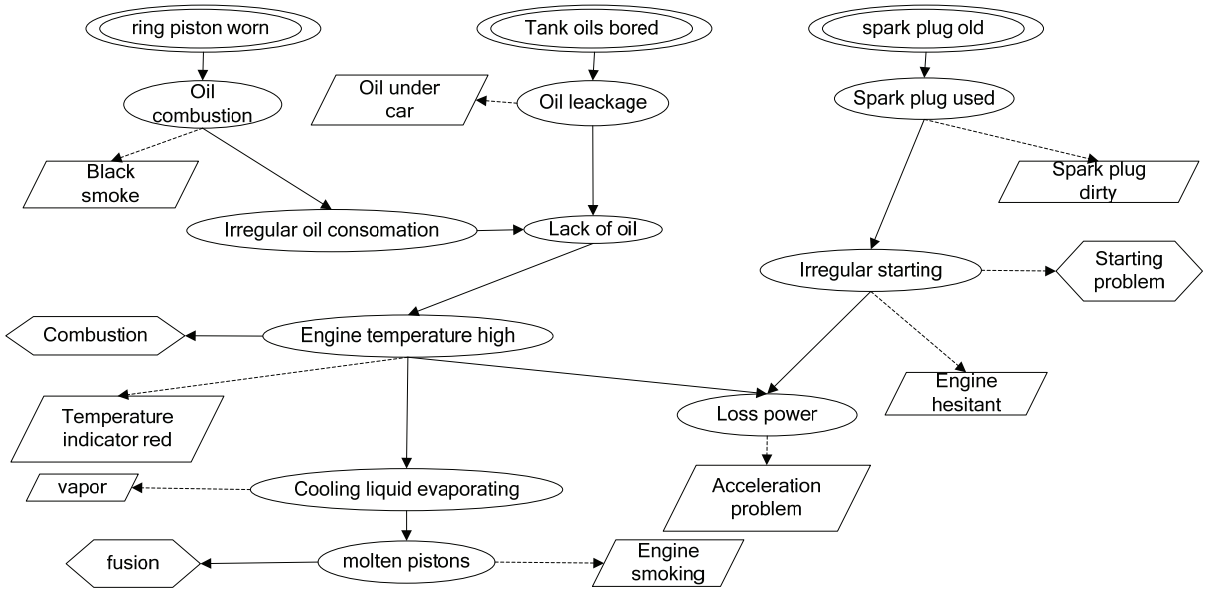

Figure I-18: Causality graph [Robin, 2003]

3.3.2 Formalization

Causality graphs have been formalized in the work of [Console, Torasso, 1991]. They distinguished 4 types of vertex in these graphs; Table I-5 lists the vertices with their characteristics.

Vertex type	Representation
Initial cause (without antecedent)	
Intermediate state: not observable	
Manifestation: observable effect	
Hypothesis: corresponds to a conjunction of intermediate states	

Table I-5: Vertex of causality graph [Robin, 2003]

In this case the task of the diagnostic algorithm is to navigate through the graph generated from the manifestations observed by the technician to the root causes. It is like a depth search algorithm; moreover (as shown in Figure I-18) the edges of the graph can be weighted in order to privilege certain paths leading to more probable causes. The advantages of this model are based around the possibility of obtaining an explicative diagnosis. If <*G*,*HYP*,*OBS*> is the tuple of:

- *G*: causality graph
- *HYP*: the hypothesis of faulty and not faulty components
- *OBS*: the set of manifestations.

The set {*I1*, …, *Ip*} is composed of all the initial cause nodes in *G*, also called "abductible terms". An assignation is a set {*I1*(*X*1),…*Ip*(*Xp*)} where *X*j is an admissible value of the attributes of *I*j. For example, for a cause node "leakage in oil tank" a nominal attribute could be the range between 1,5 and 2 liters.

For a diagnosis problem <*G*, *HYP*, $\psi^+ \cup \psi^-$ > an assignation *W* is an explanation if the following two conditions are satisfied:

- *W* covers the set of observable manifestations (noted ψ^+) :

$$\forall m \in \psi^+, G \cup W \rightarrow m$$

- *W* is coherent according to the set of unobservable manifestations (noted ψ^-):

$$\forall m \in \psi^-, G \cup W \rightarrow m$$

The set of the explanations are the diagnosis of a problem <*G*, *HYP*, $\psi^+ \cup \psi^-$>. If the edges of the graphs are weighted by probability indices then the relationship between 2 nodes A and B must be understood as "A may cause B with a probability of x%". These semantics allow the handling of uncertainties and can be corrected, for example with Bayesian networks. [Kleer, 1986]
However, the introduction of this probability and 'supposition notion' causes the diagnosis to be characterized into 2 states:

- If the set of explanations *W* is confirmed and, if for each element of *W*, a supposition alpha is confirmed; then the diagnosis is also confirmed.
- Else the diagnosis is called parsimonious

The difference that occurs rely on the coverage of the observed manifestation of a diagnostic problem and if it covers the set of observation. In this case the diagnosis is based on the coverage. In the other cases, if the set of observed manifestations of a diagnostic problem is empty then the diagnosis relies only on the **coherence of the graph**.

An important enhancement of this causality graph is to add the knowledge of normal behavior and failure modes for the components. This makes sense for complex systems where the causality relations are estimated at around 1000 (for simplified models of cars), and the search algorithm may take a great deal of time to explore the graph. The fault localization process can be improved where the search algorithm asks (at runtime) for the confirmation of abnormal observations. This allows it to incriminate or discriminate sub-graphs and leads to far greater efficiency in the determination of root causes. Despite all there is still important information missing in these models, they are just a chain of causes and consequences. The nature of the components, and therefore the type of fault that exists can not be deduced.

3.3.3 Outlook

As seen the spectrum of model based diagnostic strategies rely on principles that the behavior of the system to be diagnosed is expressed by behavioral models of its components and run with a domain independent model interpreter. This last group of strategies is very different from the rule based or case-based methodologies due to its reliance on declarative logic, such that:

- The model can be at an abstraction level hiding unnecessary details
- It allows easy maintenance and extension (a requirement from introduction)

- The model gives valuable insight into a system that supports predictions or simulations
- These strategies allow explicative diagnosis

In comparison the other presented strategies have only surface knowledge compared to MBD, which rely on deep object model knowledge [Ueno et al., 1992]. These methods are particularly well adapted for the diagnosis of digital circuits as outlined by [Borgelt, Kruse, 2005] and therefore are adapted to the increasing part of electronics in vehicle. Again, having said this, the method also suffers from several weaknesses:

- The amount of information needed for correct diagnosis: compared to the traditional expert-rule based system, the models have to be enhanced with the causal dependencies between components.
- The topology of the networks depends on the current failure mode.
- Obtaining correct diagnosis for systems where many different physical domains interact.

In causality networks the diagnosis can be considered as a local propagation in networks in order to generate reasonable list of candidates.

3.4 Discussion

3.4.1 Synthesis

After the report of the current state of the art strategies in the field of diagnostics, it can be remarked that the methods based on models have the highest ratio of correct answers by number of diagnosis. Despite this high performance in terms of precision, these methods need finer models for the diagnosis, which means that the overall maintenance costs for the model edition increases. All methods present several drawbacks if it has to be started from scratch with the data edition process. The principle as well as the main drawbacks and advantages of the most common strategies are summarized in Table I-6.

Approach	Principle	Advantages	Drawbacks
Expert System	Rules: if... then...	Easily operational	Construction, maintenance, reusability
Case based	Similarities	Learning, decomposition, update	Need a full case base and a similarity metric
Bayesian networks	Causal links and probabilities	Take incertitude into account	Determination of the probabilities
Neuronal Networks	Based on solved cases	Learning, stable, flexible	Training basis, result interpretation
Decision trees	Test series	Visualization, interpretation and comprehension	Re-construction of the tree
Model based	Comparison between real and simulated values	Flexible, extensible Reasoning on complex systems	Precise models
Fault model	Comparison of fault modes	Quick fault localization	Need all fault modes
Markovian models	Established on the basis of component's states	Possible prognostics	Evaluation, recognition of learning

Table I-6: Comparison of major strategies

The CBR methods as the SDA strategy could be implemented in SIDIS Enterprise and the signature vectors of the known case pre-compiled during the publication process. Actually fault code and symptoms as SDA agents with the characteristics active or inactive a series of candidates can be assigned to each case. These signature vectors can be automatically indexed by a vehicle version index based on the validity of the suspicion rules for the specific vehicle variant. But the main problem is that a lot of experts are working on different sub domains of the vehicle models and it is nearly impossible to assign precise causes to a particular signature and vice versa for a given breakdown. A discussion between experts has to be scheduled in order to determine the possible coordinates of the signature vectors. Moreover, the precision of these methods decreases once a new aspect is encountered in a diagnostic problem.
The decision trees are a very powerful approach because they consider the diagnostic as an ordering of the tests which can then been optimized with the implementation of an objective function. Moreover, the workflow of this procedure fits completely in SIDIS Enterprise's guided fault finding modus. The algorithm is also easily implementable into SIDIS Enterprise but it still remains very abstract because it realizes an optimization depending on the criteria used in the objective function, which causes problems for the results'interpretation. Finally the causal networks belong to one of the most interesting strategies because they are easily implementable, compatible with SIDIS Enterprise knowledge's structures and they allow to simulate breakdowns. The strong advantages rely on the fact that there is no modeling or interpretation of the physical interactions between the components but they still contribute to a self-explained and interpretable diagnosis.

3.5 Conclusion

In view of the balance between the cost to establish the models and the global quality of the diagnosis, the domain of the optimization of the test ordering in the guided fault finding procedure will be investigated in chapter III section 2.2. This study is based on a short report of the experimentation with different heuristic metrics for a multi-criteria optimization (cf. chapter III section 1.2.2). On one hand, this strategy can help to increase the efficiency of the current algorithm if the objective function combines the different parameters (depending on their availability) in a way that makes sense. The construction of an objective function was reported in the meta-heuristic approach (cf. chapter III section 2.2). On the other hand, the realized simulation with a meta-heuristic approach investigates the impact and importance on the diagnosis of the criteria or factors that are put in balance.
The diagnostics by causal networks is also one of the MBD strategies that shows promising results and the possibility to build a simulation module. Moreover if implemented in SIDIS Enterprise, this strategy allows to cross the chasm between a traditional expert systems based technology and the MBD technology. Despite that they are qualitative and an abstracted model of the physical interaction they allow to model the impact of a faulty component. As a matter of fact, the vehicles electronics subsystems become more and more interconnected and the reasonable over cost in the modelisation phase the causal networks prove of value with the defined objective in the scope of this thesis. A horizontal or vertical combination with the current algorithm of SIDIS Enterprise can allow to bridge both technologies. But there are still some questions that need to be answered for such an integration as well as for the decision tree based approach:

- How to ensure the propagation through the causal network?
- How to optimize the combination of the candidate generation between the expert rules and the causal relationships?
- How to build an objective function that orders the test ranking that minimize the path of a guided fault finding procedure

These questions and scientific challenges are addressed and discussed in chapter III (cf. chapter III section 1). Chapter III section 2 reports on the tests of an approach based on the decision tree and a combined heuristic and MBD approach.

Chapter II: Symptom search and ODX exchange file processing

1 Introduction

SIDIS Enterprise (see Figure II-1) is a tool for the diagnosis of car breakdowns and is used by technicians in the after-sales network of car dealerships. It is based on the acquisition of fault codes (the tree on the bottom left in Figure II-2) from the car's ECUs (Electronic Control Units), by a vehicle communication interface. The acquisition of the fault codes allows the determination of which component is defective. Moreover, in order to maximize the information input for the initialization of the diagnosis session, the client can describe the perceived symptoms (the tree on the top left in Figure II-2). The system utilizes a perceived symptom database, like "the engine is jerking after starting during the cold stage". The diagnosis algorithm uses both sources of information to initialize the diagnosis session and localize the component which may be broken. The selection or acquisition of some default codes or symptoms initiates the expert's rules which propagate the inspection of car entities which are modeled in the diagnostic tree (tree on the right in Figure II-2). For the incrimination or discrimination of suspected objects, tests (associated to diagnostic objects) are performed. Because of the large amount of data in the diagnostic object tree, expert rules are used to speed up the localization of the fault by the use of the perceived symptoms. The test can involve communication with ECUs or consists of visual verifications like "Is the loss of contact with the battery due to rust and oxidation?" Depending on the results of these tests, the diagnostic algorithm proceeds forward with a deeper inspection of the diagnostic tree.

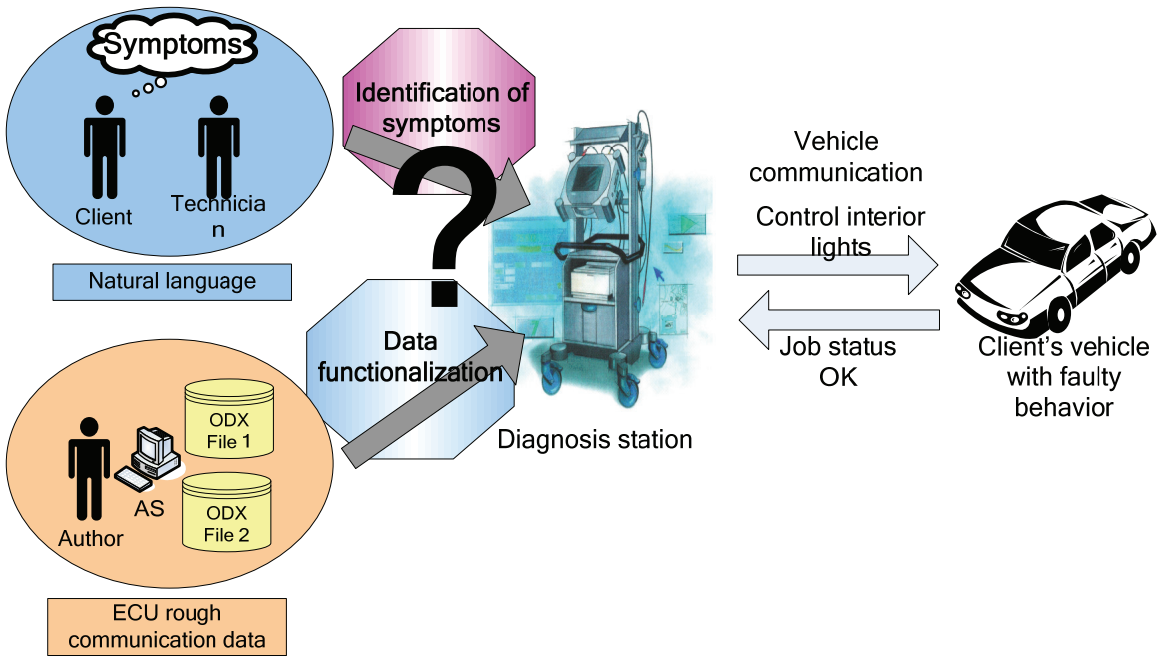

Figure II-1: Diagnostic station [Manuel utilisateur SIDIS-E, 2004].

The first issue of this chapter is concerned about the identification of the perceived symptoms given by the client to the technician in the workshpos. The perceived symptoms are organized in a tree depicted in Figure II-2 (on the top left) and grouped according to observable functions in such a way that the path from the root to the leaf of the tree is a symptom. Other intermediary nodes in the symptom tree allow symptoms to be regrouped by functional categories and to facilitate user navigation, with the drawback that inexperienced users may loose time navigating the tree towards the selection of the desired symptom. In the last years the amount of information has strongly increased (around 2000 leaves in the symptom tree) with the consequences that technicians do not select any perceived

symptom in a diagnostic session regarding the complexity and the amount of information in this tree (with the consequence of protracting the diagnostic session, because less expert rules are initiated, and more intermediate tests need to be performed). To speed up the diagnostic session and accelerate the search process through the symptom tree, the system is equipped with a search module in natural language with a dedicated interface. This chapter aims at reporting the experiences gained in the realization of a domain specific information retrieval module on the diagnostic system: SIDIS Enterprise (developed by Siemens AG) which realizes the mapping of sentences in natural language on the symptom tree in French, German, and English. We will address the main challenges about the machine interpretation of natural language. Chapter III section 3.1.3 reports about the development of a method that permits to use the feedback system to improve the computation of the results.

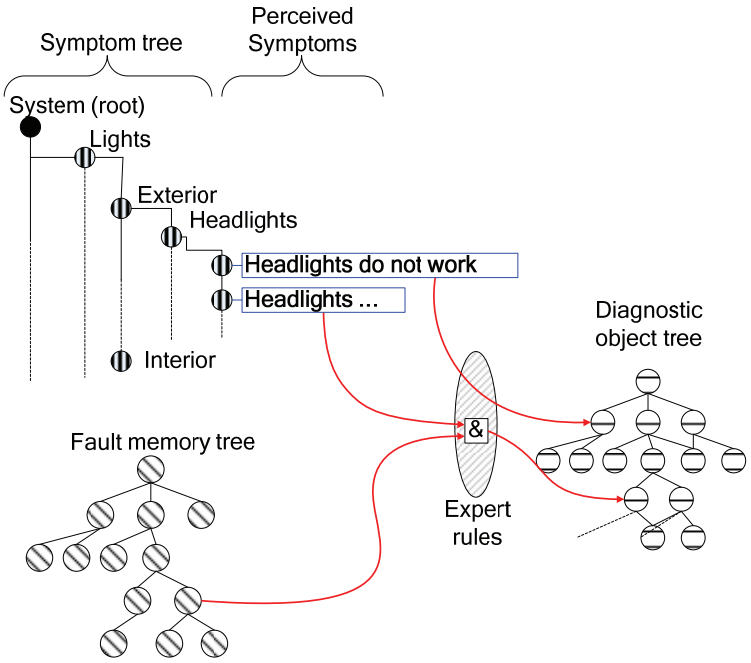

Figure II-2: Interaction of diagnosis knowledge

The second issue of this chapter is concerned with the fault memory tree (on the bottom left in Figure II-2). The information about the ECU data is used in many different knowledge structure of SIDIS Enterprise in particular in the fault memory tree, in the function tree and in the tests. Both the ODX information structures and the actual Authoring System from Siemens AG have been developed in order to manage the great amount of data in the best possible way. In both cases, two main strategies have been identified to do so:

- Development of concepts to reduce information redundancy
- Development of concepts to make a functional-oriented description of data

The data reduction possibilities have been especially developed in the ODX data model with complex inheritance mechanisms, which imply that the first challenge to be faced when importing ODX files is to find a way to avoid duplicating information unnecessarily in the Authoring System. In the Authoring System, the tree structures form an object-oriented database which also offers possibilities to reduce data redundancy. The second challenge when trying to reduce data redundancy concerns the functional description of data. Hence, a special effort has to be made when importing the ODX information in order to understand the meaning of the data. The following points lead to the meeting of objective 3 with the reduction of the cost of the development of diagnosis data.

Both issues are focused to improve the usability of the diagnosis station in the workshops with: on one hand the prolongement of the natural language in the software; especially the client's symptoms or faulty behavior on his vehicle which can helps to speed up the diagnosis. On the other hand, to resolve the ECUs variant by an interpretation and abstraction of the rough data communication files, in order to simplify the work of the authors and excecute specific jobs on an identified broken vehicle in the workshop by an abstract call, e.g. turn on interior lights (related to both interfaces in Figure II-1 under the question mark sign).

2 Information retrieval

2.1 Basic Problems of natural language

This section details the principal difficulties of the automatic comprehension of natural language and the human cognitive processes implicated in the decoding of a message in natural language. Figure II-3 is an example of the number of linguistic links involved for a precise, robust and reliable research in natural language of a request.

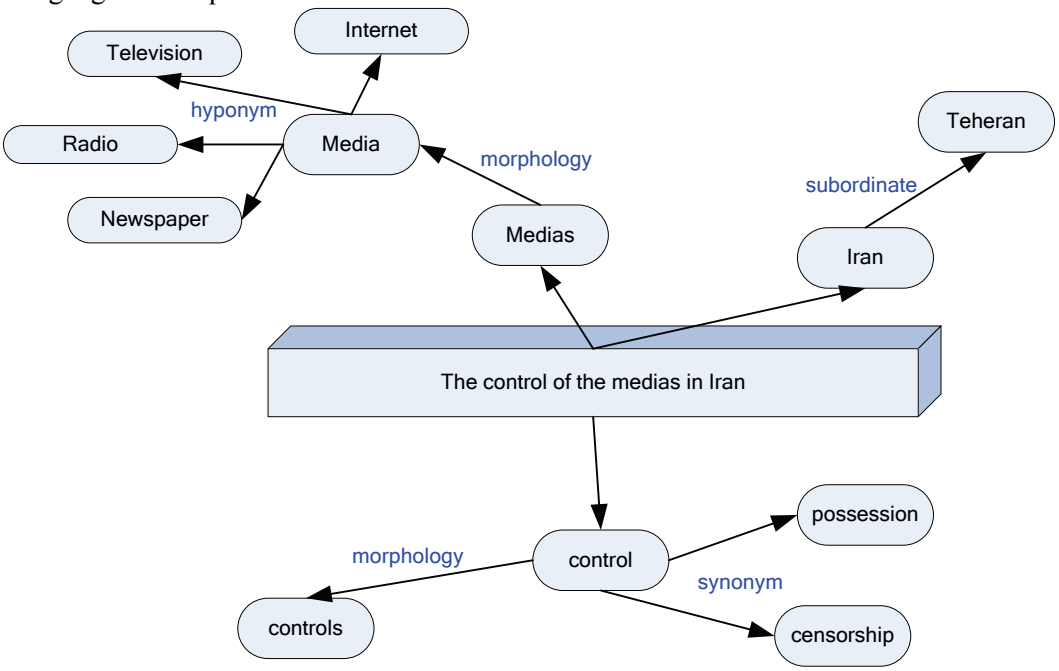

Figure II-3: Linguistic links in a sentence or request [Loupy, 2000].

One common problem of all languages is the decoding of the messages. A message contains a lot of implicit information linked to the pragmatic, knowledge level, age, etc. of the sender that can be only interpreted with an analysis of the context in which the message is sent. Another phenomena is the redundancy, when the meaning is gliding due to connotation, the symbolic (metaphoric expression) used in the message. This type of difficulties is generally rectified with the help of large statistics of word's usage and will probably be the hardest scientific challenge for the comprehension of natural language by machines [Lefevre, 2000]. The following subsections detail the inherent properties of natural language which causes problems for the comprehension by machines.

2.1.1 The graphics

A word can be written with a capital letter or can contain typing errors. These characteristics diminish the chances for the word to be recognized by an algorithm. Sometimes it contains also information of the meaning of a word, or for example in French where accent are often put aside it permits to distinguish different words.
Some research algorithm therefore proposes an automatic correction of the request to increase the precision of the delivered results.

2.1.2 Grammatical variants

A word can be used with different grammar etiquettes which impact the semantic and consequently the nature of the concept. For example a verb used as a substantive in a sentence usually refers to the action of the verb.

2.1.3 Morphological variants

Plurals, conjugations, declinations and irregular verbs work towards a partial or complete modification of the word which reduces the probability for a computer program to recognize it. In French the termination of words are very rich compared to German and English. This characteristic is the most penalizing property of the language. In chapter II section 2.2.3 a stemming algorithm is presented that allows to remedy to this difficulty.

2.1.4 Bound expressions

In particular in some specialized domains, this type of expression appears particularly in French. In this case, an automated IR-system has to recognize the expression and not to search for each word separately.

2.1.5 Synonyms, multiple senses, and hyponymy

Sometimes an IR-system needs to consider synonyms of query terms in order to find new sources. But this problem is also related to multiple-meaning of words, source with the same words as in the query but with a different meaning must not be returned by the search engine. Finally the father-child relation (hyponymy) between words needs also to be considered by an IR-System.

2.1.6 Terminology

Sometimes words can belong to a specialized terminology. In that case, an IR-System has to specifically recognize the request.

2.1.7 Segmentation

In particular in German, words are often conjoined to designate a nuance or new concepts. The meaning of a word is not given by the addition of the concepts of the composed words.

2.1.8 Order

A request is not a series of strings but an ordered sequence of ideas and concepts. An IR-System needs to take the order of the words into account.

2.1.9 Summary

The difficulties in meeting these requirements are not limited by purely technical means, but in particular by the precedent list of properties of natural language [Loupy, 2000]. Beside these difficulties, an automated IR-system has also to satisfy other types of requirements regarding its flexibility, reliability, precision and response time.

2.2 Usual Information retrieval approaches

The most difficult task in information retrieval is to find the words which correspond to the informational content of a symptom. Generally it is admitted that the more often a word appears in a document the more important is the concept inside the document. The next paragraph presents briefly how to extract the main concept of a document and of a request. Moreover, the last section defines performance indicators for the evaluation of the IR algorithms.

2.2.1 Informational content

The first approach in IR consists in the calculation of the frequencies of the words. Once this statistical analysis is done, it appears that the more often words are used the more functional the words are. For example for a text in English, the words “of”, “the” “a” have the highest frequencies. But one interesting characteristic appears once the words are sorted by their respective ranking: the law of Zipf, in (Eq. II-1). If the words are ranked in the decreasing order of their frequencies then:

$$Rank \,.\, frequency \approx cst$$

(Eq. II-1)

The distribution of the quasi-constant of Zipf has been made for different words for an extract of *Hammlet* in Figure II-4:

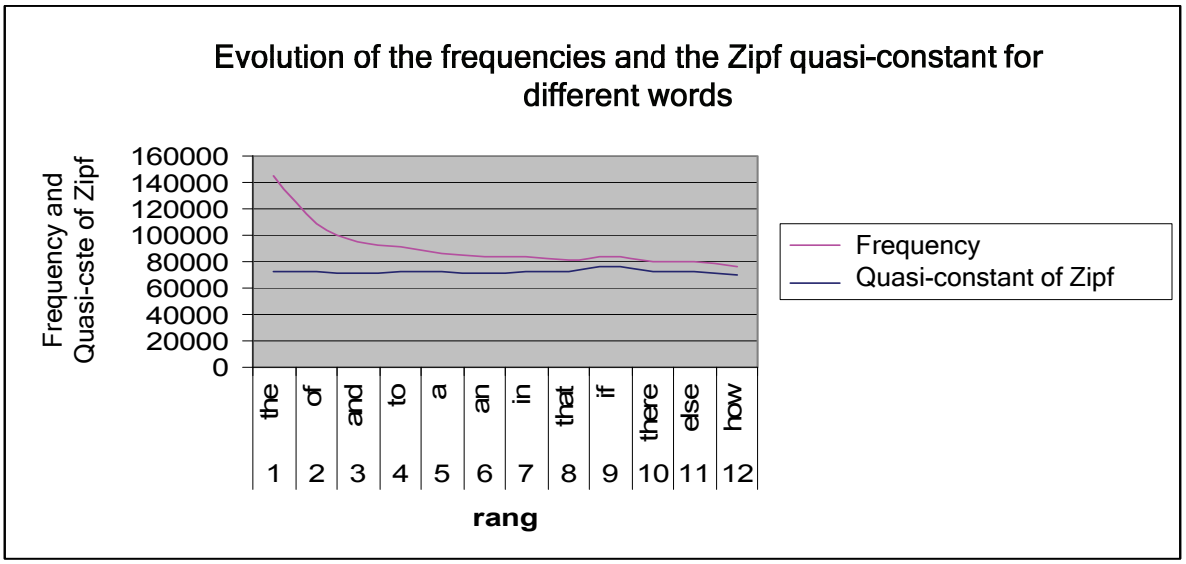

Figure II-4: Evolution of the frequencies and the Zipf quasi-constant

It is trivial that these empty words cannot be taken as description of the informational content of a document. This explains why IR algorithms use filters to eliminate words that contain only syntax or logical information. The Figure II-5 represents the a priori distribution of the informational content of a document in function of the rank of the words.

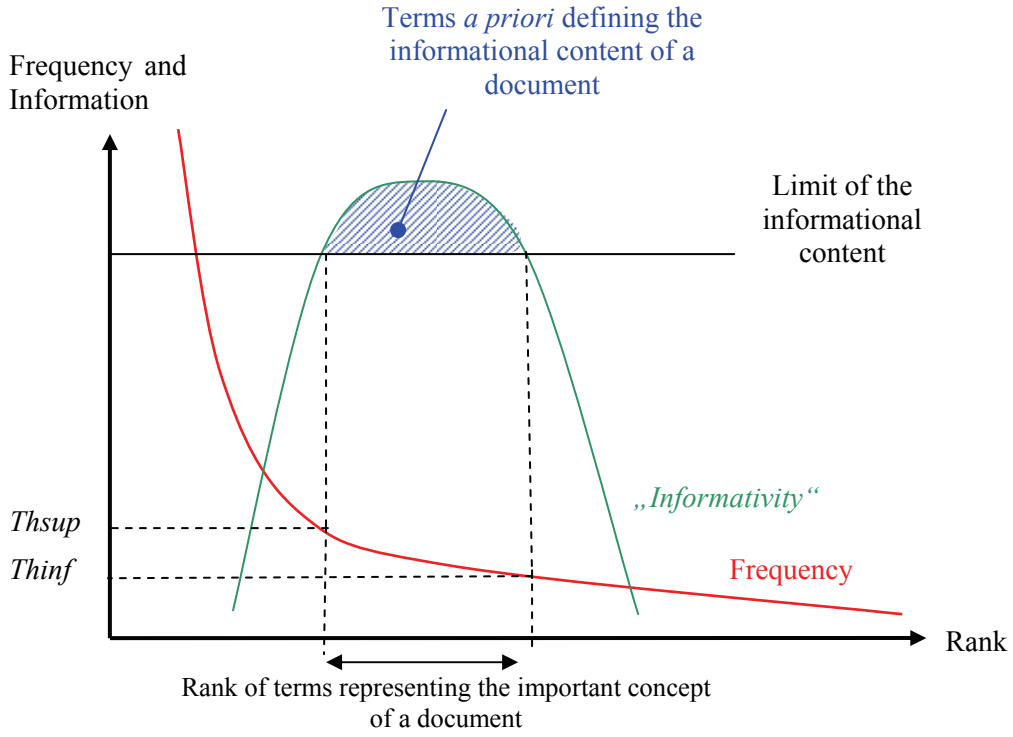

Figure II-5: Informativity in function of the frequency [Nie, 2004].

Two limits appear: *Thsup* and *Thinf* in Figure II-5, the first one is the inferior filtering limit: all the terms with a lower rank are empty or functional words. The second limit filters words that are used too rarely to be considered as key-concepts of the documents. It is possible with these two limits to build an indexation procedure that permits to select the words that describe the a priori key concepts of a document (blue area on Figure II-5).

2.2.2 Performance indicators

As performance indicators considering the standard couple: precision and recall [Risjsbergen, 1979]. Once a request in natural language is specified and the IR-system has been executed, the set containing all symptoms can be divided into 4 subsets (cf. Figure II-6, Table II-1):

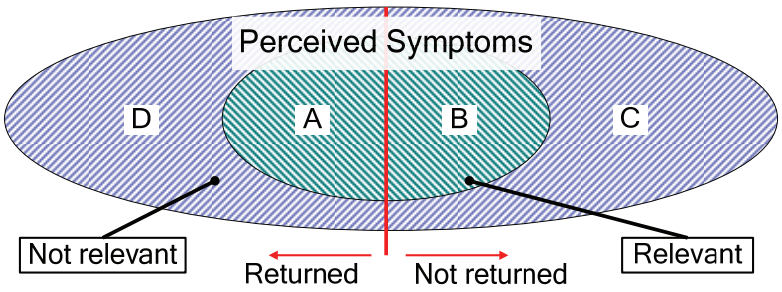

Figure II-6: Subsets of symptoms

A :	Set of relevant documents which are returned by the research algorithm.
B :	Set of relevant document which are not returned by the research algorithm
C :	Set of non-relevant document which

	are not returned by the research algorithm.
D :	Set of non-relevant document which are returned by the research algorithm.

Table II-1: Subset definition.

The performance indicators are defined according to the cardinal of these subsets (cf. Table 1). The precision and respectively the recall are defined in (Eq. II-2) and (Eq. II-3).

- Precision:

$$\Pr = \frac{Card\{A\}}{Card\{A \cup D\}}.100\%$$

(Eq. II-2)

- Recall:

$$\mathrm{Re} = \frac{Card\{A\}}{Card\{A \cup B\}}.100\%$$

(Eq. II-3)

These measures are based on the essential **notion of relevance**. This concept has **no formal definition** and depends on a number of factors. In the literature, many papers describe the relativity of this notion, like [Salton, Lesk, 1968] who report in an experience that only 30% of users agree on the relevance judgment. In 1960, the binary evaluation (yes/no) of the relevance was already criticized by [Maron, Kuhn, 1960]. The evaluation basis of a research algorithm is not stable [Sparck-Jones, 1975], [Mizzaro, 1997], and most of the current IR-systems accord a relevance probability to each document.

2.2.3 Pre processing steps

For the development of an efficient algorithm in order to speed up the diagnosis session, it is important to use certain common language processing methods that help to eliminate the difficulties presented in section 2.1.

The following pre-processing steps are common for all methods applied to both symptoms and queries.

- All text inputs are tokenized. Punctuation marks and symbols are omitted. Format of strings is set to lower case. Special characters are replaced (e.g. ‘é’,‘ç’ respectively replaced by ‘e’,‘c’,…). These steps avoid mismatching words in particular due to the accent in French.

- Queries and symptoms are filtered by adapted stop lists in order to avoid that empty words are taken into account

- Words with more than 4 characters are stripped by the Porter stemming algorithm [Porter, 1980], [Porter, 2005]. This process increases the recognition of words and rectifies the problems due to the morphologic variants.

- A German specific processing step consists in the searching of words over 5 characters in a dictionary. If they are not found, they are cut in different positions in order to consider bound words as 2 distinguished entities. If the 2 fragments are not identified, the original word is kept [Flämig, 1991], in order to increase word matching with segmentation problems.

- Finally, in German, English, French irregular verbs are replaced by their respective infinitive forms, in order to avoid mismatching due to morphological variants.

With this process in natural language a simple request in French like “Problème avec les feux de positions” becomes “probleme feux position”. In this sentence all characters were set to lowercase,

accent and plural terminations were removed. This series of steps permits to partially remove the morphological variant (cf. chapter II section 2.1.3) and to filter the relevant words of a symptom.

A last method is used in certain experiments: the use of a thesaurus. Hodge defines a thesaurus as: « Structured vocabulary based on concepts and describing the links between the different terms. The hierarchical relations (subordinate), equivalence and association are the main considered links. These relationships are often denoted GT for general term, ST for specific term, SY for synonyms and AT for associated terms. For information retrieval the preferred terms for the indexation (2.2.4.1) are defined and all the other terms are linked to preferred terms." [Hodge, 2000]

Replacing the terms of the request by preferred terms may significantly impact the performance of an IR algorithm especially for a short request. It is also possible in certain cases to affect a confidence index for the relevance of a link between two terms in a thesaurus. For example for an SY relationship if term 1 and term 2 are not synonyms of each other, a factor Θ between 0 and 1 can describe the relevance of the SY link. If Θ is equal to 1 then both terms are synonyms each other, if Θ is equal to 0 then there is no link between the terms. This confidence index is later taken into account in the similarity, see section chapter II 2.2.4.1.
For IR purpose a thesaurus is represented as formalized in (Eq. II-4) and if necessary the confidence index is defined:

$$term_i \rightarrow \underset{j \in J}{OR}\left\{\left(term_j \;;\theta_j \right)\right\} OR\left\{\left(term_i \;;1 \right)\right\}$$

(Eq. II-4)

2.2.4 Usual IR methodologies

2.2.4.1 Index search methods

The index search methods were the first developed search methods and have encountered a great deal of success in the first generation of IR-Systems, they are all based on the term frequency approach. By parsing all symptoms and collecting the different stems (word stripped from their affixes), the base of terms (BoT) is build. Representing a symptom as a column vector where all components are the term frequencies of the corresponding term in the BoT, and repeating this operation for all symptoms will allow a matrix (called TF for Term Frequency) representation of all symptoms by the concatenation of all symptom-vectors expressed in the BoT [Salton, 1989]. To estimate the informational differentiation of the symptoms, the index search method introduces a weighting operation on the term frequency matrix by regarding the distribution of a term into all symptoms. There are many weighting formulas, but all use the document frequency (called dFT) which is the number of symptoms in which the term T appears. In this investigation four different weighting formulas were examined:

- *tf.idf* (Term frequency, inverse document frequency) [Salton, 1989] :

$$W(T_i, S_j) = tf_{S_j}(T_i).\log\left(\frac{N}{df_i}\right)$$

(Eq. II-5)

- Harman weight [Harman, 1993]:

$$W(T_i, S_j) = \frac{\log\left(tf_{S_j}(T_i)+1\right)}{\log\left(Length(S_j)\right)}\log\left(\frac{N}{df_i}\right)$$

(Eq. II-6)

- Croft weight [Croft, 1983]:

$$W\left(T_i, S_j\right) = K + (1-K)\frac{tf_{S_j}(T_i)}{\underset{i}{Max}\left\{tf_{S_j}(T_i)\right\}}$$

(Eq. II-7)

With $K \in]0,1[$; Often K = 0.5

- Okapi weight [Robertson et al., 1994]:

$$W(t_i,\ S_j) = \frac{tf_{S_j}(T_i)}{tf_{S_j}(T_i) + A} B$$

(Eq. II-8)

With A and B defined in (Eq. II-9) and (Eq. II-10):

$$A = \frac{\| S_j \|}{\underset{k \in [1,N]}{Average} \{ \| S_k \| \}}$$

(Eq. II-9)

$$B = \log\left(\frac{N + df_i + 0.5}{df_i + 0.5} \right)$$

(Eq. II-10)

Table II-2 describes the parameters used in (Eq. II-5) to (Eq. II-10):

N	Total number of symptoms
$W(t_i,\ S_j)$	Weight of term *i* in Symptom *j*
$tf_{S_j}(T_i)$	Term frequency of term *i* in Symptom *j*
df_i	Document frequency of term *i*
$\| S_j \|$	Length of Symptom *j*

Table II-2: Parameter description

All these weighting formulae accord a high weight to terms which appear in few documents and, inversely, a low weight to terms uniformly distributed across the symptoms. This means that terms that are used in few documents allow to differentiate the document from they content, on the contrary of terms uniformly distributed across the symptoms in order to quantify the informational differentiation introduce by a term. The *tf.idf* formula is the more common; introduced by Salton [Salton, 1989], Harman's [Harman, 1993] and respectively Croft's [Croft, 1983] formula take into account that the global term frequencies are higher in long documents and counter-balances this effect by introducing two new parameters: the length of a document or symptom and the maximal term frequency (Eq. 6). The Okapi formula [Robertson et al., 1994] normalizes the document length (referring to the average symptom length) reflecting the fact that the longer a symptom is, the greater the likelihood that a particular query term will occur by chance (Eq 7). The weighting formula is also applied on the column vector of the query (other terms, as used in the BoT, are ignored). The particularities of each weighting formulae involve a different informational differentiation of the symptoms as:

- Logarithmic behavior for the *tf.idf* formula.
- Pseudo-linear for Harman formula.
- Linear threshold effect for the Croft formulae.

The Okapi formula combines the *tf.idf* behavior for low and high *tf*, otherwise the behavior is quasi linear.

To determine the relevance scores of a query across the symptoms one type of similarity measure called a 'cosines formula' can be used (usual scalar product divided by the product of the norms of both vectors) [Piwowarski, 2003]:

$$SIM(q, W_j) = \frac{< q \mid W >}{\|q\| . \|W_j\|}$$

(Eq. II-11)

Piwowarski provides a comparative analysis of similarity measures in the case of text objects with the following formulae: the cross-product, pseudo-cosines, cosines, Dice and the covering formula [Piwowarski, 2003]. The cosines formula (Eq. II-11) is particularly well adapted for SIDIS Enterprise due to its advantage of taking only the angle between the two vectors into account, parameters like length or components of a vector will have no influence (unlike the Dice or pseudo-cosines formulae).
The last step is to sort out the symptoms by the similarities obtained from the cosines (Eq. II-11) between their query and symptom weight vectors and to fix an acceptable limit for the relevance score. The different steps of the index search methods are depicted in Figure II-7 as a flowchart.

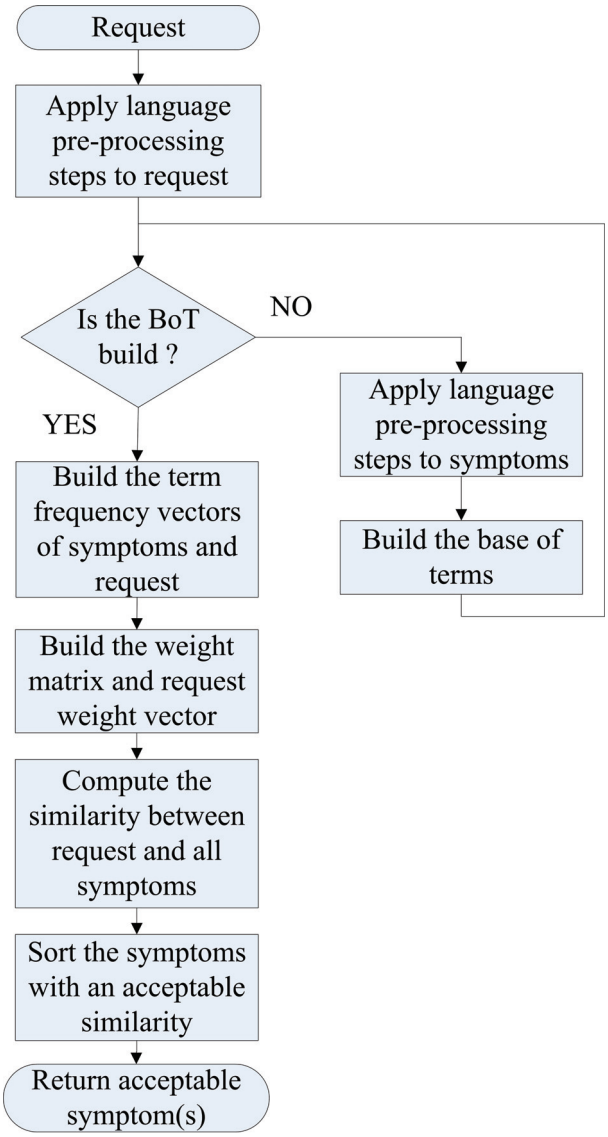

Figure II-7: Flowchart Index Search Methods

The main problem of this approach is that it doesn't take into account the relationship between the concepts expressed in words. The Boolean reasoning used to construct the term frequency matrix allows representation of only a limited vocabulary. The stemming algorithm helps to rectify this issue.

The stripping of the usual affixes increases the chance to recognize two words declined in a different manner [Krovetz, 1993]. But sometimes the stemming adds a semantic ambiguity, for example the stems of the French words “gravitation” and “gravité” are the same but the first word means “attraction” and the second “serious” [Krovetz, 1997]. Another possibility to increase the chance of recognizing words relies on the implementation of a thesaurus, thus enriching automatically the query terms with synonyms (cf. chapter II section 2.2.3), because words in the query that are not in the base of terms will be ignored and thus important concepts of the query could be ignored.

2.2.4.2 Semantic search methods

The idea of a semantic search engine relies on a tree structure where nodes represent concepts from the more general to the more specific. The symptoms are attached to the leaves of the tree. This time the process for the research is slightly different as the precedent one described in Figure II-7. First, the similarity is computed with all the concept of the first layer of the semantic tree. Then depending on a certain acceptability limit the operation is repeated again with the second layer of the tree until the relevant leaves of the trees are filtered. Then the similarities between the attached symptom of the relevant leaves and the request are computed.

Figure II-8 gives an example of the principle of the semantic search engine [Alcides, Schmidt, 2004] for the following request: « Problem with the ventilation of the air conditioning system: the temperature is always too cold”.

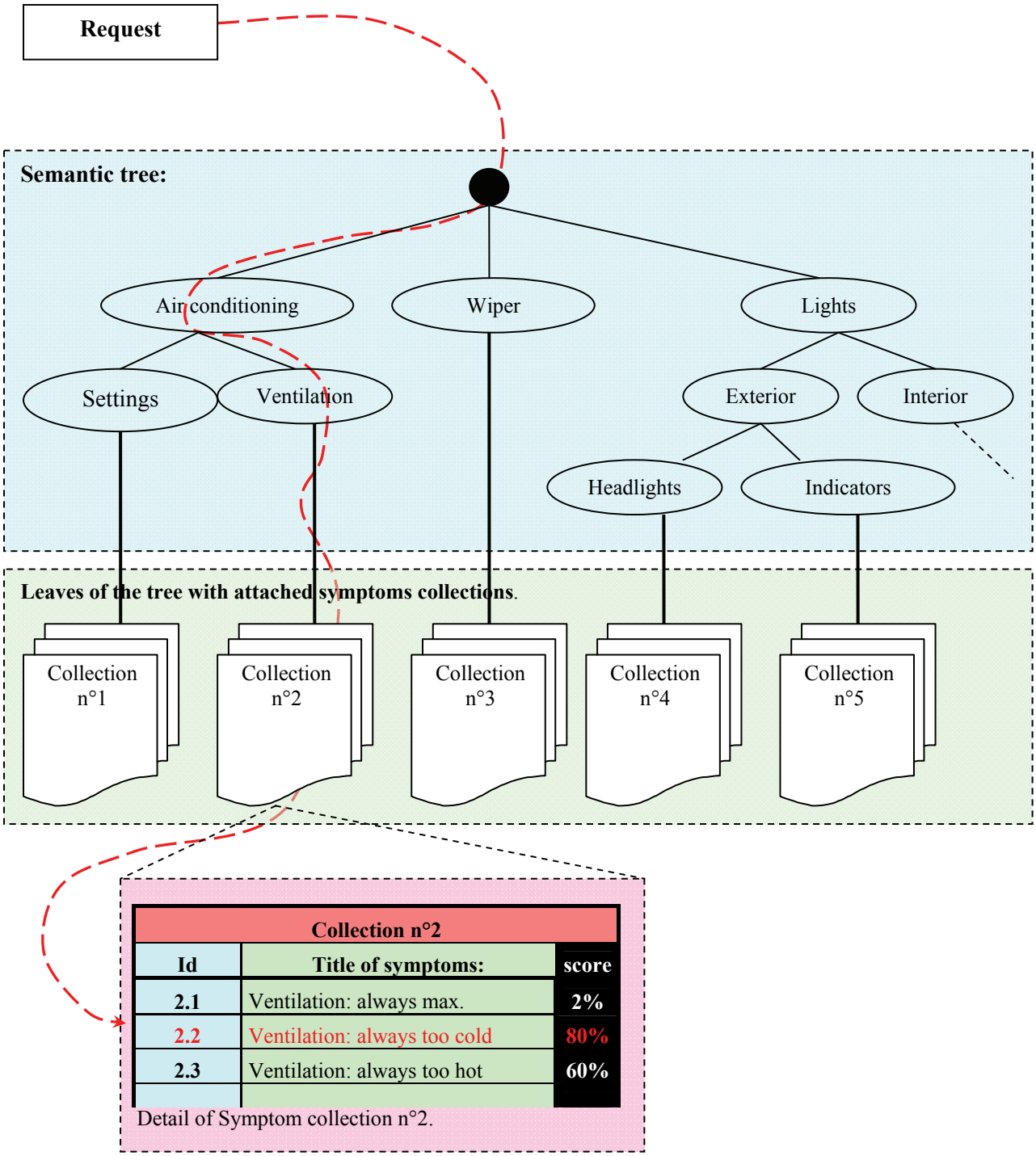

Figure II-8: Semantic search engine

This method is not very different from the index search methods except that terms in the semantic tree are affected with a higher weight to allow the early discrimination of irrelevant nodes and their attached symptoms.

The problem of this method is that the semantic tree needs to be created and maintained by a human operator and therefore the flexibility of the method is far lower than the classical search index method. Another drawback of this strategy is that the semantic tree needs to be re-designed in each different language; a direct translation of the tree may cause symptoms to be classified under an irrelevant label. In terms of performance this method is slightly lower than the index search methods due to the fact that the algorithm needs to navigate through the tree. But the difference is only a matter of some

hundreds milliseconds. Due to the deceiving results provided by this algorithm (less than 11% precision), no simulation based of this method are related in this work.

2.2.4.3 A new approach based on a correspondence graph

For the specific IR-Module of the car-diagnostic-system SIDIS Enterprise a new method has been developed which is based on a correspondence graph.
This approach is inspired by an artificial neuronal network and in particular by Rosenblatt's perceptron [Taylor, 2005]. A two-layer graph is used where the first layer of cells represents concepts and the second layer represents the symptoms. Both layers are connected by weighted edges, where the weight coefficient is obtained by minimizing the errors of the known output values of a training base. The last operation of the 2nd layers cell is a normalization which allows a comparison of the activity levels of the symptoms including a varying number of concepts. An overview of the network is depicted in Figure II-9.

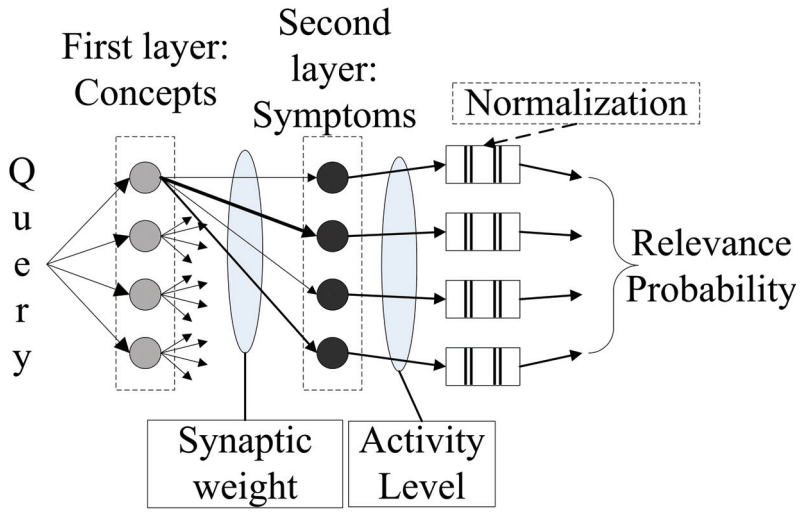

Figure II-9: Overview of the search engine

2.2.4.3.1 First Layer of the graph

The cells of the first layer act as a Radial Basis Function (RBF). The query (after the usual pre-processing steps) is stored in a string array. The cells of the first layer contain different concepts. These concepts consist of one or more words with a similar meaning and they are stored as string arrays. For example the concept "problem" contains in the prototype the array: Problem, broke, not working, defect, etc. (Note that here and for all these sections the considered words in the examples have not been stripped by the pre-processing steps in order to facilitate the comprehension, normally they would be). These concepts are collected manually by an operator. In order to compute a similarity between the query and the concepts used in the cells, the idea is to edit a distance between these two arrays of strings.
Concepts are lists of words expressing the same concept in this context. The idea is to edit a distance between these two arrays. To do this the Levenstein-distance is used to compute the number of different characters between two strings (each different character at the same position of the strings increment the distance by one). An example is given in table 3 between the strings "Firewalls" and "Farewell".

String 1:	F		I		R		E		W		A		L		L		S	
String 2:	F		A		R		E		W		E		L		L		·	
Comparison:	=		≠		=		=		=		≠		=		=		≠	
Levenstein =	0	+	1	+	0	+	0	+	0	+	1	+	0	+	0	+	1	= 3

Table II-3: Levenstein distance.

In the present case, this operation is performed until all strings of a query and of a concept are consumed. This computation returns a Matrix called *A* and is depicted in Figure II-10:

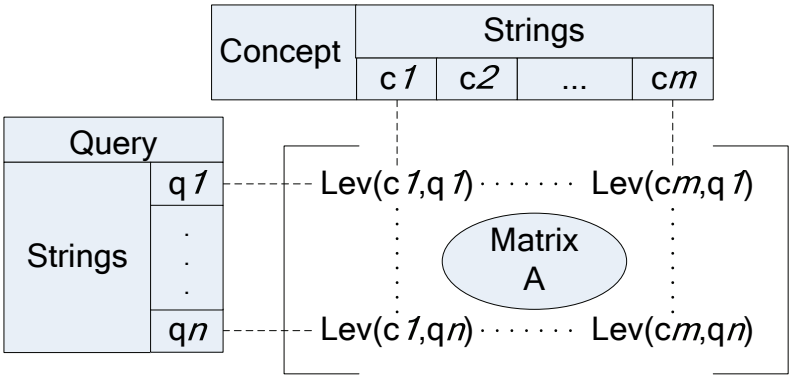

Figure II-10: Levenstein matrix between query and concept string arrays.

The coefficients of the matrix *A* are given by the Levenstein distance between all string combinations of columns and rows. To obtain a scalar number from this matrix (the distance between query and concept noted *dis*), we take the minimum value over all coefficients of the matrix *A*. By including this operation in the process, the third axiom of a distance is not respected since a permutation of the component of the query string array will have the same "distance value" as the original. Thus the operation of the first layer in the network is no longer a radial basis function. The used distance formula is summarized in (Eq. II-12) with *n* as the dimension of the string array of the query and *m* of the concept *c*.

$$dis = \underset{\substack{i\in[1,n]\\ j\in[1,m]}}{Min}\left\{\left(a_{i,j}\right)_{\substack{1\le i\le n\\ 1\le j\le m}}\right\}$$

(Eq. II-12)

This distance is zero if two stems in the query and concept arrays are identical; otherwise the number will be higher. The objective here is to have an output that is high (equal to one) where this distance is zero and approaching zero otherwise. To handle these conditions an inverted *tanh* function is (cf. Figure II-11) applied to the distance to obtain the *synaptic activity*:

$$Activity = \frac{1}{2}\left[1-\tanh\left(\frac{dis-\mu}{\sigma}\right)\right]$$

(Eq. II-13)

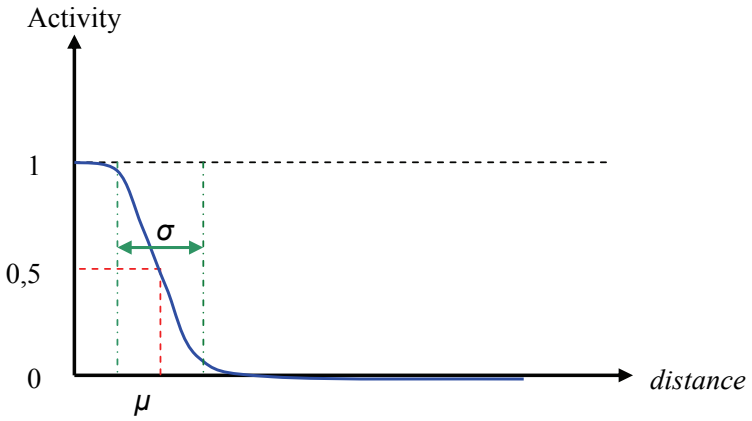

Figure II-11: Activity level with tanh-threshold function

The parameters μ and σ modulates the threshold effect and the distribution respectively (Figure II-11). They represent the accordable tolerance margin, with which can be afforded the assumption made implicitly: i.e. that the symbol strings comprise the semantic information and that this semantic information differs with a lack of the same characters in the two strings. Once we obtain the activity (like a neuronal network) as output of the first layer, it is subsequently propagated onto the second layer.

2.2.4.3.2 Second Layer of the graph

The second layer of the cells represents the perceived symptoms. As depicted in Figure II-9, the propagation of the activity is done by means of a weight coefficient (*wi,j*: concept *i* and symptom *j*) representing the strength of the link between a concept in a symptom. The exact value of the coefficient will be determined later by solving a linear system for known queries and the output by minimizing the error on the output. Then, in the cells of the second layer, all entries are summed to obtain the activity of each symptom for all listed concepts in the first layer including the weight parameter:

$$Out_j = \sum_{k=1}^{first\ layer} w_{k,j} \,.\, Activity_k$$

(Eq. II-14)

k varies from 1 to the number of cells in the first layer. *j* is the index of the symptom in the set of symptoms. Then the output of the second layer is normalized in order to enable the comparison between symptoms calling different numbers of concepts. This normalization is done with the weighting coefficient of each cell:

$$S_j = \frac{Out_j}{\sum_k w_{k,j}}$$

(Eq. II-15)

The central problem is to determine the synaptic weight coefficient by a training base. By listing all the symptoms, all intervening concepts must be collected. This operation allows to quantify the persistence of a concept in a symptom. Table II-4 shows this list:

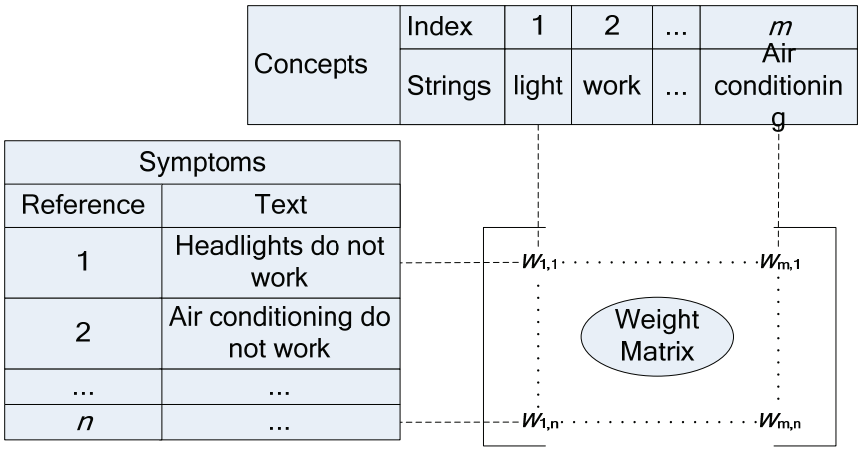

Table II-4: Synaptic weight matrix

There is a possibility to compute the coefficient values with a training basis. Similar to the computing of the synaptic weights of a neuronal network with a training basis, the collection of queries must be collected from the maximal set of varying future users (different ages, car-knowledge, expression forms, etc…) to be particularly polyvalent. By successive iterations the computer can solve the systems of ($w_{i,j}$) ($i \in [1,m]$, $j \in [1,n]$) for a collection of known queries Q_k ($k \in [1,t]$)and outputs ($O_{k,l}$) ($k \in [1,t]$, $l \in [1,n]$) by minimizing the errors of the results obtained from the correspondence graph.
This system will be able to identify the concepts of the query and the weighting of the links between concepts and symptoms. Concepts are the core of the research system since an effective information retrieval system has to determine the probability of relevance between a query and a document. However, the main reason to apply such a system is the uniformity of the class of users. If the class of users were too large and too different, such a system could not be applied to a specific domain; the average error margin obtained after determining the synaptic weight matrix (based on the training basis) would be too large due to significant meaning variations around concepts.

2.2.5 Outlook

2.2.5.1 Discussion

Table II-5 gives a feature comparison of the different presented search methods based on criteria like performance, linguistic ability and implementation effort.

	Methods		
	Index	Semantic	Correspondence Graph
General criteria :			
Precision	Acceptable	Mean	High
Recall	High with stemming and thesaurus and low without	High with stemming and thesaurus and low without	High
Multi-lingual	yes	no (The semantic tree is language specific)	no (The graph is language specific as well as the weight coefficients)
Results	simple	simple	Not evident (black

interpretation			box)
Specificity of the method	Comparison of two terms between request and symptoms adapted for request in natural language	Terms to terms comparison in order to find the relevant symptom collections	Reasoning rather based on perception adapted for request in natural language
Linguistic Difficulties			
Grammatical variations	Not taken into account	Not taken into account	Not taken into account
Morphological variations	Partially taken into account with stemming	Partially taken into account with stemming	Partially taken into account with stemming
Synonyms	Taken into account only with a thesaurus	Taken into account only with a thesaurus	Taken into account if the list of concepts is well designed
Hyponyms	Taken into account only with a thesaurus	Taken into account only with a thesaurus	Taken into account if the list of concepts is well designed
Handling unknown terms	Possible with a feedback algorithm	No	No
Algorithm			
Implementation	Easy	Complex	Average
Complexity	Polynomial	Exponential	Polynomial
Flexibility	High	Low	Very low
Linguistic resources	Stoplist, stemming, thesaurus	Stoplist, stemming, thesaurus, semantic tree and symptom collections	Stoplist, stemming, list of concepts and training basis

Table II-5: Comparison of existing solutions.

Under the general criteria, the precision, recall and the multi-lingual aspect of the solutions were considered. The result interpretation is also an important criterion, because users accord much importance if they understand intuitively how the results are sorted. For example a word of the request could appear in bold into the returned symptoms in order to ground the results. Under specificity of the method, the main characteristics and the reasoning for the engine is described. The two last examined criteria are the linguistic difficulties (morphological variants, synonyms, hyponyms…) and the specificity of the algorithm of the search engines which includes the implementation difficulty in SIDIS Enterprise, the complexity, the flexibility in case of a modification of the symptom tree and at last the necessary linguistic resources like dictionaries, thesaurus, etc.

2.2.5.2 Conclusion

The main features that are particularly important for SIDIS Enterprise according to the objectives are the precision, recall, the answer time and the possibility to run the engine in different language. The

index search engine generates much noise due to the use of a stemming algorithm which decreases the precision. A possibility to increase the precision consists in the coupling of two index search engines where the first one makes a first filtering and the second one searches only in the returned results of the first one. The semantic search engine had a higher precision due to the fact that in the early phase of the search process symptom collections are eliminated. Moreover, the semantic search engines cut the search spaces in functional categories which explain the good precision of the results. The correspondence graph rely more on the perception and analogy for the computation of the results. Both characteristics depend on the weight coefficient of the graph which can be set with the help of a training basis. But the problem of the correspondence graph and the semantic search engine is the flexibility and the multi-lingual performance. If symptoms are added, removed, or changed the corresponding collections of symptoms or nodes for the correspondence graph need to be changed (and eventually the weights need to be adjusted). The structures of the semantic tree as well as the topology of the graph are different for each language. For example in French the concept “position” must have a strong link with symptoms related to the headlights “feux de positions” and the symptom “position of the rear mirror” in the graph. This property is due to the meaning of the bound expression “feux de positions” which makes the network specific to each language. The index search engine can automatically construct the base of terms which present a strong advantage compared to the other methods. Concerning the recall, all search engines have nearly the same score. Two factors influence much the recall: the first one is the stemming which helps to remove morphological variants but also increase the noise. The second factor is the thesaurus which permits to add synonyms or preferred terms and thus allowing a more exhaustive research into the symptom basis.
The complexity of the algorithm is detailed in appendix A section 2.2.2, it is important that the module had a correct answer time. The correspondence graph has the fewest operations; it is the most performing method. The choices for the development of a prototype concern the correspondence graph and the index search method. The next section reports briefly the obtained results of both engines.

2.3 Results

2.3.1 Implementation

This section details the implementation of both engines. Due to portability constraints of SIDIS Enterprise, the module has to be developed in C# to be executed on different platforms.

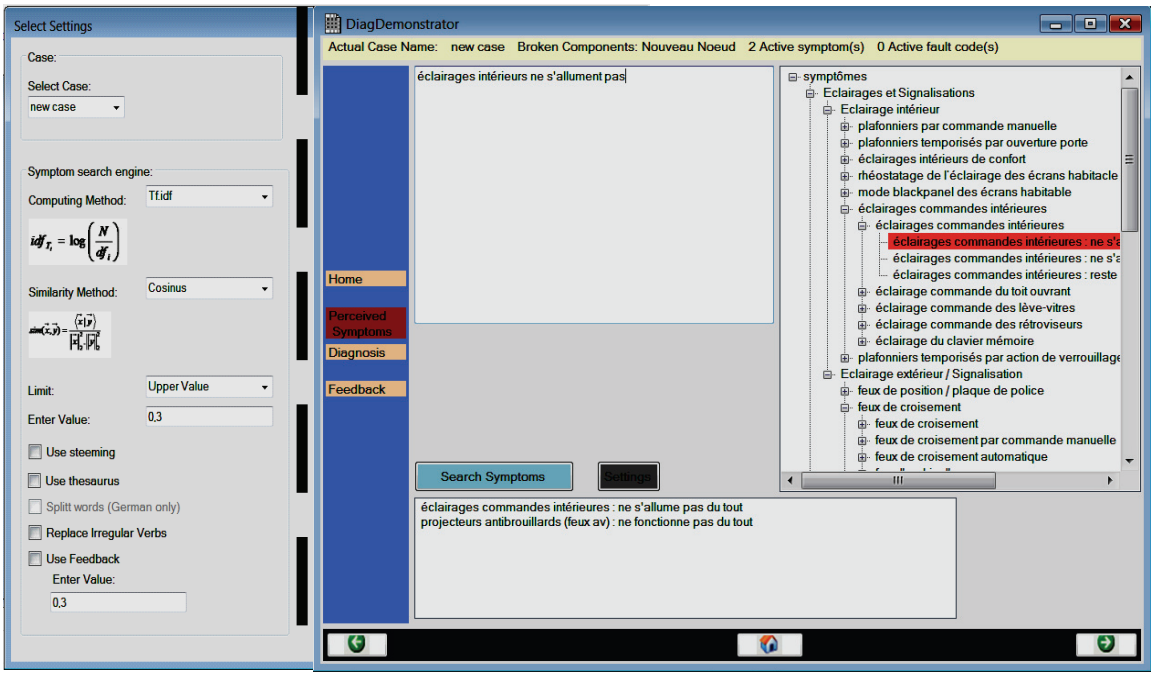

Figure II-12: Settings of the implemented search engines with example

Figure II-12 shows the interface of the prototype. On the left side the following list of settings can be chosen for a search:

- **Computing Method**: For Index search methods the following weight formulas are implemented: *tf.idf*, Croft, Harman, Okapi and the second strategy which consist of the correspondence graph
- **Similarity method**: The following formulas are implemented for the computation of the similarity: *cosinus*, *Jaccard*, *Euclide*, *Dice* and the similarity for the correspondence graph.
- **Limit**: Two possible acceptance limits have been implemented, the first one is a lower bound that the user can fix, and the second one is heuristic.
- **Word processing**: It is possible to use or not the following options: stemming algorithm, thesaurus, replacing of irregular verb, feedback
- **Feedback value**: If the feedback is used it is possible to assign a certain coefficient in order to set the importance of the feedback values. The feedback algorithm is detailed in chapter 3.

For the prototype the simplified information model depicted in Figure II-13 was used (methods relative to the pre-processing steps were not represented). The basis class *Neuronal_Searching* contains all initialization methods in order to get the matrix, instantiate the cells of the graph and get the query. The class *Neuronal_Cell* is used for both types of cells in layer one and two. In the first case, the attribute Concept contains a string array with the different concepts, and in the second case the same attribute contains the symptoms. The attribute *Activity* is initialized with the value 0, after the computation of the distance and appliance of the threshold function, the activity level of each cell in the first layer is stored in this attribute. For the second layer of cells this attribute contains the sum of the weight coefficients of the incoming links in order to increase the computation of the relevance score.

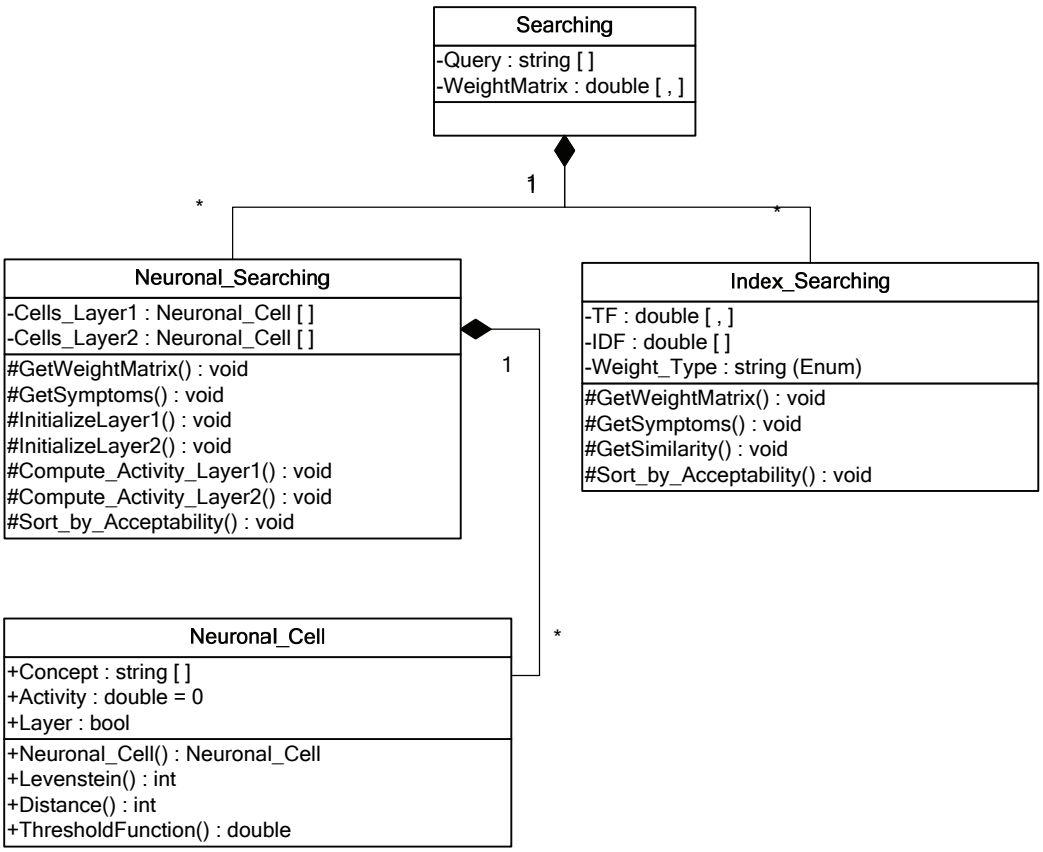

Figure II-13: Simplified UML Diagram

The weight matrix and other linguistic resources (thesaurus, stop list, irregular verbs, etc.) where serialized into files. During the initialization of the search module, these resources are de-serialized.

The weight matrix's coefficients are null for 97.2%, therefore only non-null values were stored in order to save space and speed up the initialization. For the correspondence graph, the complexity of the algorithm depends of the product of the subsequent 4 parameters:

- The number of cells in the first layer called N
- The number of cells in the second layer called M
- The length in words (after pre-processing) of the query called Q.
- The average length of the concept arrays for the cells of the first layer, noted $\overline{c}$.

For the index search methods the complexity is $o(n.m)$ with n the number of symptoms and m the length of the base of terms.
Both strategies were simple to implement in the authoring system. During the publication process the nodes of the graph or the base of terms can be generated and distributed to the workshop system via the update server.

2.3.2 Precision and recall

The precision and recall of the different methods were compared using a set of 100 symptoms, 20 test queries and a training basis of 30 symptoms. By taking into account the methods of calculation used, one will notice that each algorithm assigns a score of similarity to each symptom ranging between 0 and 1. The peculiarity of this module aims to decrease the laboriousness of the selection of the symptoms for the mechanics at the diagnosis sessions. It is desirable to turn over all the symptoms with a score of similarity and to let the mechanic interpret this score and select the symptoms which appear pertinent to him. Consequently, it is necessary to fix an acceptability limit of the similarity beyond which the symptoms are regarded as relevant. Initially, this limit of acceptability is fixed to 0.3 (series n°1). But, during the tests, it can be observed that the similarity curves of the returned symptoms is like a step function, depicted in Figure II-14.

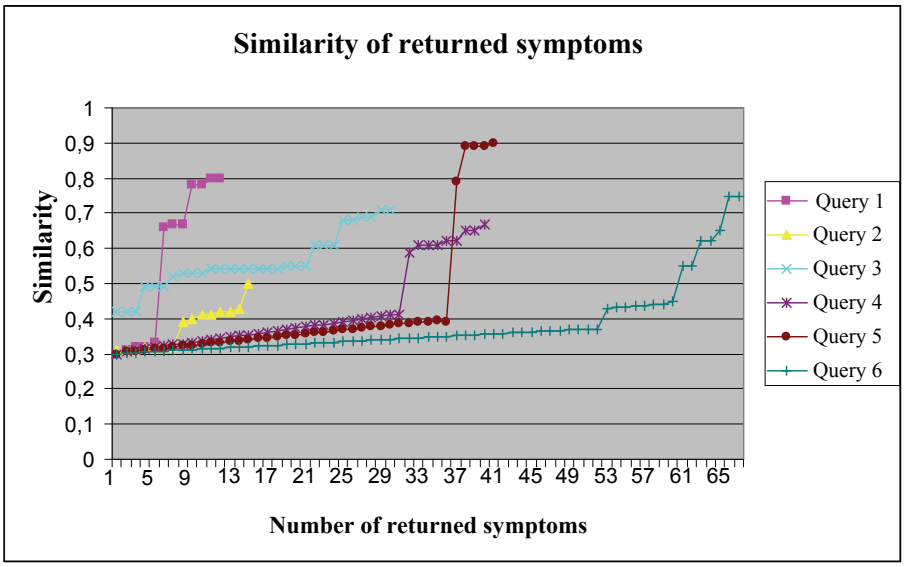

Figure II-14: Similarity of returned symptoms.

It appears frequently that the relevant symptoms for a given query belong to the last steps (with the highest similarity). But the number of steps and their lengths depend on the query. In order to increase the precision and select only the symptoms with the highest similarity an empirical law was implemented and a second series of measurements was performed. Let M be the max similarity value and L the acceptance limit. For $j \in [1,5]$, N_j are the numbers of symptoms with a similarity in the interval $[0.3+(j-1)/10 , 0.3+j/10]$ and N_6 is the number of symptoms with a similarity $[0.8,1]$. If $N_6 \neq 0$ then L=0.8, else let j be the first index so that $N_j = 0$ for j decrementing from 5 to 2. Then the limit is given by:

$$L \mapsto \begin{cases} \text{if } N_{j-1} \geq 20 \Rightarrow \mathrm{L} = 0.95\,\mathrm{M} \\ \text{if } 10 \leq N_{j-1} < 20 \Rightarrow \mathrm{L} = 0.98\,\mathrm{M} \\ \text{else} \Rightarrow \mathrm{L} = 0.2 + (j-1)/10 \end{cases}$$

(Eq. II-16)

The series of measurements give the following average values of the couple Precision and Recall. They are summarized in Table II-6 (where Pr. and Re. stand for Precision and Recall):

		Index search methods				Correspondence graph
		tf.idf	Harman	Okapi	Average	
Series 1	Pr	24%	25%	25%	25%	20%
	Re	96%	96%	98%	97%	30%
Series 2	Pr	44%	38%	43%	42%	60%
	Re	51%	43%	51%	48%	50%

Table II-6: Results of search methods.

2.3.3 Summary

2.3.3.1 Conclusion

In the first campaign of measurements (series 1 in Table II-6) the results given by index search methods prove to be more efficient than the correspondence-graph search algorithm. The similarity score returned by the correspondence graph method gives an approximate constant function which does not realize the sorting between relevant and irrelevant symptoms. But in the second experience, when the empirical law of the acceptance limit is implemented, the new method shows better results in terms of precision with an approximate equal Recall value, compared to the index search method. In this case, the new method fulfilled its role in the distinguishing of relevant and irrelevant symptoms (cf. Table II-6), because the repartition of the similarity scores of the symptoms is taken into account and implies that the returned function beholds a higher deviation fulfilling the sorting of symptoms between relevant and irrelevant ones.
The results of the Croft weighting formula in the index search method have proven a lack of precision (bellow 11% precision) and other measurements made by the coupling of two index-search methods show disappointing results.

The index search methods have the advantage to generate automatically the base of terms and the weight matrix, and this process can be realized in different languages. In contrast, in the proposed methods the concepts need to be collected manually by an operator. And, the structure of the graph is different for each language. Both methods need some linguistic resources like stop lists, affixes (for the stemming process), or a dictionary (for queries addressed in German). The complexity of both algorithms is polynomial, but for index search methods the most important coefficient is higher than for the proposed approach. This has an impact on the response time, which is around 1200ms with the index search methods and around 200ms with the proposed approach. In terms of precision and recall the new method shows better results. This can be explained by the structure of the graph. The weighted edges of the correspondence graph between concepts and symptoms realize an associative relationship which can be compared to a relevance association between a symptom and a concept, with the impact that it increases the similarity of relevant symptoms. This reasoning is well adapted for a specific IR system used **by a similar class of users**. But this method presents also a major inconvenience, the collection of the concepts and the computation of a weight matrix.

2.3.3.2 Outlook

The search algorithm of the first generation based on the vector modeling operates on the boolean comparison of words in order to recognize them. The weight formulae allow to assign an importance to the terms of the documents which permits to distinguish their content. This sort of information retrieval system compounds all languages, a simple query with the word „Information“ could return documents in German, English, and French… But these methods do not handle the resolving of ambiguities of words in the query.
In the second generation of research algorithm the idea emerges of classification by the assignment of semantic etiquettes to words. Other parameters were considered like the popularity of a document in the methods of the third generation based on the construction of ontologies trying to analyze the needs of the user depending on his query. The future of IR-Systems relies probably on the interaction with the user in order to understand the need behind the query.
For SIDIS Enterprise the index search methods are the best available strategy due to the fact that they can automatically handle the multi-language aspect, which is a powerful feature. With the help of a heuristic acceptability limit the overall performance (precision and recall) could be increased and in chapter III. This method will be enhanced with the feedback engine.
Finally, this module allows technicians to easily find symptoms before a diagnosis session starts and thus increases the set of hypotheses. This information gain at the beginning of the diagnosis session can help to earn around 15% performance for the diagnosis with the hypothesis that selected symptoms are always right and in average linked with the correct diagnostic object by suspicion links. This step met the objective n°2 for the increasing of the diagnosis performance.

3 Intermediate language for Automatic sorting of ECUs description files

Due to the shrinking of the lifecycle of electronic devices in modern vehicles it appears that many electronic components change. The electronic field in the automotive industry contains 90% of innovation in order to propose customer new comfort and security options. In this context the diagnosis in the workshop becomes more and more complicated for the testing of new ECUs (cf. Figure II-15). The information about new vehicles needs to be entered in the diagnosis system which generates high maintenance costs. In order to decrease this costs, this section is related to an interpretation engine for the automatic importation, classification of ECUs description files into the authoring system of SIDIS Enterprise.

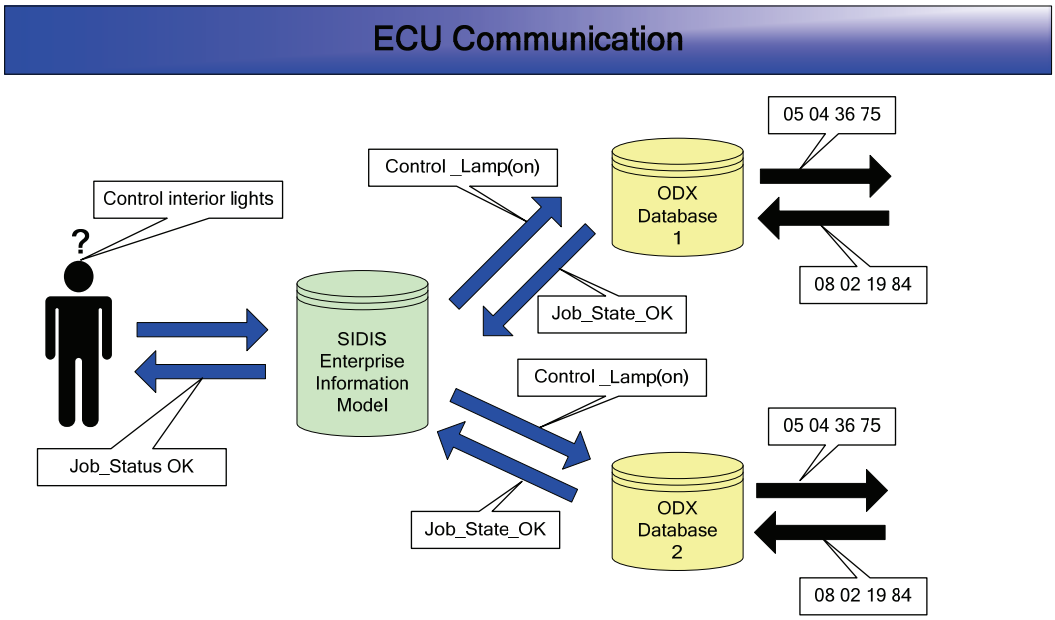

Figure II-15: ECU Communication

This section describes about the development of this module aimed at supporting the work of the author, one key point of this thesis is to get the author to ”know what he knows” so as to make the maximum use of his knowledge when importing new data. However, at the time being, the possibilities offered by the authoring system to structure and abstract the large amounts of heterogeneous information the author is confronted to are rather restricted. This implies that one central difficulty to be faced in this task is to find intelligent possibilities for modeling and managing the authors' knowledge so as to find a concrete substitute for tacit knowledge, personal competences and skills of experienced employees.
Once a meta-level description of information is made possible for the author, the next major issue to be raised is the reuse of this data. The key problem to be faced is to find a means which makes it possible to take the biggest advantage of the formal representation of the authors' knowledge when unknown data is to be imported in the authoring system. Following ambitions are concretely striven for in this module:

- Reduce and simplify the work of the author as much as possible.
- Support the author when new functionalities are featured by ECUs.
- Define each piece of information and of knowledge uniquely in the authoring system so as to eliminate actual redundancies.
- Simplify the writing of diagnostic programs.

Several difficulties to reach these goals have been identified. The first one is linked to the actual data structure in the authoring system, which features data redundancy and does not model knowledge but interprets each piece of information separately. The second difficulty is linked to the ODX data model. On one hand, no ODX data have been used with the authoring system yet and on the other hand, the ODX data model is complex and thus subject to multiple interpretations.
The method for automated ODX-data import in the authoring system which is going to be developed in this thesis is meant to support the author's work in four different domains:

- Transfer of the relevant data for a usable vehicle description.
- Creation and update of the trees in the Authoring System.
- Creation and update of the links between the trees.
- Linking of the fault symptoms and of the trouble codes with the corresponding diagnostic objects.

The subsequent section describes the ODX Data model and a summary of possible approaches which concludes with a short summary. Then, a new approach is proposed based on a multi-agent system.

3.1 Basis of the ODX Data model

ODX (Open Data Exchange) is an international standard which has been established by the ASAM e.V. in order to define the requirements for ECU diagnostic and programming data. It covers the needs of the entire vehicle lifecycle from system engineering to the service shop. The standard [Backmeister et al., 2006] consists of a description of diagnostic data and communication interfaces of vehicle ECUs and includes the UML data model for ECU diagnostic and programming data as well as its XML implementation. ODX files consist of five main parts [Kricke, 2005]:

- **Diagnostic protocols** (DIAG-LAYER-CONTAINER), where diagnostic communication objects, requests and responses of ECUs are defined.
- **Communication parameters** (COMPARAM-SPEC), where the parameters for communication (e.g. timings) are set.
- **Jobs** for more than one ECU (MULTIPLE-ECU-JOB-SPEC), where Java job code for several ECUs making use of the diagnostic communication objects is stored.
- **Vehicle information** (VEHICLE-INFO-SPEC), where topological information of vehicles, physical links and logical links to ECUs are described.
- **Flash data** (FLASH), where memory layout and ECU-programming data are to be found.

The DIAG-LAYER-CONTAINER and the MULTIPLE-ECU-JOB-SPEC are especially relevant for the diagnosis as they describe the ECU functionalities. Further on, their structure is also very interesting as a value inheritance mechanism is implemented to reduce considerably the amount of

data. In order to provide a maximum of flexibility, two different inheritance mechanisms can be used: the Short-Name resolution principle and the ODX-Links. From the most general to the most specific diagnostic layer, the DIAG-LAYER-CONTAINER contains following information layers:

- **ECU-Shared Data**: this layer provides a pool of information, which is available for each ECU.
- **Protocol Layer**: contains data which is standardized through the diagnostic protocol used.
- **Functional-Group Layer**: contains data of ECUs having similar functionalities but dealing with different physical components (e.g. trunk doors and side doors).
- **Base-Variant Layer**: this layer describes a group of ECUs which have similarities (for instance because they are produced by the same manufacturer).
- **ECU-Variant Layer**: this layer contains the description of a real physical variant of an ECU and hence is the most specific one.

The principal information in the Diagnostic Layer is the DIAG-COMM object, which is a diagnostic communication element. The ODX model specifies two types of DIAG-COMMs, the DIAG-SERVICE, which defines the basic communication element and the SINGLE-ECU-JOBS, which consist in Java code and build on one or several DIAG-SERVICES to return interpreted output parameters. A DIAG-SERVICE refers to a request and to one or several responses (positive or negative), which are defined independently from the services. Each request or response is built with parameters, which are to be interpreted with independently defined DOPs (Data Object Properties).
Besides this information, the VEHICLE-INFO-SPEC data also plays a central role during the diagnosis session to identify the in vehicle ECUs using the vehicle identification data and vehicle topology description [Zimmermann, Schmidgall, 2006].

3.2 Interpretation of ODX Files

3.2.1 Scientific challenge

Both the ODX information structures and the actual authoring system from Siemens AG have been developed in order to manage the great amount of data in the best possible way. In both cases, two main strategies have been identified to do so:

- Development of concepts to reduce information redundancy.
- Development of concepts to make a functional-oriented description of data.

The data reduction possibilities have been especially developed in the ODX data model with complex inheritance mechanisms, which imply that the first challenge to be faced when importing ODX files is to find a way to avoid duplicating information unnecessarily in the authoring system. In the authoring system, the tree structures form an object-oriented database which also offers possibilities to reduce data redundancy. The second challenge when trying to reduce data redundancy concerns the functional description of data. The ODX model is set up with objects such as functional classes which make it possible to link diagnosis data (eg. diagnostic services) with their functional description but the SIDIS Enterprise model has to go much further because the signification of each object has to be determined to perform the car diagnosis. Hence, a special effort has to be made when importing the ODX information in order to understand the meaning of the data. The following points lead to special difficulties when wanting to perform an automatic import of ODX files in the authoring system:

- The functional information is not always separated clearly from ECU-specific information, which means that the knowledge is not always described on an abstracted level.
- For some elements, there is a lack of abstracted data.
- Data redundancy.

3.2.2 Possible approaches

As only selected data have to be imported in the authoring system, an automated import of ODX data means defining which information has to be imported and how it has to be functionalized. A filtering of the ODX data files using predefined rules should pre-select the interesting data for the diagnosis. After that a detailed analysis in a functional layer can be run in order to fully functionalize the

considered ODX data. Two possibilities have been examined considering the role of the functional layer:

- The first one consists in **not changing the data structures** in the authoring system. This means that the data analysis would have to feature knowledge discovery methods [Alpar, Niedereichholz, 2000], [Lefevre et al., 2004], which rely on interpretation of natural language [Karanikas, 2000] and possibly link mining techniques [Getoor, Diehl, 2005] or that the functional layer would need a data structure in which the knowledge from the author can be represented, for instance by an ontology. The strength of ontologies consists in their good capacity for knowledge representation but the time spent to formalize the knowledge has to be taken into account, because it rests on cost-benefits analysis, on stability of the knowledge and on the question if knowledge can reasonably be formalized [Liao et al., 1999]. In order to improve the results, ontologies and data mining can be combined using expert knowledge in the data mining process [Brisson et al., 2006], [Brisson, Collard, 2007].

- The second possibility consists **in changing the information structures** in the authoring system which permits to make a reliable functionalization of the data. A possible approach for the realization is to use an approach based on the principle of the ontological based databases [Pierra et al., 2004].

One constraint to be considered is that whatever the model for data in the authoring system is, it shall fit into tree structures as they are the basic information structures in the authoring system. The solution which is going to be developed bases on a generic and ECU-independent representation of the ECUs' functionalities using groups of features directly linked to one characteristic. It develops the concept of ontological based database presented in [Pierra et al., 2004], where data as well as ontological models are supported in two distinct parts of the same database.
The constraints lead to the three part model presented in Figure II-16, which identifies three distinct parts for functional vehicle communication : who, what and how.

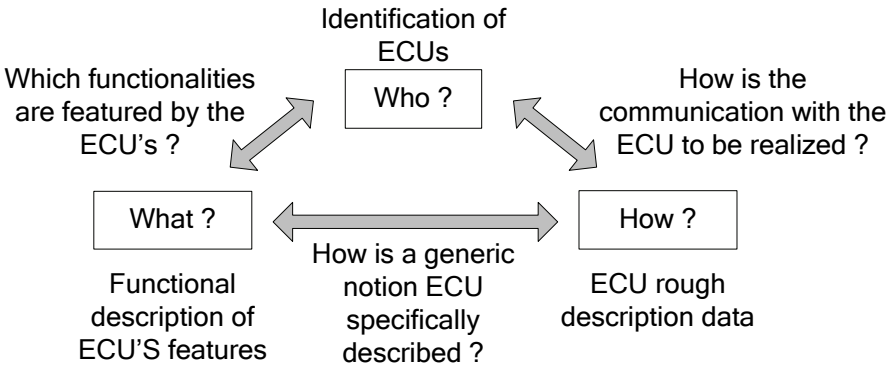

Figure II-16: Parts of vehicle communication

This model has the clear advantage to separate functional information from ECU-specific information and thus makes it possible to have an internal and unified access to ECUs.

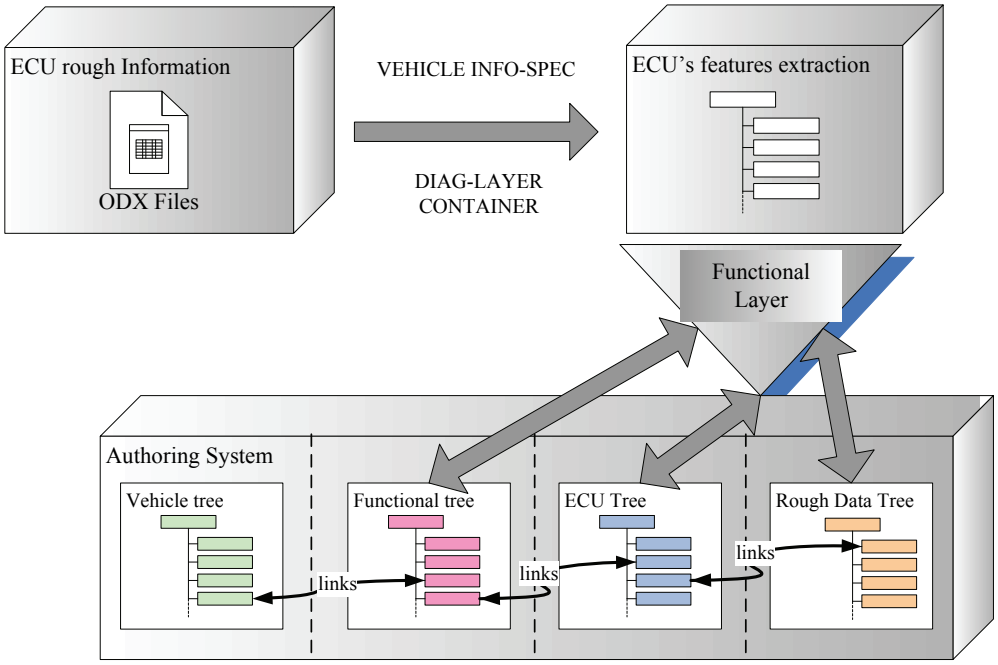

Figure II-17: ECUs Diagnostic related knowledge structures

By the same time, the ECU-specific part benefits from a maximum flexibility to adjust to different ECU description formats such as ODX. This is fundamental in that sense that the structures developed on the ODX side can take the maximum advantage of the ODX structures to reduce information redundancy. Moreover, this structure is interesting because it gives the possibility to:

- Write generic diagnostic program in relationship only with functional information.
- Perform diagnostic session using an unified language.
- Enable automated information import.

At the Functional Layer's level, a module having artificial intelligence capabilities enables to compare the abstracted concepts from the functional trees with the new ODX ECU description data so as to make links propositions between ECU-specific objects and functional objects. So as to automate the import of ODX data, the functional layer should be able to write new nodes in the rough data trees and to create the links to the functional trees. However, its first role is not to set new nodes in the functional trees, so that a read-only access to these trees is modeled in Figure II-17. Of course, when the functional layer is not able to link all the objects contained in the ODX files, it can make new functional nodes propositions to the author.

In order to eliminate completely the data redundancy which appears in tree structures, the proposed information model makes a wide use of Virtual Hierarchy Links (VHLs) in the proposed tree structures. VHLs allow reusing a node or a subtree defined in another tree and thus lead to the definition of virtual parent nodes. So as to be defined, one child node has to have exactly one real parent but can be reused by defining several virtual parents. Using this VHL concept, the following functional trees are introduced and strongly linked together:

- **The request parameters tree,** which contains the definition of request parameters and their possible values.
- **The response parameters tree**: this tree contains the definition of response parameters and possibly their expected values.
- **The fixed functions tree**: this tree contains the definitions of fixed functions.
- **The fixed functions lists tree**: this tree contains nodes (fixed function lists) grouping a set of fixed functions together.
- **Fault texts tree**: this tree contains fault texts in different languages, which give a description for ECU-specific default's codes.

The functional information contained in these functional trees is defined independently from a specific ECU in order not to be duplicate one abstracted element when new ECUs are introduced. The **ODX rough data** tree contains a partial copy from the information contained in the DIAG-LAYER-CONTAINER, and thus is made of an information structure which makes widely use of VHLs. This is for instance useful to represent the ODX-Fields, which are defined independently from diagnostic services. Once the functional data and the ODX data is present in the authoring system, it is possible to establish links between them using the **ECU description tree**, making the ODX nodes turn into "cases of" functional nodes. As the ECU node is the root node of the ECU description tree, it plays a central role in this model. As the functional part has to remain generic, i.e. independent from specific units, a master data tree (**unit tree**) containing a description of unit groups is introduced to convert ODX parameter values into functional parameter values of different unit.

As well as the parameter values have to have a concrete meaning for the ECU, they have to make sense for the technician or the running service program. To ensure an abstract meaning to the values, a conversion has to be possible between a value sent (or received) to the ECU and a value used in the service program. For instance, a French ECU sending back the value "Allumé" has to be interpreted into "On". Considering this aim, the principle of value linking has been developed, where different value types can be defined in the functional trees as well as in the ODX-rough data trees. The value type is defined according to the use case of the value (transmit a random value or convert a value into another one) and its data type (string, numerical, byte field etc.). In the considered information model, the fault's codes have also to be considered as functional information due to their central role in the diagnosis process. However, only the signification of the code is interesting and not the code itself so that the default's code in the considered model consists in:

- A functional fault text, which is the basis for the diagnosis.
- A node containing the ODX data default's code value and its associated ODX text value.

3.2.3 Proposed approach

The data structures which have been introduced make it possible to formalize the authors' knowledge on the one side and to model the ECU description on the other side. This intelligent import module takes fully advantage of these structures when importing new ODX data to:

- Select the relevant data to be imported in the authoring system.
- Propose a functional interpretation of the objects to be imported.
- Identify new ECU functionalities which do not have a generic model in the authoring system yet.

Performing the analysis of a new ODX file is a complicated task which is difficult to execute with a monolithic processing system because of the complexity and the irregularity of the information content. Consequently, a solution using a distributed problem solving method based on a multi-agent system is proposed in this section for the functional layer module. According to the information structure presented in Figure II-17, the role of the multi-agent system is to determine which functionalities are featured by the ECU and to justify its proposal by showing how functionality is mapped on the ODX side. As the data structure introduced in the previous section describes a case base, the cognitive skills of our agents is adapted from a CBR (Case Based Reasoning). Considering the review of case retrieval methods given in Chapter I section 3.1.1, our agents use the nearest neighbour method formula where the similarity between a case C and a query case Q is given by:

$$Sim(Q,C) = \frac{1}{n_f}\sum_{f=1}^{n_f} w_f \sigma\left(q_f, c_f\right)$$

(Eq. II-17)

With σ a comparator and w_f the local weight of the attributes [Wen et al., 2003].

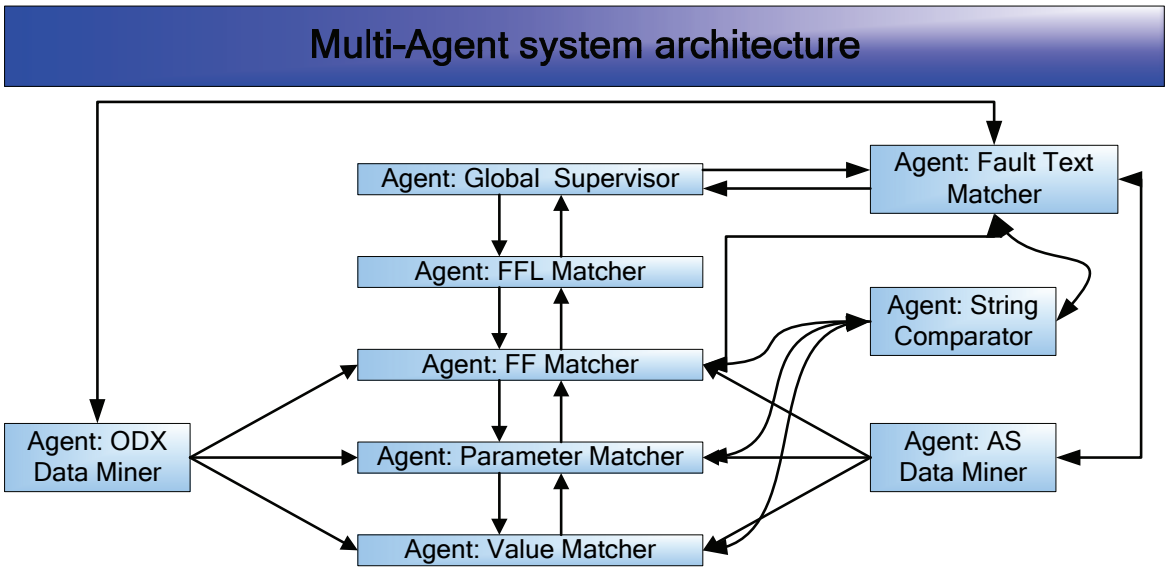

Figure II-18: Multi-Agent system architecture

The multi-agent system depicted in Figure II-18, which has been developed is based on three agent types listed bellow:

- A global **supervisor agent**, whose role is to organize the work of the other agents considering the author's knowledge contained in the functional trees.
- Agents responsible for **finding information** in databases (ODX DataMiner and AS DataMiner).
- Agents capable of case based **reasoning** on specific objects.

The role of each agent is detailed in Table II-7.

The last category of agents is built considering the structures to analyze by following a top down logic: one agent is responsible for determining the match of Fixed Function Lists (FFL Matchers), whereby its conclusions rely on the analysis results from the agent FF Matcher (matches the Fixed Functions). Analyzing a fixed function also means finding a match for the parameters contained into it and their values, which is the role played by the agents FFParameter Matcher and Value Matcher. In this approach, some agents are to be considered as auxiliary agents, which provide a clever access to the data, which is the case from ODX data miner and AS data miner.

Agent	Role
ODX Data Miner	Find information in ODX File
AS Data Miner	Find information in Authoring System
FFL Matcher	Determine if a FFL shall be attached to an ECU Determine the priority of FF analysis
FF Matcher	Determine which new ODX DiagServices match best with a given Fixed Function Determine the priority of the FFParameters analysis
FF Parameter Matcher	Determine which ODX parameters match best with a functional parameter Order value analysis
Value Matcher	Determine how the values of an ODX parameter and functional parameter shall be put into relationship

Fault Text Matcher	Determine if a fault text can be linked to an ECU
String Comparator	Gives a degree of similarity between two string expressions

Table II-7: Name and role of each agent

The role of the agent FF-Parameter Matcher is to determine how the parameters of a new ODX DiagComm match with functional parameters stored in the database. To do so, it receives a new ODX DiagComm and the Fixed Function that is suspected to map the DiagComm functionally from the agent FF Matcher. The first step when trying to find a functional interpretation for ODX parameters is to look up to the information already contained in the authoring system and in the ODX file, so that an analysis order is sent to both AS data miner and ODX data miner to get back a precise description of the parameters. The case representations sent back by these agents consist of several information types:

- Parameter identification data (Short Name, Long Name etc.)
- Parameter description data (Parameter type)
- Functional data (OIDs, semantic)

As all the features to be compared are strings, the agent FF Parameter Matcher requires the services of the agent String Comparator to estimate the value $\sigma(q_f, c_f)$.

3.2.3.1 Principle of the agent StringComparator

A lot of information such as functional classes and object names need to be compared when trying to import new ODX objects. As this information is described in natural language, the agent StringComparator proposes the service of determining the similarity of two expressions to other agents. Chapter II section 2.1 listed the series of problems raised when comparing two expressions in natural language. Here, the following three main causes are faced:

- **Grammatical and word spelling difficulties**, including problems linked to non-recognition of words (morphological variations).
- **Semantic difficulties**, including problems linked to synonyms and polysemy.
- **Multilinguism**, when several languages are used for a same concept.

Consequently, the techniques presented in chapter II section 2.2.3 can be helpful in this situation (e.g. stoplists or stemming). One of the implemented techniques here is the so called “bags of words”. Other techniques such as Inexact String Matching have also been experimented in [Sculley et al., 2006]. They capture some level of sequence information in strings using sequences of contiguous characters in order to detect words in particularly difficult domains such as spam filtering. As the data available in ODX data is not supposed to contain word obfuscation (character level substitutions, repetitions and insertions), the agent StringComparator is based on the “bags of words” principle. Thus, the first work to perform is to identify the words contained in a string.

Once the word lists of two expressions E_1 and E_2 have been processed, the similarity of E_1 and E_2 is computed using the classical Dice Formula (Eq. II-18), because it has the advantage of a normalization of the similarity coefficient independently from the number of words contained in the two expressions.

$$Sim(E_1, E_2) = \frac{2n(E_1 \cap E_2)}{n(E_1) + n(E_2)}$$

(Eq. II-18)

With:

- $n(E_i)$: the number of words contained in expression E indexed by i
- $n(E_1 \cap E_2)$: the number of words two expression have in common

3.2.3.2 Principle of the agent FFMatcher

When receiving an order from the agent FFLMatcher to proceed to the analysis of a fixed function, the agent FFMatcher tries to find the DiagServices from the new ODX file which could be associated to the considered Fixed Function. As the same DiagService is not necessarily used for all the functional parameters, the agent FFMatcher makes DiagService proposals for each functional parameter. Once all the cases *C* of DiagServices in the authoring system and all the cases *Q* of new ODX DiagServices are available for the agent FFMatcher, he computes *Sim*(*Q*,*C*) for all couples request and case noted (*Q*,*C*). This is done using the services of the agent StringComparator to compute $\sigma(q_f , c_f)$ for string values. The comparison of two integers *m* and *n* used for structural description bases on their relative difference as given in (Eq. II-19).

$$\sigma(m,n) = \frac{|m-n|}{|m|+|n|}$$

(Eq. II-19)

When *Sim*(*Q*,*C*) is computed for all the DiagService couples, the agent FFMatcher has to pick up the best matches from the matrix represented in Table II-8 to request a parameter analysis from the agent FF Parameter Matcher.

	DiagService from new ODX file		
From the AS	ODX DiagService 1	. . .	ODX DiagService *n*
DiagService 1	*Sim*(1,1)	. . .	*Sim*(1,*n*)
.	.		.
.	.		.
.	.		.
DiagService *m*	*Sim*(*m*,1)	. . .	*Sim*(*m*,*n*)

Table II-8: DiagService matching matrix

The selection method of the best matches is a crucial step because it determines where the agent FF Parameter Matcher will have to resume the searching and thus which data is going to be compared further on. Hence, two different selection strategies are at the disposal of the agent FFMatcher:

- Blind selection: only the best matches are taken into account and no further criterion for their selection is used. This selection method can lead to focus the next steps of the analysis onto only one new ODX DiagService if all the best matching are contained in the same column of the matrix represented in Table II-8.
- Selection with diversification: this selection method also selects the best matches, but a maximal number of matches per column are allowed for further analysis so as to consider several new ODX DiagServices in the parameter analysis step.

Once the selection is done, the analysis of the selected couples should be deepened through searching for parameters from the selected new ODX DiagServices corresponding to the functional parameters to be matched. The agent FFMatcher finishes his job by sending an analysis request to FF Parameter Matcher.

3.2.3.3 Principle of the agent FF Parameter Matcher

The agent FF Parameter Matcher receives its analysis orders from the agent FFMatcher as a triplet consisting in a functional parameter, a suspected new ODX DiagService and the related DiagService in the authoring system which lead to suspect the new ODX DiagService. This information is not necessary to find a new ODX parameter matching with the functional parameter but makes it possible to guide the search of the agent FFParameter Matcher through reducing the number of cases to be compared.

A similar approach to the one used by the agent FFMatcher is used, as the agent FF Parameter Matcher first looks for information. He first requires the list of parameters linked to the suspected new ODX DiagService from the agent ODX data miner and then requires information about the parameter linking the DiagService from the authoring system to the functional parameter. As all the features to be compared are strings, the agent FF Parameter Matcher requires the services of the agent String Comparator to estimate $\sigma(q_f , c_f)$. A parameter matching matrix can be built on the same principle as in Table II-8.
The matrix of the similarities between the AS parameters and the new ODX parameters shows the list of all new ODX parameters that are potentially in relationship with a functional parameter and indicates their similarity with the parameter from the authoring system which leads to suspect them. Once the similarity between the different cases is established, the agent FFParameter Matcher has to select the parameters which he holds for most appropriate. Similar to the agent FFMatcher, a first possibility is to make a blind selection at the risk of focusing the value search on one parameter from the authoring system and thus on one new ODX DiagService. Hence FF Parameter Matcher has a selection method which is based on diversification so as to keep a maximum of new ODX parameters per parameter from the Authoring System. Another strategy is to lead the parameter selection by taking the belief of the agent FFMatcher concerning the matching of the DiagServices into account so as to enlarge the perception of environment in the decision process [Woolridge, Jernings, 1995].

3.2.3.4 Principle of the agent ValueMatcher

The agent ValueMatcher aims at finding a unique ODX value to link with each functional value he has to analyze. As the agent ValueMatcher is located at the lowest level of the decision process, there is no need to make several proposals so as to explore several possibilities like by other agents. It receives its analysis order from the agent FF Parameter Matcher as a couple consisting of a suspected new ODX parameter and the related parameter in the authoring system which lead to suspect the new ODX parameter.

	DiagService from new ODX File		
From the AS	New ODX Value (1,1)	. . .	New ODX Value (1, *n*)
Value 1	*Sim*(1,1)	. . .	*Sim*(1,*n*)
.	.		.
.	.		.
.	.		.
Value *m*	*Sim*(*m*,1)	. . .	*Sim*(*m*,*n*)

Table II-9: List of new ODX Values which are potentially linked with 1 functional value

As the relationship between a functional value and a new ODX value is a one-to-one relationship, a maximum of one element can be chosen per line and per column in the matrix represented in Table II-9. This affects directly the selection method of the agent ValueMatcher as it determines the links successively by choosing the closest match in the entire table and then retires the couple of values found from the eligible candidates list. This step is repeated until each functional value has its counterpart on the ODX side or until each ODX value has its counterpart on the functional side, depending where fewer values are available.

3.2.3.5 Mapping the results

The agents FFMatcher, FF Parameter Matcher and Value Matcher do not have a large enough view of the environment to determine if a fixed function can be executed by a new ECU to be imported in the authoring system or not. The intellectual and communication capacities they have been endowed with (see chapter II from section 3.2.3.1 to 3.2.3.4) only make it possible for them to make proposals for the best counterparts for functional elements so that they store their mapping results so as the agent FFL Matcher can interpret them.

The saving of the interesting mapping results is done step-by-step by using an XML shared file. Once the complete mapping results have been put into the shared file, it is the job of the agent FFL Matcher to interpret them so as to decide how to pursue the analysis in a first time and then to decide if a fixed function list is appropriate or not to describe the functionality of the new ECU.

3.3 Conclusion

This approach has presented an overview of the data structures in the actual authoring system from Siemens. The preliminary study of these structures have lead to the conclusion that an author realizes a selection and a functional interpretation of the ECU-description data when importing it into the authoring system and introduces new knowledge-related data for each ECU to be described in the authoring system. This procedural method does not help for the automated import of ODX data because it means that the first step before processing the import is to get to "know what the author knows", outgoing from widespread knowledge. Thus, techniques making it possible to formalize knowledge and retrieve it have been presented. In order to take advantage of the data reduction possibilities offered by the ODX data model, new information structures have been developed using the concept of value inheritance and of virtual hierarchy links. With these data structures for functional vehicle communication, the author's knowledge is represented independently from the physical ECU description in one unique database, which makes the development of distributed problem solving method based on an analysis of the ECU capabilities robust and easy. The multi-agent system which has been developed uses a case-based approach to analyse the new ODX data and to automatically import propositions.
Three main ideas, which are being developed at the present time concern the introduction of reinforcement learning techniques, the improvement of the pro-activeness of agents and the possibility to give them alternative cognitive capacities.

4 Summary

This chapter has described the development of two modules. On one hand, a search engine in natural language and on the other hand an ODX interpretation and classification engine. The first module allows saving costs during the diagnosis due to the fact that selected symptoms speed up the diagnosis process. With the actual system users have to navigate through the symptom tree and select their symptom manually which leads to bypass this step and begin the diagnosis session with less information. The benefit of this system is specially related to the quality of the symptoms and their links to diagnostic objects. The information gain at the beginning of the diagnosis session can help to earn around 15% performance for the diagnosis with the hypothesis that selected symptoms are always right and in average linked with the correct diagnostic object by suspicion links. This step met the objective n°2 for the increasing of the diagnosis performance. The selected symptoms at the beginning of the diagnosis session allow discriminating more efficiently the suspected candidates and thus avoiding unnecessary tests, which save times. Moreover, this module is build in order to maximize the interoperability with other language, and can re-compute the base of terms in case that symptom have been removed, modified or added. The automatic index procedures are time saving and the structure of SIDIS Enterprise composed of the AS and WS allow to centralize the procedure only once in the AS. A last advantage of this strategy is that it does not need much linguistic resource: *stemming* algorithms, dictionaries, stop lists can be used from the open source world.
The functional model which has been elaborated is a powerful tool to reduce redundancy in data and in description of knowledge elements in the authoring system. The relationships between functional and ECU-specific part have also been minimized using the value-linking principle. This has been done in order to avoid link redundancies, but it is obvious that it leads to more elaborated path searching in the graph to retrieve information, which can lead to slowing down the system. To improve the performance at runtime, one can for instance introduce links between functional parameters and ODX parameters but to the expense of obtaining a database model which does not respect the normal forms anymore.

The multi-agent based module is far more interesting in terms of cost saving for the maintenance of the diagnostic system. Currently operators are interpreting the ODX files and sort them into the different tree structures. The automatic processing of this task met the objective of reducing high post-development cost for the maintenance of models. This holds true for the programming of test sequences. No quantified evaluation of this module could be made in terms of cost or performance due to the fact that it was difficult to access manufacturer's data.
Finally, both module presented in this chapter are parts of the developed modular framework for the diagnosis developed in this thesis and cover the objective of increasing the overall diagnosis performance and saving maintenance costs.

Chapter III: Diagnosis Algorithm

1 Current Technology

Managing an interconnected knowledge as the ECUs in automotive requires an efficient diagnostic system which is difficult to realize in practice because the amount of data and the number of possible problems and combinations are huge. As the technology used in a vehicle evolves, the diagnostic system needs a continuous amelioration. The general problem addressed in this chapter is how to improve the diagnosis algorithm and combine the different available sources of knowledge for a time performant fault localization (see Figure III-1).

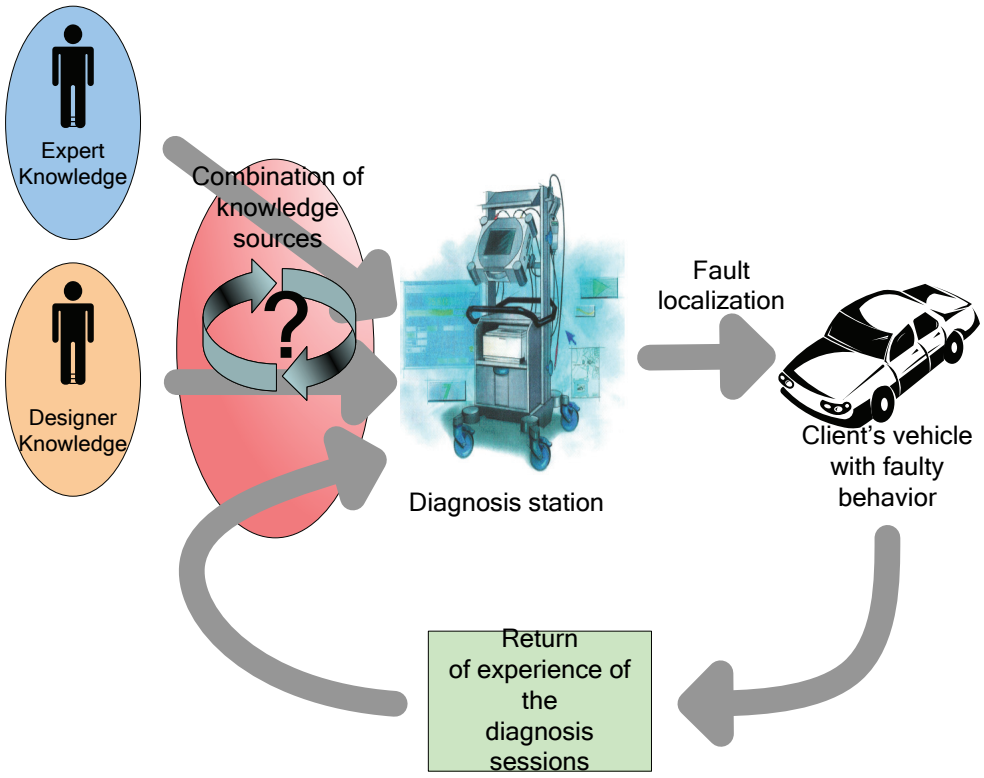

Figure III-1: Combination of knowledge sources for efficient diagnosis

The first part of this chapter describes the actual diagnosis algorithm of SIDIS Enterprise and its limitations. The second part of this chapter details how the algorithm is optimized through meta-heuristics with small modification of the information model. The last part examines strategies which need to extend slightly the information model and auto-complete the models through a feedback strategy in order to evaluate the return of experience.

1.1 Current Situation

SIDIS Enterprise is based on an inference interpretation engine for the guided fault finding modus. In this chapter, the studies are only concerned with this mode, which guides the user step by step to the faulty component. SIDIS Enterprise also has an expert diagnostic modus, but this functional module was designed for other requirements (e.g. verification of a new vehicle series in the factory, post-production tests, data verification and validation) from car manufacturer more than the normal use in

the workshops of the after-sale network. The workflow of the guided fault finding procedure is depicted in Figure III-1.

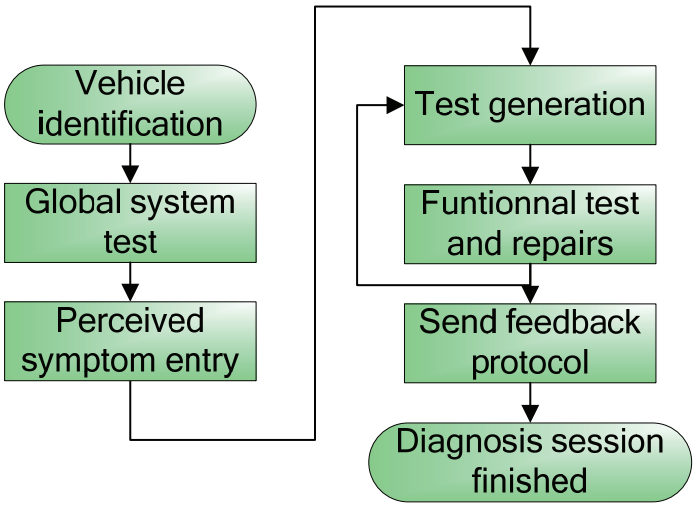

Figure III-1: Guided fault finding process

The first steps allow to filter the data and select only the equipments and components that are present on the vehicle. The second step consists in the global system test where all ECU of the car will be addressed and fault codes will be read. Then the technician can select perceived symptoms from the pre-defined symptom tree via the search engine detailed in chapter II. The test generation is made on the basis of the fault's codes from the ECUs of the car and the selected perceived symptoms. The 7th most suspected components of the car are shown to the technician and he selects to refine one of elements of the list. Once this cycle of "establish and refine" is finished and one or more reparations are made, a feedback is sent to a central server (which is detailed in chapter III section 3) and the diagnosis session is finished. The generation of candidates is detailed in the next subsections.

1.1.1 Candidate generation

The candidate generation process is given in Chapter I section 2.6.2, the main equations of the calculation of the suspicion moment are given in this section. The prime test agenda with the 7 most suspected objects is generated depending on the number of suspicion rules pointing to a diagnosis object and their respective rank:

$$PM_n = GS\frac{(k+1)-RSL_n}{\sum_{i=1}^{k} RSL_i}$$

(Eq. III-1)

With:

- *GS*: the weight of the symptom
- *PM*: the partial moment or partial suspicion degree induced by a rule
- *k*: the number of rules from the symptoms
- *n*: index of the considered suspicion rule
- *RSL*: rank of the considered suspicion rules

Once the suspicion moments denoted *SM*, given by the sum of all partial moments of the diagnostic objects have been evaluated, the objects are ranked by a heuristic metric *h* given by:

$$h = \frac{SM_x}{Tc_x}$$

(Eq. III-2)

With:

- *SM*: the total suspicious moment (or the sum of all partial moments)
- *Tc*: the test cost for the precised element given in index

Once a test is executed, three different scenarios can occur:
The result of the partial moment depending of the test results is given in the following list:
"NotOK" declaration: Computation of the partial moments for each hierarchical relationship in the diagnostic tree (type "Is composed by[*j*]" relationship), depending on the total number of "is composed by" relations called *NB*(*DOx*) is given by:

$$PM(DOx, DOx\_Child[i]) = GDO.\frac{(NB(DOx)+1-i)}{\sum_{k=1}^{k=NB(DOx)} k}$$

(Eq. III-3)

With:

- *GDO*: the weight of the diagnostic object
- *PM*: the partial moment or partial suspicion degree induced by a "is composed by" relation between two diagnostic object nodes
- *i*: the child index
- *NB*(*DOx*): the number of child relation of a diagnostic object node *DOx*

"OK" declaration: All child objects of *DOx*, which for the corresponding test is giving the result OK are decelerated as OK (except if the confidence of the Test is <100% see point 3). The OK state of a node can only be changed if the same test is repeated or another test explicitly declares that the node state has been changed to "NotOK".
"?OK" declaration: If the test result is "OK", but the test confidence is less 100%. The explicit suspicion due to the list of suspected objects *DO*[*i*]. The evaluation of the partial moments for each diagnostic object in the list of suspects is given by:

$$PM(DO[i]) = GT$$

(Eq. III-4)

With:

- *GT*: confidence of the associated test to the considered diagnostic object node

1.1.2 Use case

In this section the diagnosis tree with 30 diagnostic object nodes depicted in Figure III-2 as well as 11 symptoms and 28 suspicion rules are considered:

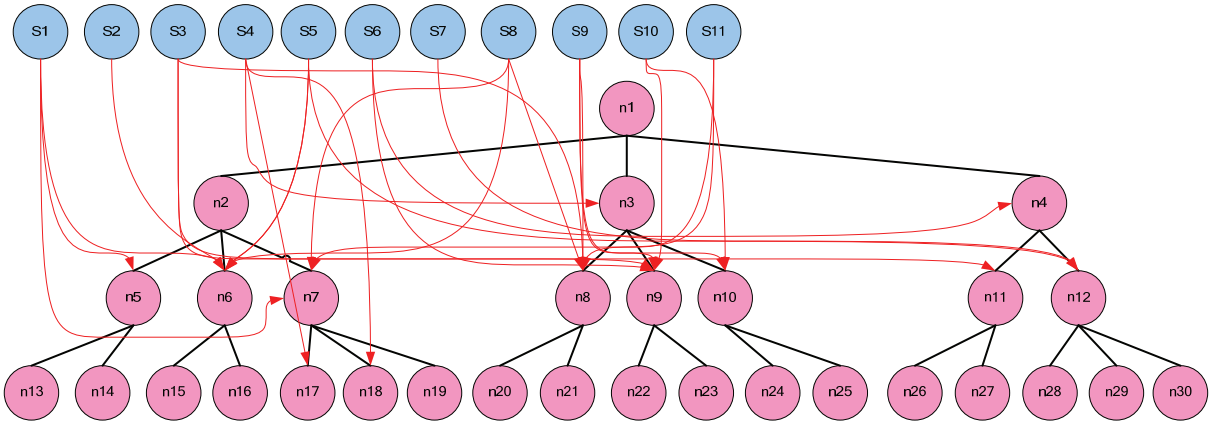

Figure III-2: Simplified model

The details about this model (e.g. Test cost, test confidence) are given in appendix A section 3.1. In order to illustrate the diagnosis algorithm of SIDIS Enterprise, the following scenario will be examined: when symptoms S1, S2, S3 are active and node n16 is supposed to be broken. For this scenario the model is simplified and only the relevant information is depicted on Figure III-3:

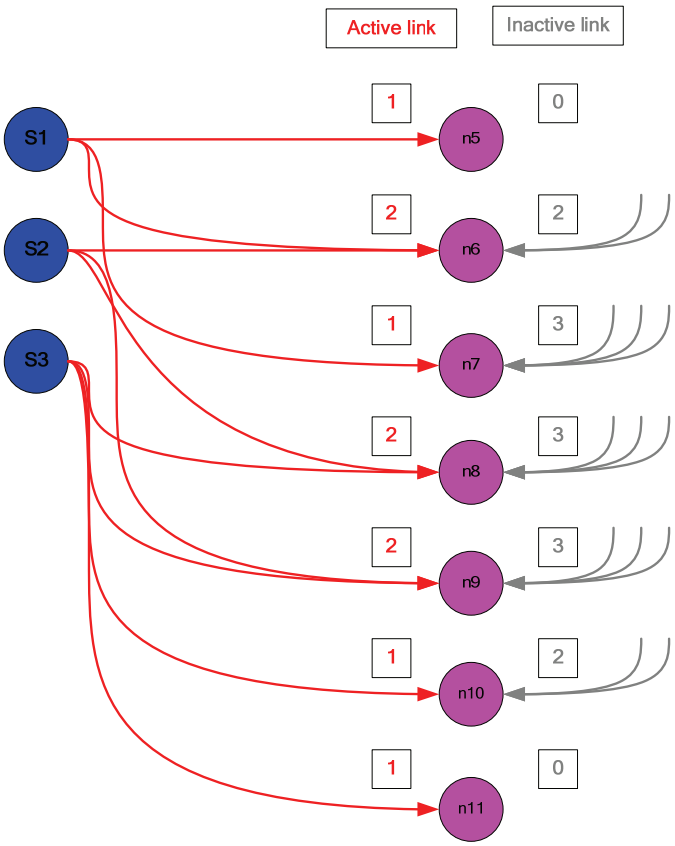

Figure III-3: Relevant data for scenario

After application of (Eq. III-1) the prime test agenda can be computed as detailed in Table III-1:

Prime test agenda				
Node	Suspicion Moment	Test Cost	Test rank	Rank
n5	100%	6	16,66667	6
n6	200%	7	28,57143	4
n7	100%	6	16,66667	7
n8	100%	4	25	5
n9	200%	3	66,66667	2
n10	100%	2	50	3
n11	100%	1	100	1

Table III-1: Prime test agenda for scenario

With the hypothesis that the technician always selects the first object in the test agenda the path of the diagnosis session will be as depicted in Figure III-4:

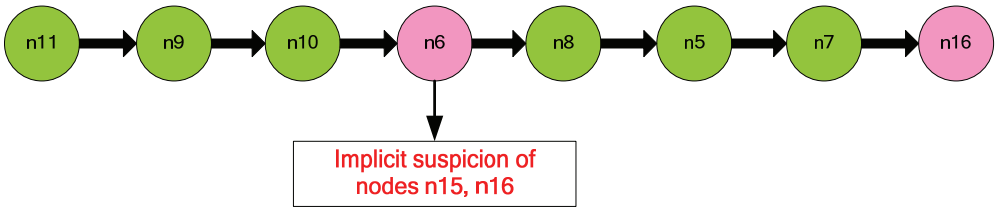

Figure III-4: Path of the guided fault finding procedure

As a conclusion of this use case, the orbit described during this diagnosis session has a cardinal of 8 whereby with adequate weight coefficients the optimal path could be reduced to a cardinal of 2 which would avoid 6 unnecessary tests.
The same use case would be more dramatic if wrong test results occur. The most acid version of this use case would include a 100% test confidence with wrong test answers (eg. OK for node n6), which is a very usual use case because authors do not commit themselves to configure the weight parameters. If the test of node n6 answers 'OK' in this scenario, the orbit will increase from 8 to 14.

The next section will describe the overall performance of SIDIS Enterprise's diagnosis algorithm and its main drawbacks.

1.1.3 Problem statement

For a case base of 15 cases described in appendix A 3.1, a simulation of guided fault finding procedures delivers the results depicted in Figure III-5:

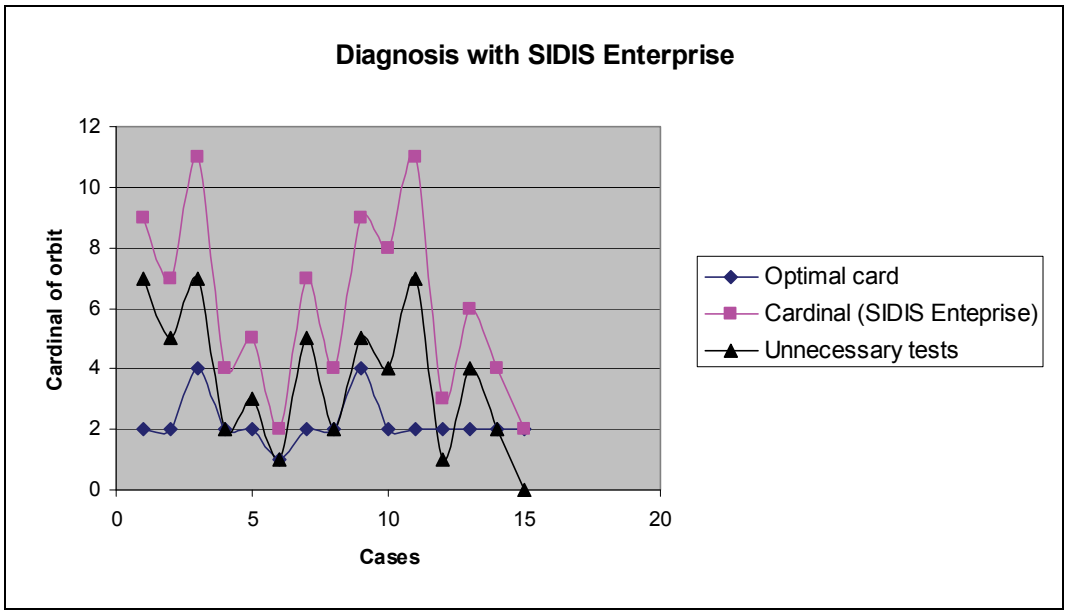

Figure III-5: Guided fault finding with SIDIS Enterprise

For each case, the algorithms always perform unnecessary tests which had a strong impact on the time of each diagnosis session. SIDIS Enterprise's diagnosis algorithm performs in average 3.66 unnecessary tests in each diagnostic session. Compared to the optimal cardinal of each orbit of each session, SIDIS Enterprise is in average 278% higher which impact lower performance and more user effort. These numbers illustrate average results of very usual scenarios due to the fact that authors do not define a weight to the symptoms or a rank to the suspicion links, and under these conditions the broken components which is in 95% of the cases a leaf in the tree is tested once all nodes of a higher level are tested.

Another strong factor that impacts the path of the guided fault finding procedure are the test cost. In this case base the standard deviation of the test cost is 2.36 minutes (for a maximum of 10 and a minimum of 1 minute), and some times the path of the guided fault finding procedure is oriented toward the false sub-tree due to the fact that a normal component has a lower test cost than the broken one.

The four characteristics that directly affect the performance of an algorithm and that needs to be improved for SIDIS Enterprise are listed bellow. They are based on the three arguments given by [Struss, 1997].

- **Compatibility**: As there are presently available lots of model variants and options, it can be considered that every vehicle is unique. Therefore, the algorithm must know which vehicle is diagnosed to avoid mistakes and inappropriate answers.

- **Time**: In a real acid test for a diagnosis engine, the technician should have an answer in a correct time interval.

- **Reliability**: The algorithm should run without errors. Moreover, in an appropriate measure it must consider variables which may affect the result.

- **Relevance**: The object in the first rank of the test agenda should lead to the faulty component.

- **Minimum effort from user**: The diagnostic engine must run in a transparent way; the generation of the candidates must be explained and justified. Furthermore, the graphical user interface must present the results in a comprehensive manner.

- **Maintenance / Costs of models**: the costs of the model are often traded with the precision and reliability of the diagnosis. These costs must not be limited for a commercial product.

1.1.4 Synthesis

With the above mentioned results it is possible to summarize SIDIS Enterprise's diagnosis performance with the criteria from [Struss, 1997]. Concerning the compatibility, the knowledge structures in SIDIS Enterprise makes it possible to avoid redundancies for subsystems that are present on different vehicles, but the number of suspicion links leads some clients to the implementation of specific high priority suspicion links. This is a well known drawback of expert systems already outlined by Price [Price, 1999]: "*the overview of the model is lost*". The second criteria: the time, is strongly dependant on the test, which often needs to call the vehicle communication. It is crucial for a test to cover all possible failure modes of a component in order to deliver correct answers and finally to approach a good ratio of correct answers by time. If the cost of the tests are known (instead of defining the same default value for each test), there is a possibility to optimize the test agenda due to the heuristic metric of SIDIS Enterprise, which depends on the test cost (Eq. III-2). Several diagnosis strategies have been presented in chapter I section 3. Table III-2 summarizes the main advantages and drawbacks of these methods. Based on the features provided by these methods, the next section investigates the possible benefits in SIDIS Enterprise for the strategies based on the expert system and their extension, the consistency based approach and the causal networks.

Name of the method	Advantages	Drawbacks
Expert System	Qualitative reasoning, suited to unstructured problems. Handles incomplete data and knowledge and extendable with other	Integrate hardcoded knowledge. Model overview lost for large systems [Tyler 2007].

	variables [Tyler, 2007].	
Distributed diagnostic agents	Distributed nature and parallel information processing. Modularity (reconfiguration possibilities) [Pavlicek et al., 2007].	Complete System description not known as a whole. Reasoning on separate sub-parts [Alonso et al. 2007].
CBR	Response time very slow [Martin, Plaza, 2004].	Black box principle, Interpretation of results difficult [Martin, Plaza, 2004].
Decision trees	Differential diagnosis, Response time, Adapted objective function [Milde, Hotz, 2001].	Construction of the tree, Discrepencies between prediction and observations [Moore, 2008].
Consistency based approaches	Automatic fault detection and localization [Parka, Suguraman, 2004].	Exhibit low discriminative power. Needs a system description in form of predicates [Parka, Suguraman, 2004].
Causal networks	Suited for diagnostic problems affected by uncertainty. Effective formalism for knowledge representation [Grasso et al., 2007].	Model overview lost for diagnostic problems with high connected sub-parts.

Table III-2: Summary of diagnosis strategies

1.2 Intuitive ideas for SIDIS Enterprise's Diagnosis

1.2.1 Model based diagnosis principle

In the model based diagnostic theory an observed system is composed of the tuplet (*SD*, *COMP*, *OBS*) with:

- *SD*: System Description, a set of logic formula of first order describing the function of the system.
- *COMP*: Component, a finite set of constant representing a set of component $\{c1, c2, \ldots, cn\}$.
- *OBS*: Observation, is a set of atomic formulas representing the observations.

The **detection** of a faulty component is then based on an incompatibility of the sets *SD* and *OBS* and the normal state of all components [Gorny, Ligeza, 2001]. The diagnosis consists in finding the component in an abnormal state in order to re-establish the compatibility. Mathematically formalized these definitions can be expressed as in (Eq. III-5) and (Eq. III-6).

- A diagnosis exists if and only if:

$$SD \cup OBS \text{ proven satisfactory}$$

(Eq. III-5)

- The system is functioning normally if

$$SD, \{\neg Ab(c) / c \in COMP\} \text{ consistent}$$

(Eq. III-6)

With:

- *SD*: System description
- *OBS*: Observation

- *Ab*(): Predicate abnormal
- *COMP*: set of components

For Reiter a diagnosis for a tuplet (*SD*, *COMP*, *OBS*) is defined as a minimal set $\Delta \subseteq COMP$ that:

$$SD \cup OBS \cup \{\neg Ab(c) / c \in COMP \setminus \Delta\} \cup \{Ab(c) / c \in \Delta\} \text{ consistent}$$

(Eq. III-7)

There are 2 different algorithms to find out the R-diagnosis:

- The first one is a naïve one and consists in generating all the subsets Δ of *COMP* by beginning with subsets with a minimal cardinal and test the consistency of $SD \cup OBS \cup \{\neg Ab(c) / c \in COMP \setminus \Delta\}$. This algorithm is very inefficient due to the combinatory explosion of all different subsets to test end generate.
- The second algorithm relies on the definition of conflicts and hitting sets [Zhao, Ouyang, 2007]. A conflict *C* is a set of components $C = \{c_1, c_2, ...c_k\} \subseteq COMP$ that $SD \cup OBS \cup \{\neg Ab(c) / c \in C\}$ is inconsistent. A conflict *C* is defined as minimal if and only if there is no smaller subset of *C* which is in conflict. The notion of hitting set relies on a collection *C* of sets. *H* is a hitting set of *C* if and only if:
  - $H \subseteq \bigcup_{S \in C} S$
  - $\forall S \in C, H \cap S \neq \varnothing$

A subset $\Delta \subseteq COMP$ is a diagnosis for (*SD*, *COMP*, *OBS*) if and only if Δ is a hitting set of the collections of the minimal conflict sets. For the diagnosis the algorithm build the collections of all conflicts and deduce the hitting sets of each conflict which are the diagnosis.

The second algorithm can be used in SIDIS Enterprise's diagnosis engine very easily were *SD* are the inference rules between symptoms and components, *COMP* the diagnostic objects and *OBS* the symptoms. To take benefit of the hitting set algorithm [Zhao, Ouyang, 2007], the symptoms will be weighted in functions of their discrimination power in the diagnosis object tree which corresponds to the possible conflict. The weight of each symptom will be proportional to the number of independent linked diagnostic objects (by a suspicion rule).
Figure III-6 represents an example of a diagnostic object tree, one symptom S1 and 3 suspicion links. The set of related diagnostic objects to the symptom S1 is depicted and defined as *E*1. The set of all components is defined as *T*.

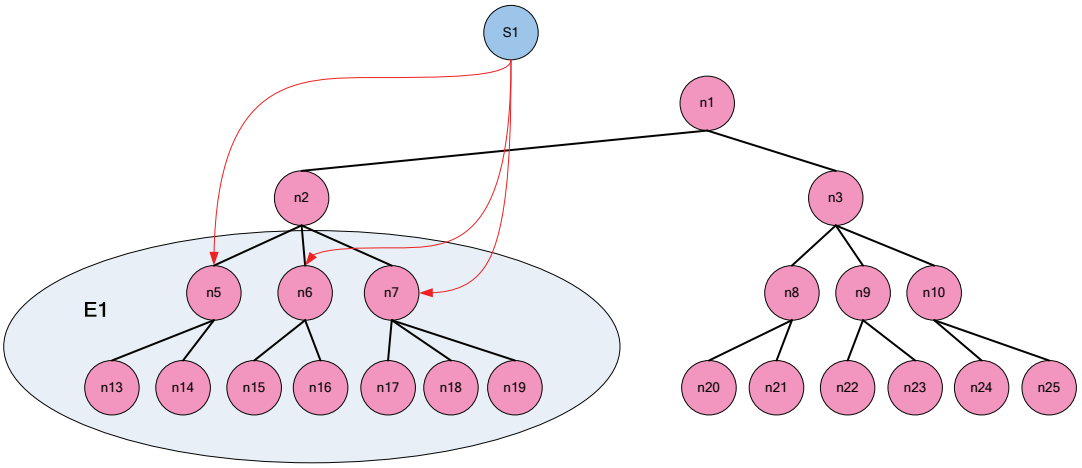

Figure III-6: Symptom weight proportional to hitting sets

Under this condition the weight of the symptom will be given by:

$$W_{S_1} = \frac{card\{E_1\}}{card\{T \setminus E_1\}}$$

(Eq. III-8)

With:

- E_1: the set of suspected nodes including their child nodes
- T: the set of all diagnostic object nodes

In the particular case depicted in Figure III-6 the weight of the symptom S1 is equal to 5/6. It is to notice that the set E_1 is composed by the object that are targeted by a suspicion rule AND their child nodes due to the implicit "is composed by" relationship which is part of the set *SD*.

If the symptoms weight is now computed with this rules and a simulation run through the same case base as in section 1.1.3 a significant performance improvement is obtained, as depicted in Figure III-7:

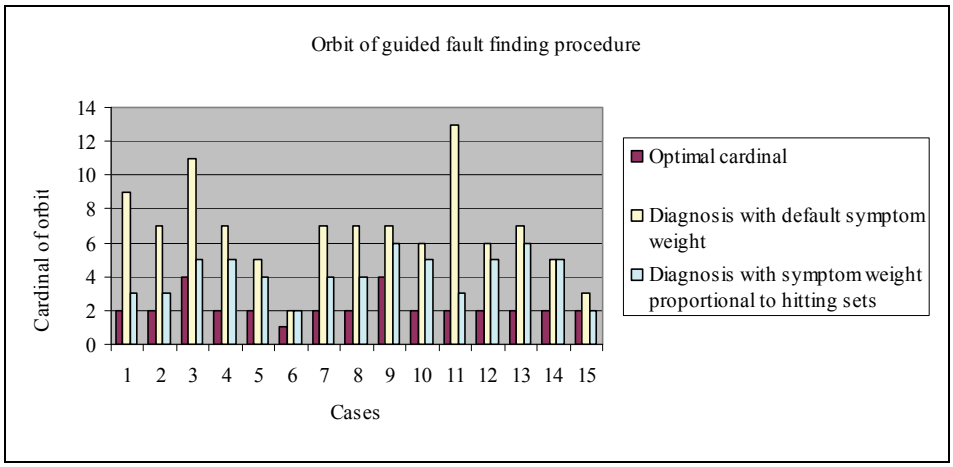

Figure III-7: Cardinal of guided fault finding procedure

This simple symptom weight formula permits to decrease the number of tests by an average value of 2.67 tests, which corresponds to a decrease of 40% for the average cardinal of the orbit of the guided fault finding procedure and a gain of 35% time in average.

This simple principle of the model based diagnostic theory proves immediately its advantages in terms of performance. Moreover, the implementation effort in SIDIS Enterprise is very low. During the diagnosis session, once the data have been filtered and the relevant symptoms activated an iterative procedure can be initiated during runtime to count the suspected nodes for each activated symptom and its child nodes. Then the normal diagnostic engine of SIDIS Enterprise can be run without any changes.

This procedure has the strong advantage to accord a higher importance for symptoms having a higher discrimination power. Moreover, it is very practical for new vehicles when all symptom weights are always set to their default value, the simple loop on all activated symptoms permits to affect the weights in a manner that optimize the test agenda and the overall guided fault finding procedure.

1.2.2 Test Agenda ranking

One other interesting aspect of SIDIS Enterprise is the ordering of the tests in the test agenda. As seen before in (Eq. III-2) the ordering is done by ratio suspicion moments by test cost. The implementation of this heuristic is done in order to approximate an optimal ordering in each circumstance, but there are many ways to combine both parameter suspicion moment and test cost in order to approximate a multi-criteria optimization.

Table III-3 summarizes the results obtained for different heuristic metrics, which were implemented in the simulator and tested on the same case base as before. The results are given in form of a cardinal of the orbit for the guided fault procedure with symptom weights set to their default value and compared to the usual ordering metric. The best results are always given specially when a parameter appears in the corresponding metric (a variation study of the different parameters of each metric is given in appendix A section 3.3) .

Metric Name	Literal expression	Results	Comparison to the orbit of SIDIS Enterprise
Balanced suspicion and test cost	$\alpha SM + (1-\alpha)\frac{Tc_{\max}}{Tc}$ **(Eq. III-9)**	6.6 for $\alpha = 0.6$	+7%
Balanced and normalized suspicion and test costs	$\alpha SM + (1-\alpha)\frac{Tc_{\max}}{Tc}100\%$ **(Eq. III-10)**	6.6 for $\alpha = 0.9999$	+7%
Soft constraints balancing between test costs and suspicion	$\alpha^{\beta} SM + (1-\alpha)^{\beta}\frac{Tc_{\max}}{Tc}$ **(Eq. III-11)**	11.9 for $\alpha = 0.5$ and $\beta = 0.1$	+95%
Soft constraint between suspicion and test costs with first depth inspection	$\alpha SM + (1-\alpha)\frac{Tc_{\max}}{2}\left(\frac{1}{Tc} + \frac{1}{\overline{Tc_{child}}}\right)$ **(Eq. III-12)**	5.8 for $\alpha = 0.5$	-6%

Table III-3: Simulation with different heuristic metrics

For (Eq. III-9) to (Eq. III-12) the notation is defined in the list below:

- SM: the suspicion moment of the considered diagnostic object
- $Tc_{\max}$: the maximal test costs in the whole diagnostic object tree
- Tc: the test cost of the considered diagnostic object
- Tc_{child}: the test cost of the child node of a considered diagnostic object
- α or β: two parameters between zero and one
- $\overline{X}$: the average value of a set of values X

One of the most advantageous results obtained here in terms of performance appears to be the metric, which balances the suspicion moment and an average moving test cost. What is more interesting is that α is set to 50% for this result, which means that suspicion moment and test cost had an equally importance in the ordering of the test agenda. With this metric the test are ordered depending on the test cost and with a first depth inspection the average test cost of the child nodes. It is also interesting to notice that for small values of α all the different metrics from (Eq. III-9) to (Eq. III-12) prove a dramatic increase of the orbit, which means that for the approximation of a time-optimized test agenda it is necessary to accord a higher importance to the suspicion moment than to the test cost (if the metric uses a linearization of both parameters). The last formula (Eq. III-12), which allows to obtain improved performance makes a first depth inspection to show better results due to the fact that the suspicion moment is correctly weighted in the formula and that the next test agendas are anticipated orienting the diagnosis on a branch of the diagnosis object tree, where tests cost are lower.

This section reported intuitive ideas of changing the heuristic metric of the test ordering of SIDIS Enterprise. The advantage of these metrics is that they can be directly implemented in SIDIS Enterprise without changing the information model. Thus, the gain in terms of performance is very significant, in particular, if the 6% (cf. last row in Table III-3) benefits are considered for the whole after-sale network of the car manufacturer.

1.2.3 Information gain

One last intuitive idea to increase the current situation of the diagnosis consists in the optimization of the information at the beginning of the fault finding procedure after the calculation of the prime test agenda. The previous results simulated on a prototype show that the prime test agenda is the crucial step of the whole diagnosis session. Thus, it needs a careful discrimination of the candidates in order to direct the diagnosis to the broken components. To achieve this objective, a method called "symptom questioning" is implemented and run after the prime test agenda. The methods examine if the child objects of the diagnostic nodes in the test agenda are linked to perceived symptoms through suspicion rules. If yes, then the diagnosis engines ask the user if this symptom is observed. If the symptom is observed, then the prime agenda is recomputed, but in order to avoid too much user effort (in question answering) a limit has been set to a maximum of 3 questions. The configuration of the triggering of a symptom questioning is depicted in Figure III-8.

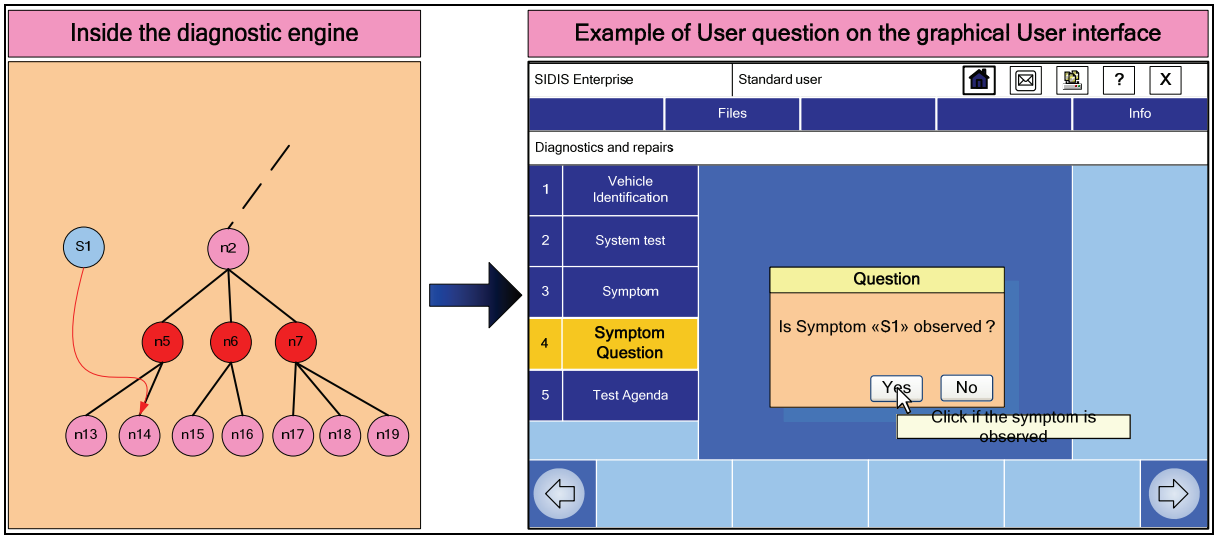

Figure III-8: Perceived symptom questioning

With the implementation of this methods and the simulation of the diagnosis through the previous case base, the gain in terms of performance is minimal (less than 1%), because only one case has the particular configuration where the symptom question occurs. This is due to the fact that the perceived symptoms are rarely used for most of the car manufacturers, but the trend is that perceived symptoms will have more and more importance in the future and manufacturers spend more effort to complete the perceived symptom database.

1.2.4 Outlook

This section has presented three different features that achieve the final objective of increasing the performance of the guided fault finding procedures. It is important to notice that these features can be implemented without changing the knowledge networks or the models for the diagnosis. The first feature (cf. section 1.2.1) is issued from the theoretical research of the field of model based diagnosis and the hitting sets. This simple fact of affecting the symptom weight in dependence on the discrimination power of their respective suspicion links allows significant saving of time for a complete guided fault finding procedure. Moreover, this method is particularly well suited for new vehicles when no return of experience is available and when all the weights of the symptoms are set to their default value, which is often the case, because authors did not commit themselves to define weight coefficients. The second feature (cf. section 1.2.2) consists in changing the metric for the ordering of the test agenda. With a balanced suspicion moment and moving average test cost through a first depth inspection the time-optimal agenda can be better approximated than the current one. At

least the last feature (cf. section 1.2.3) allows to improve the information at the beginning of the diagnosis session, speeding up the fault localization step of the diagnosis process.
The three features of this section can be implemented with minimal changes inside SIDIS Enterprise knowledge networks. When combined, they allow to obtain a gain of 45% for the orbit of the guided fault finding procedure, which means in average 2.22 less tests to perform for each diagnosis session. In practice the users economize two selections in the test agenda. Despite this performance gain the combined performance gain of all three features is still 157% higher than the optimal diagnosis path! The next sections continue to examine different strategies to optimize the guided fault finding procedure.

1.3 Synthesis

The current algorithm of SIDIS's Enterprise guided fault finding procedure performs around 278% (based on the previous case base) more tests than the optimal diagnosis path. This low performance can easily be explained by the fact that most of the parameters of the objects in the diagnosis related knowledge structures are not sets to relevant values by authors. In order to minimize the maintenance costs for the diagnosis of models, car manufacturers do not spend the necessary time to set relevant values to these parameters with the consequence of a lack of performance. The other reason relies on the complexity of the vehicles. For example in a modern vehicle, if a problem occurs with the motor rpm sensor usually read from the motor management ECU, there are around 50-60 default codes which appear in other related subsystems as for example the ABS. These failure codes initiate between 90 to 170 suspicions rules (depending on the vehicle) with the consequences that at least 100 diagnostic objects are candidates in the first test agenda. Therefore, the fault localization process is extremely long and needs to discriminate a lot of candidates through tests. Despite the three features proposed in section 1.2, the current diagnosis engine of SIDIS Enterprise does not fit with the complexity of modern and future vehicles. Moreover, car manufacturers use more and more multiplexing technologies to reduce the length of the cable and transmitting several signals on the same cable, which limits the measurability and observability of symptoms. As outlined a crucial drawback of SIDIS Enterprise is that it does not support the automatic integration of the return of experiences. In the current version of the product there is only an option of sending a protocol of the diagnosis session to a server but no algorithm of knowledge discovery or automatic learning is included. And as seen before an adaptation of the weights of the symptoms could strongly increase the performance of the diagnosis (cf. 1.2.1).

Managing such an interconnected knowledge and in particular the consequences of one failure of all connected subsystems without changing the knowledge networks of SIDIS Enterprise is nearly impossible. To realize an efficient fault localization and identification in the increasing number of possible failures and their repercussions, the diagnosis engine needs a continuous improvement to fit with the requirements of modern and future vehicles. It has to be remembered that SIDIS Enterprise is designed for the whole after sale network of a car manufacturer and that for example a gain of only 1% performance for the guided fault finding procedure of SIDIS Enterprise can save 31 thousands minutes working time for a country like Germany. With an extrapolation this saved time can reach 131.000 hours a year, an extrapolation of the power man/month could not been made due to the fact that data about car manufacturers are confidential. But it is important to keep in mind that the worldwide scale of a manufacturer's after sales network makes each little gain of performance extremely significant. To achieve the final objective of this thesis, two different approaches are examined deeper in the next chapter. The first one relies on the implementation of meta-heuristics. The second is issued from the model based diagnosis theory and consists of the implementation of causal networks which are already examined in chapter I section 3.3. The last part of this chapter is concerned with the objective to minimize the post-development cost for the model for the diagnosis and therefore examines how knowledge discovery and automatic learning can be integrated in order to obtain an efficient framework for the diagnosis of modern and future vehicles.

2 Proposal of new diagnosis strategies

2.1 Industrial requirements

In this section, the crucial industrial requirements are summarized in the subsequent list of 3 criteria [Struss, 1997]:

- **Compatibility**: This is the most important point. Indeed, working on an improvement that cannot be run is quite useless. Therefore, the properties of the authoring and workshop system must be kept in mind so that data of the objects can be captured and then implemented and then still remaining accessible for the workshop software.
- **Usability**: Having a new system which is compatible with the older one but which is not used, because the investment to edit the models is too important would have no commercial success. The trend for manufacturers is clearly to minimize theses costs, therefore for each new information needed the balance between cost of the information and gain in terms of performance must be quantified. The usability concerns also the graphical user interface of the workshop system. It is crucial that the delivered result of the diagnosis engine is well presented and interpretable. A study of the ADAC had proven that in 30% of the cases a computer aided diagnosis system, which gives wrong, non interpretable answers or takes too much time is not used again by the technician in the workshops.
- **Data Base**: The evolution will irremediably have consequences on the database. But these should remain acceptable. In practice, it means that its size should no more than double, what would prevent a long searching time and limit the needed storage space. For a vehicle model a new parameter increases the size of the database by approximately 2kb (for a variable of type float).

These properties will serve as a scale to evaluate and compare between different solutions. It is important to notice that all criteria are interconnected: a change in the information model implies changes in the database and more time during the diagnosis session alone through the repeated access time to the database. Therefore, the evaluation basis is not completely independent. But the most important quantified performance indicators of a solution are the ratio of correct answers and specifically for a commercial product the ratio of correct answers divided by the cost for the models.

2.2 A meta-heuristic strategy

As described previously in chapter I section 3.3.1, the approach of decision trees shows several advantages for the diagnosis in accordance with the above mentioned industrial requirements as:

- They are simple to understand and interpret.
- They have values even with little hard data.
- They use a “white box” model.
- They can be combined with other decision techniques.
- They are computationally cheap.
- There are alternatives to information gain for splitting nodes (e.g. the efficiency of the tests).

Moreover, the guided fault finding procedure of SIDIS Enterprise relies in the optimization of successive test agendas which can be seen as discrete function or a decision tree. In chapter I, section 3.2.3 reports about different types of multi-criteria pseudo optimization through heuristics metrics and as seen in chapter III section 1.2.2, the ordering of the tests with a different metric had proven to possibly increase the performance. This section will investigate deeper the evaluation of the suspicion moment. Section 1.1 which provides an analysis of the current situation, reports about the parameters used by the diagnosis engine of SIDIS Enterprise, which are reminded in the subsequent list:

- **Symptom weight**: One of the most important parameter for the propagation of the suspicion because the suspicion of the diagnostic objects is proportional to the weight of active symptoms.
- **Rank of suspicion rules**: This parameter is quasi always set to the default value. But if for a symptom different rankings have been set to its related suspicion rules, then the propagation of the suspicion is regressive from the rules ranked first to the last ones.
- **Diagnostic object weight**: The implicit 'notOK' declaration of a test propagates a suspicion to the child nodes of the concerned object, which is proportional to the weight of the diagnostic object.
- **Test confidence**: If a test answers 'OK' but if the test confidence is not equal to 100% then the suspicion moment of the corresponding object is set to the value of the test confidence.
- **Test cost**: finally in order to optimize the ordering of the test agenda, the test cost is taken into account for the ranking.

But there are many other parameters that should be taken into account for the computation of the suspicion moment:

- **Precision**: The more precise a rule is, the higher should be the suspicion moment in order to enhance a precise diagnosis.
- **Lifetime**: Each component has a lifetime (often represented as the bath thumb curve). Moreover, the vehicle lifetime in km is always known at the beginning of diagnosis session, because it is read during the global system test step.
- **Causal relationship**: As reported in chapter I section 3.3, causal relationships may help to speed up the fault localization process.

Because the system should emphasize the elements which are the most suspected, resp. whose causes play an important role, the idea is to use the information theory. Another reason is that the information entropy is an important indicator in statistical analysis (and therefore for a feedback engine). The information entropy given in (Eq. III-13) is based on the Shannon entropy [Shannon, 1948], detailed in Shannon's work:

$$H(X) = -\sum_{i=1}^{n} p(x_i).\log_b\ p(x_i)$$

(Eq. III-13)

Where:

- X: is a finite discrete probability distribution with n output
- n: the number of outputs of X
- $p(x_i)$: more the probability of output x_i
- b: the base of the logarithm which is equal to 2.

One important property of the information entropy, which will be used later is that it is additive. The next subsections report on the implementation of the information entropy in the diagnosis engine with parameters different at each time.

2.2.1 Parameters for a meta-heuristic metric

2.2.1.1 Basic suspicion through expert rules

As reported in section 2.2 the meta-heuristic strategy does not change the information model, but it is difficult to apply the entropy concept for the suspicion of rules. The probability depends on the weights of the symptoms of an expert rule and if the symptom is initiated. Thus, the formula for the suspicion is given by:

$$PM_{node} = \log_2\left(o + \frac{\sum_i e_i GS_i}{\sum_k GS_k}\right)$$

(Eq. III-14)

With:

- PM_{node}: the partial suspicion moment of a diagnostic object node.
- e_i: equal to 1 if the symptom indexed by i is active, and 0 else.
- GS_i: is the weight of the symptom indexed by i.
- o: an offset equal to 1.

The partial suspicion moment is then logarithmic dependant on the ratio between the active symptoms and the total number of symptoms which have links pointing on the considered diagnostic object.

- Example: In Figure III-9 a diagnostic object tree is depicted with 3 symptoms S1, S2, S3 (with the weights GS_1, GS_2, GS_3), and 6 suspicion links. The symptoms S1 and S2 (painted in dark in Figure III-9) are activated.

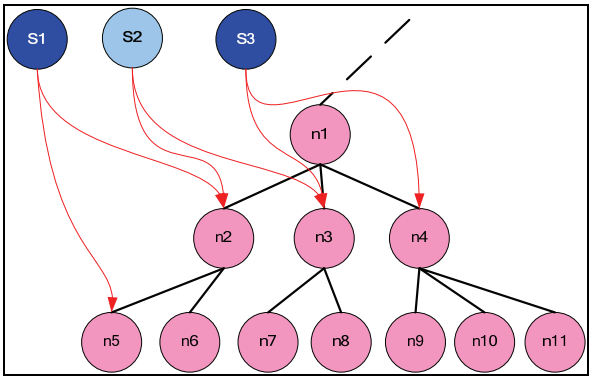

Figure III-9 : Perceived symptom questioning

The partial suspicion moment of the nodes n2, n3, n4, n5 is then given by:

$$\begin{cases} PM_{n2} = \log\left(o + \dfrac{GS_1}{GS_1 + GS_2}\right) \\ PM_{n3} = \log\left(o + \dfrac{GS_3}{GS_2 + GS_3}\right) \\ PM_{n4} = \log\left(o + \dfrac{GS_3}{GS_3}\right) \\ PM_{n5} = \log\left(o + \dfrac{GS_1}{GS_1}\right) \end{cases}$$

(Eq. III-15)

With this formalization the partial suspicious moment of a symptom increases with a logarithmic inclination from 0 to 1. The more active symptoms are connected to a diagnostic object the higher the suspicion of the object. This behavior changes a lot the distribution of the suspicion in SIDIS Enterprise. With the same example and with a default value of 100 for the weight of the symptoms the partial suspicion moment of the nodes is given in Table III-4:

Partial suspicion moment		
Node	With SIDIS Enterprise's	With the entropy

	algorithm	derived formula
n2	50 %	58 %
n3	50 %	58 %
n4	50 %	100 %
n5	50 %	100 %

Table III-4: Comparison of the partial suspicion moments

The distribution of the entropy derived formula has clearly a higher discrimination power and moreover the suspicion increases with a higher number of active linked symptoms. The irradiation of the suspicion is no more proportional to the symptom weight but logarithmic ascending, causing a higher dispersion. Applied to the previous case base this metric permits to reduce the orbit of the diagnosis session by 5.3% (with all symptoms weights set to their default values). Therefore, the idea of using the entropy for the candidate generation makes again a little performance gain.

2.2.1.2 Inclusion of the precision

As previously mentioned the inclusion of the precision in the candidate generation should also be taken into account due to the fact that the more precise a rule is the higher the information gain is. Adapting again the entropy formula to the precision of an object delivers the formula given by:

$$PM_{node} = \log_2\left(o + \frac{Level(node)}{Depth(tree)}\right)$$

(Eq. III-16)

With:

- PM_{node}: the partial suspicion moment of a diagnostic object node
- *Level(node):* the depth in the diagnostic object tree of the considered node
- *Depth(tree):* the depth of the diagnostic object tree
- *o:* an offset equal to 1

The behavior of the partial suspicion moment is then quite similar to the one detailed for (Eq. III-15) due to the fact that the formulas are quite the same except that this time the depth of a diagnostic node is considered. Applied to the example described in section 2.2.1.1, the partial moments derived from the precision are given in Table III-5:

Partial suspicion moment induced by the precision		
Node	With SIDIS Enterprise's algorithm	With the entropy derived formula
n2	Not taken into account	73 %
n3	Not taken into account	73 %
n4	Not taken into account	73 %
n5	Not taken into account	100 %

Table III-5: Comparison of the partial suspicion moments

The deeper the node is in the tree, the higher its suspicion in order to privilege precise results in the guided fault finding process. If both formulas (Eq. III-15) and (Eq. III-16) were used to perform a diagnosis a gain of 10% of the overall performance of the guided fault finding procedure is obtained. But the balance between both factors precision and suspicion links changes the overall results. In this series of experiences (reported in appendix A section 3.4.2) the balance was set as given by:

$$PM_{node} = \log_2\left(o + \frac{\sum_i e_i GS_i}{\sum_k GS_k}\right)100\% + \log_2\left(o + \frac{Level(node)}{Depth(tree)}\right)10\%$$

(Eq. III-17)

A similar balance of both factors leads to an increase of the performance by 30%. This result proves again that the suspicion induced by the rules have a much higher importance than other factors. This conclusion is already reported in section 1.2.2 where the suspicion induced by the rules is put in balance with the test cost for the ranking in the test agendas. In conclusion the suspicion links in the diagnosis compared to the other parameters are the most important factor. The concept of precision as used in this section considers the model of the car (the diagnostic object tree) as an ontology; in information retrieval based on ontologies the similarity formulas often include the precision factor as defined in (Eq. III-16). In the diagnosis, this factor permits to take into account the distance in the fault localization procedures.

2.2.1.3 Integration of the test costs

The test cost factor is already taken into account for the calculation of the ranking in the test agenda. But in this section the test cost will be included into the meta-heuristic, as if they participate to the computation of the suspicious moment. Once again, the calculation is quite identical to the precision detailed in section 2.2.1.2. The formula of the partial suspicion moment induced by the test cost is given by:

$$PM_{node} = \log_2\left(o + \frac{Tc_{node}}{Tc_{max}}\right)$$

(Eq. III-18)

With:

- PM_{node}: the partial suspicion moment of a diagnostic object node.
- Tc_{node}: the cost of the test of the associated diagnostic object node.
- Tc_{max}: the maximal test cost over the whole set of tests in the tree.
- o: an offset equal to 1.

As for the precision, the behavior of this formula is quite similar except for the balance with the other factor. In order to privilege candidates with lower test costs a negative multiplicative factor must be placed before the expression, because the maximal value is reached for the node with the highest test cost. Thus, the behavior must be decreasing when the test costs increase. The results obtained through a simulation on the case base shows that the orbit of the guided fault finding procedure decreases when a factor in between -10% and -60% is set with around 1.6 less test to perform per diagnosis session. From -70% to -100% the orbit increases again and in some cases a diagnosis session is performed with more than 15 cases, which is definitively too long.

2.2.1.4 Using the life time of components

One important parameter for the diagnosis is of course the life time of the different components of a system. It gives precious information to the technician for the generation of suspected candidates. Thus, it is normal to integrate this parameter in the combination of a meta-heuristic. But the life time is a new parameter for SIDIS Enterprise and it needs to be entered by the authors in the model, which on the first sight seems to be in contradiction with the objective to minimize the post development cost of models for the diagnosis. But chapter III section 3.1.2 is concerned with a strategy to learn automatically the life time of components; this explains the fact that despite the precious information the life time parameter can reveal for the diagnosis it makes sense to integrate it in the heuristic for the candidate generation. As before the life-time is captured in a similar formula as in (Eq. III-18). The corresponding factor is given by:

$$PM_{node} = \log_2\left(o + \frac{LT(vehicle)}{LT(node)}\right)$$

(Eq. III-19)

With:

- PM_{node}: the partial suspicion moment of a diagnostic object node.
- *LT(vehicle):* the km reading of the vehicle.
- *LT(node):* the life time in *km* of the corresponding node.
- *o:* an offset equal to 1.

The differences with the other factors described in sections 2.2.1.1, 2.2.1.2 and 2.2.1.3 is that this partial moment has no superior limit. It increases logarithmically with the kilometer reading of the vehicle. If the kilometer reading is higher than the life time of a component the factor is higher than 1 and will contribute to increase the total suspicion moment of the nodes and privilege its ranking in higher positions in the test agenda. On the other hand, if the kilometer reading is lower than the component's life time, the factor is lower than 1 and contributes to a lower total suspicion moment which makes sense with the intuitive idea of diagnosis.

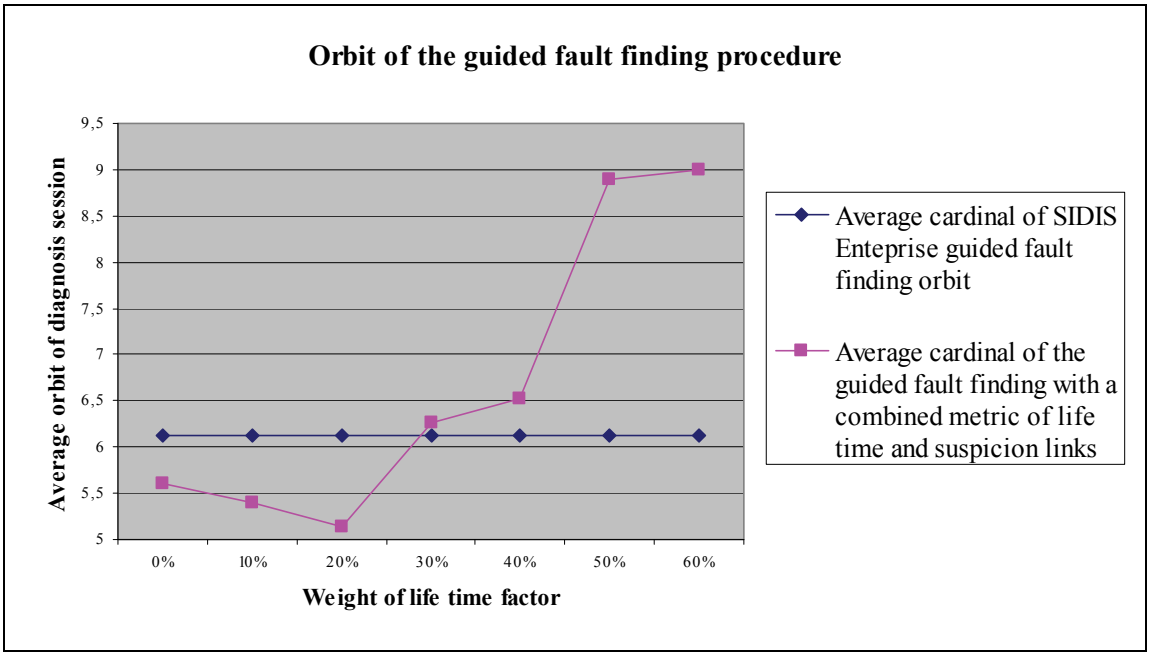

Figure III-10 : Influence of life time factor

In Figure III-10 the average cardinal of the orbit of diagnosis session is depicted for an increasing importance of the life time factor from 0% to 60%. At the beginning the curve is descending, which confirms the intuitive idea that the life time of components allows to improve the candidate generation and the overall diagnosis, but once again if the balance between suspicion rules and life time factor is set in a manner that privileges this last one, then the orbit increases over the average values of SIDIS Enterprise's standard algorithm. The importance of the hardcoded knowledge in the expert rules still remains the most crucial factor for a good diagnosis.

2.2.1.5 Using the dependencies between components

Chapter I section 3.3 reports about a subfield of model based diagnosis methods relying on causal dependencies. Causal dependencies between components can be identified and validated with the help of a feedback engine. Chapter III section 2.3.1 investigates how to automatically learn causal

relationships, which makes it possible to implement them in models without increasing the maintenance costs. In this section, the model is supposed to contain causal relationships between diagnostic nodes which can be weighted. In practice these relationship can for example model an electric connection or mechanical connection, which is a strong advantage for the creation of models of mechatronics systems thanks to the abstraction level of causal relations. This time the partial moment derived from causal relationships is quite identical to the formula given in section 2.2.1.1 for the suspicion link. The formula is given by:

$$PM_{node} = \log_2\left(o + \frac{\sum_i e_i GC_i}{\sum_k GC_k}\right)$$

(Eq. III-20)

With:

- PM_{node}: the partial suspicion moment of a diagnostic object node.
- GC_i: the weight of the causal link indexed by i.
- e_i: a boolean value equal to 1 if the causal successors of the node are already suspected and 0 else.
- o: an offset equal to 1.

Formula (Eq. III-20) allows to privilege the possible causes of failures. The first step consists in the evaluation of the suspicion rules. Then all possible candidates with a partial suspicion moment superior to zero are deeper examined. If they have outgoing causal relationships pointing to suspected objects then the factor in (Eq. III-20) will contribute to increase the global suspicion of that object. Figure III-11 shows an example of a diagnostic object tree with implemented causal relationship. The nodes in darker colour (nodes: n3, n5 and n6) are already considered as suspected objects by the ignition of the symptoms and their respective suspicion rules. In this example all causal relationships had a default weight value of 100%.

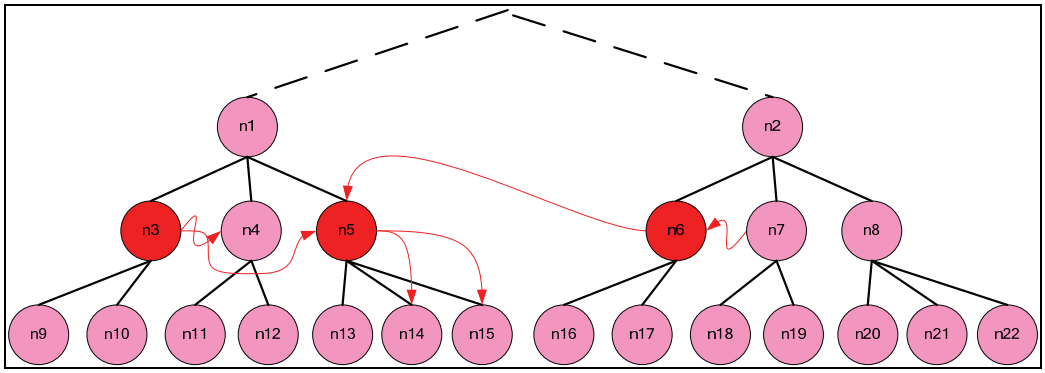

Figure III-11: Example of causal network

The calculation of the partial moment due to the causal relationships will concern the nodes: n3 and n7. Node n3 is the causal ancestor of nodes n4 and n5 whereby only the node n5 is already suspected. The ratio sum of weights of active causal successors divided by the sum of the weights of all causal successors is equal to 0.5 (one active causal link for a total of two causal links with the same weights). The partial causal moment of node n3 is then equal to log(1+0.5) which is approximately equal to 58%. Node n7 is the causal ancestor of node n6 which is suspected. Node n7 had only one outgoing causal relationship which means that its partial causal moment is equal to log(2) which is exactly equal to 100% (the logarithm is computed in base 2). The effect of this factor is to re-enforce the suspicion of causal ancestors of components taking fully advantage of the abstraction level of causal relationships in the model for the diagnosis. Moreover, if a cycle appears in the causal graph between suspected nodes, the global suspicion moments of these nodes will increase logarithmically, which causes no particular discrimination between the two concerned nodes. The causal cycle situation may occur often in real case example as reported in the simulation of real cases (chapter IV 3.3) this

situation is mainly due to the fact that the model with causal relationships covers a large number of failure modes, whereby each breakdown implies a different topology of the causality graph.

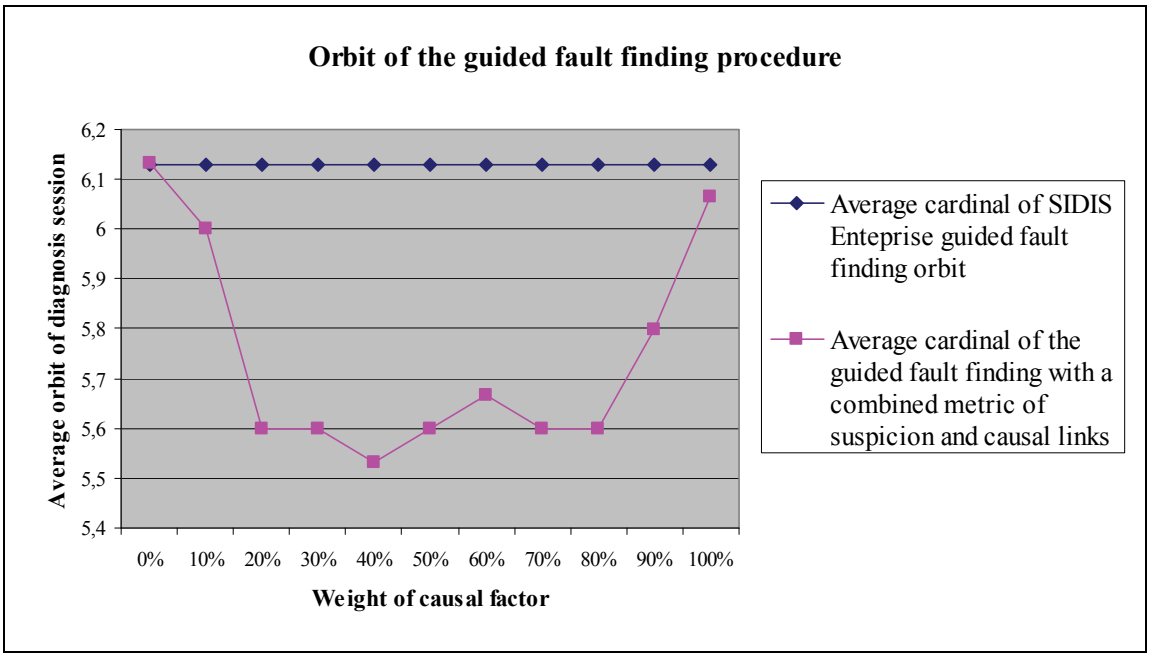

Figure III-12: Influence of causal factor

The impact of the causal factor on the orbit of the guided fault finding procedure is depicted in the chart in Figure III-12. On both extremities of the curve when the weight of the causal factor is too low or too high the orbit tends to be identical in average to the current algorithm of SIDIS Enterprise. But for a weight between 20% and 80% a gain of approximately 8% for the orbit of a guided fault finding procedures is revealed. This factor is one with the greatest impact on the performance of the guided diagnostic mode with a meta-heuristic.

2.2.1.6 Theoretical behavior of the different factors

For more convenience the previous described factors of section 2.2.1.1 to 2.2.1.5 are denoted r_1, r_2, r_3, r_4, r_5 as remembered in (Eq. III-21) and the polynomial combination of the factors is given in (Eq. III-22) with the weight coefficients a_0 to a_4.

$$r_1 = \log_2\left(o + \frac{Level(node)}{Depth(tree)}\right)$$

$$r_2 = \log_2\left(o + \frac{Tc(node)}{Tc_{\max}}\right)$$

$$r_3 = \log_2\left(o + \frac{LT(vehicle)}{LT(node)}\right)$$

$$r_4 = \log_2\left(o + \frac{\sum_i e_i GS_i}{\sum_k GS_k}\right)$$

$$r_5 = \log_2\left(o + \frac{\sum_i e_i GC_i}{\sum_k GC_k}\right)$$

(Eq. III-21)

$$SM_{node} = a_0 r_1 + a_1 r_2 + a_2 r_3 + a_3 r_4 + a_4 r_5$$

(Eq. III-22)

The factors r_1, r_2 and r_3 have a quite similar behavior which is an increasing logarithmic curve depending on the value of the ratio inside the logarithm. This means that the higher the level of a considered node in the tree, the nearer to one is the ratio *Level*(*node*) / *Depth*(*tree*). This implies that the factor r_1 is nearer to one, its maximal value. The same apply for the factors r_2 and r_3 except that r_3 has no upper bound. Indeed, the km reading of the vehicle can be much higher than the pre-defined life-time in km of a component, but that effect is intended in order to increase the suspicion in that case. The behaviour of factor r_4 depends only on the active suspicion links and on their weights, the curves are plotted in Figure III-13 for two nodes with 3 and 6 incoming suspicion links (whereby all related symptoms of the suspicion links have a default weight value of 100%). The global attitude is a simple logarithmic increase of the curves from 0 to 1 depending on the number of active suspicion links. But compared to the current algorithm, it is a different to the linear increasing. The suspicion radiation of active symptoms is re-enforced due to the logarithmic behaviour, a node with only 50% of active links will have suspicion moment of over 80% compared to 50% with the algorithm of SIDIS Enterprise. The cutting of the tree in suspected regions is enhanced due to the higher discrimination power achieved through this factor. Factor r_5 has exactly the same behaviour as r_4, except that the number of active suspicion rules must be replaced by the number of active causal links.

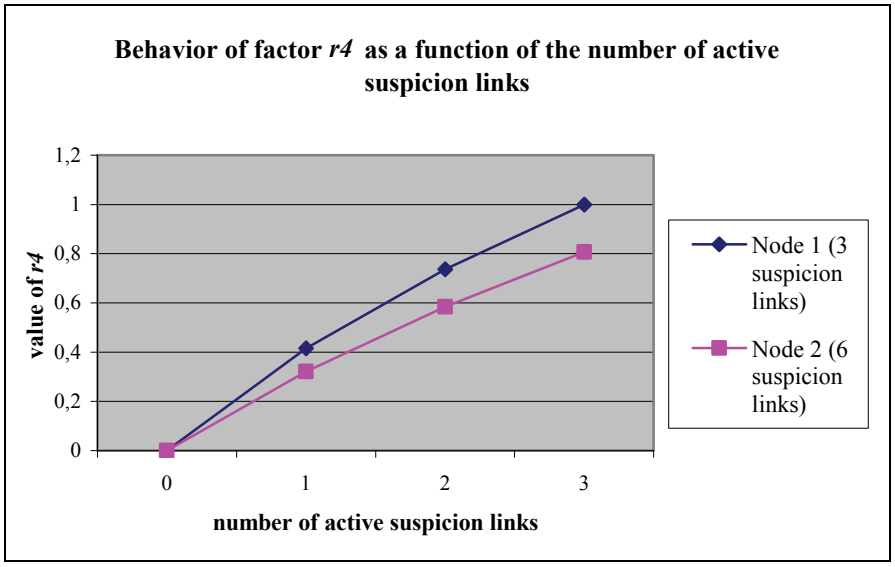

Figure III-13: Evolution of the factor r_4 in function of the active suspicion links

Finally, all described factors are related to the diagnosis but the main question is how to combine them in an efficient way to improve the performance of SIDIS Enterprise. The next section is concerned with the search of optimal function points of a polynomial combination of the different factors.

2.2.2 Combination of the parameters for a meta-heuristic metric

2.2.2.1 Examination of function points of the metric

Table III-6 summarizes the results obtained with the combination of all factors and with variation of the test cost factor (on the second row), of the lifetime (on the third row) and of the partial causal moment (on the fourth row). The results given are the average orbit of a guided fault finding procedure over 15 cases simulated for the same model as the previous sections.

		Variation (negative for coefficient a_1)								
		0%	10%	20%	30%	40%	50%	60%	70%	80%
Average cardinal of guided fault finding orbit	SIDIS Enterprise	6,13	6,13	6,13	6,13	6,13	6,13	6,13	6,13	6,13
	a_0=10%; a_2=10%; a_3=100%; a_4=20%; a_1 increasing	6,14	7,64	5,35	5	5	4,92	5,57		
	a_0=10%; a_1=-10%; a_3=100%; a_4=20%; $a2$ increasing	5,85	6,78	5,5	6,42					
	a_0=10%; a_1=-10%; a_2=10%; a_3=100%; a_4 increasing	5,64	5,57	5,57	5,57	5,57	5,42	5,5	5,5	5,5

Table III-6: Impact of the orbit for variations of the different factors

These results restrict slightly the previous obtained intervals to obtain sense fully function points of the meta-heuristic metric. For example if the test factors interval is comprised between -20% and -50% over this limit the orbit increases dramatically, because the 'discrimination' by the new formed diagnosis engine relies quasi only on the test cost, a situation which should be avoided. The life-time factor should be weighted around 20%, because this factor does not correspond to hardcoded diagnosis knowledge as suspicion rules or causal links. Thus the weight of this factors must remain low. Another disturbing arrangement of the life time is that it can only be computed through the km reading, whereby theoretically the number of working hours of the components should be taken. But this parameter cannot be read from the car inhibiting an exact approach of the component's state. Figure III-14 outlines the promising areas of the meta-heuristic based on the simulation results in appendix A section 3.3. The tolerance margin of the intervals of each coefficient a_0, a_1, a_2, a_3, a_4 is set to $\pm$ 10% for the reason that the case base is not large enough for a deeper investigation. But the model used in the simulations emphasies all real encountered situations in the diagnosis of a modern vehicle. Chapter IV includes a deeper examination of the function points of this meta-heuristic metric based on complete vehicle models.

As a conclusion the optimal space (indexed by zone A in Figure III-14) for the meta-heuristic is composed by the interval (10%, [20%, 50%], 20%, 100%, 50%) for the coefficients (a_0, a_1, a_2, a_3, a_4). This metric allows a peak of a performance gain of 9% in terms of cardinal of the orbit of the guided fault finding procedure. Combined with an adequate symptom weight computation based on discrimination power (or hitting set, cf. 1.2.1) the performance gain reaches 9.4%. If this number is scaled to the size of the after sale network of a car manufacturer in a country like Germany the potential time savings can reach 49 thousand working hours a day or 1.2 million hours a year with quick approximation. But this result must be put in balance with the manpower to create and edit the new needed data by the meta-heuristic. Indeed, two new parameters appear in this formula requiring new knowledge: on one hand the causal relationship and on the other hand the life time of the components.

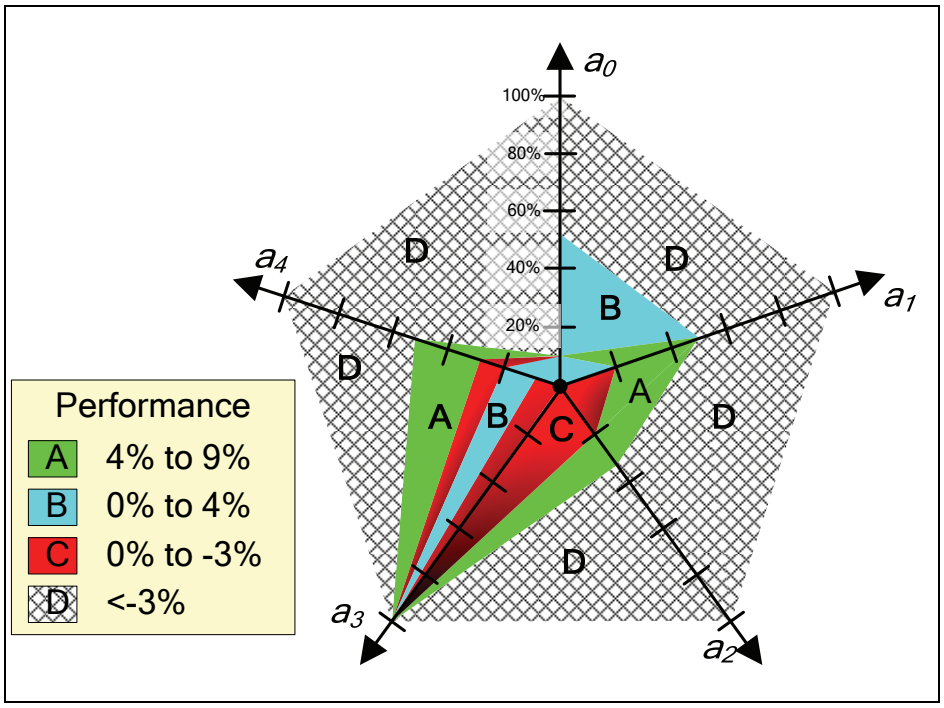

Figure III-14: Promising areas of function points

Finally, this meta-heuristic diagnosis procedure shows several advantages in terms of performance compared to modifications that needs to be edited in the models. In particular, the principle of the subfield of model based diagnostic implemented as causal networks show a significant benefit in terms of performance. Thus, the implementation of causal networks will be deeper investigated in section 2.3.

2.2.2.2 Summary and discussion

As reported in sections 2.2.1.1 to 2.2.1.5, a large number of parameters can help to improve the performance of the guided diagnostic procedure of SIDIS Enterprise. Some of these parameters can be used without changes in the knowledge networks, others like the causal links need to be implemented and edited. The question is how to combine the different factors in order to obtain a meta-heuristic that approximates a multi-criteria optimization between time and diagnosis performance? Thanks to the performed simulations, a radar diagram is depicted in Figure III-15 which shows the interesting areas for a performance gain of more than 4% and the possible intervals for the weights of each factors also denoted as the coefficients a_0, a_1, a_2, a_3, a_4 in:

$$SM_{node} = a_0 \log_2\left(o + \frac{Level(node)}{Depth(tree)}\right) + a_1 \log_2\left(o + \frac{Tc(node)}{Tc_{\max}}\right) +$$

$$+ a_2 \log_2\left(o + \frac{LT(vehicle)}{LT(node)}\right) + a_3 \log_2\left(o + \frac{\sum_i e_i GS_i}{\sum_k GS_k}\right) + a_4 \log_2\left(o + \frac{\sum_i e_i GC_i}{\sum_k GC_k}\right)$$

(Eq. III-23)

With the following notations:

- *Level(node):* the depth in the diagnostic object tree of the considered node.
- *Depth(tree):* the depth of the diagnostic object tree.
- *Tc(node):* the cost of the test of the associated diagnostic object node.
- Tc_{max}: the maximal test cost over the whole set of tests in the tree.
- *LT(vehicle):* the km reading of the vehicle.
- *LT(node):* the life time in *km* of the corresponding node.

- e_i: equal to 1 if the symptom indexed by i is active, and 0 else.
- GS_i: is the weight of the symptom indexed by i.
- GC_i: the weight of the causal link indexed by i.
- e_i: a boolean value equal to 1 if the causal successors of the node is already suspected and 0 else.
- a_0: Weight of the precision factor.
- a_1: Weight of the test cost factor.
- a_2: Weight of the life-time factor.
- a_3: Weight of the partial moment of suspicion rules.
- a_4: Weight of the partial moment of the causal relationships.
- o: an offset equal to 1.

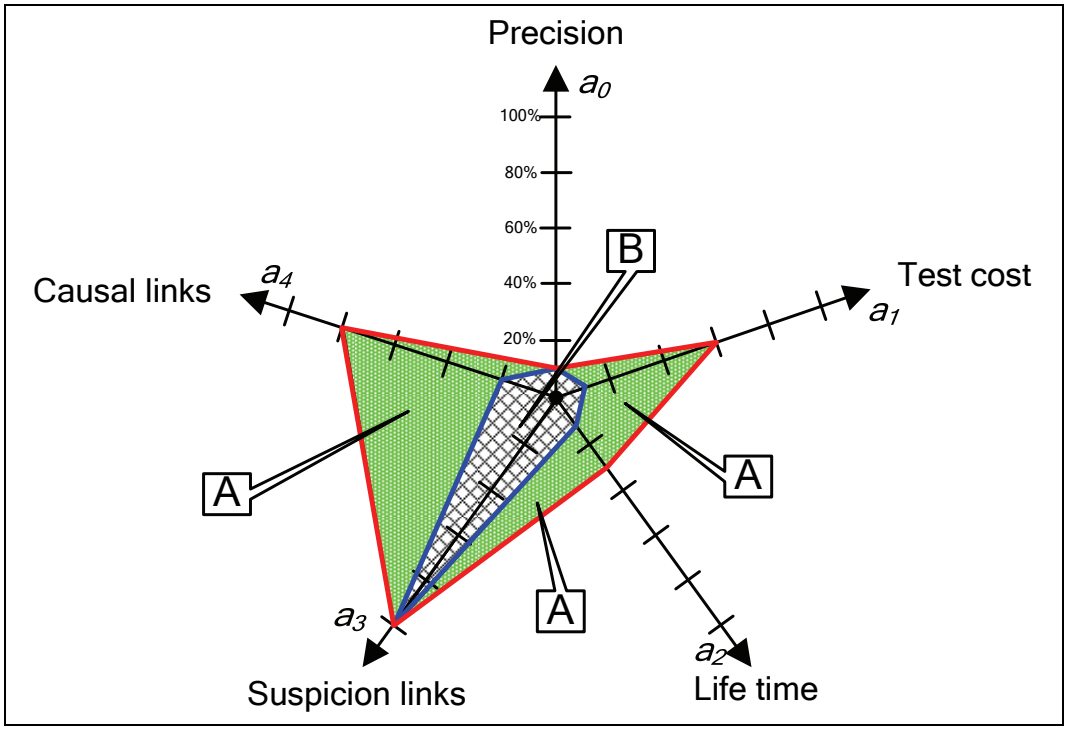

Figure III-15: Delimited areas with possible performance gain

As shown in Figure III-15 (zone indexed by A), two factors have a clearly delimited area: the partial moments of the precision and the suspicion rules, which should be multiplied by a factor of 10% and 100%. The integration of the precision allows only a small information gain in the diagnosis session. This factor is not related to any diagnosis knowledge, which explains why it has to be balanced with a low weight compared to the other ones. The crucial role of the partial moment issue from the evaluation of the suspicion rules has clearly been identified in many simulations. This factor still remains the core of the diagnosis engine, which explains why the weight must reach the highest possible value of 100% (cf. zone indexed by B in Figure III-15). Simulations where the coefficient a_3 was diminished, show deceiving results in terms of performance and have not been reported for this reason. The life time is another factor that must be taken into account only slightly; it helps for the fault localization, but if it is too much weighted the results are disappointing. The causal links have already been outlined in the introduction of model based diagnosis methodologies for possessing several advantages. This tendency is confirmed through the executed simulations where the confidence interval of the related factor to the causal links is comprised between 20% and 80%. The reason why this interval is so large lies in the model used for the simulation. This model only contains 9 causal links. Compared to the 28 suspicion links this is only 30% of the original hardcoded diagnostic knowledge of the model. But the weight to affect to this factor should remain lower than the upper bound of the proposed interval due to the fact that in a complete model, causal relationships may exceed the number of expert rules. Thus, they will increase the suspicion moment *SM* of a much

higher number of objects, which increases the needs of discrimination between the objects with the consequences that more tests should be executed. The last factor that was integrated in the meta-heuristic was the tests costs (the weight of this factor is always negative as explained in section 2.2.1.3). They are already taken into account in the calculation of the test rank, but this time they are directly implicated in the computation of the total suspicious moment of an object. It is normal that for a multi-criteria optimization of the diagnosis where the time count as one of the criteria for the cost that appear in the metric. But once again, the balance of this factor in the total suspicion moment evaluation should clearly be limited to a maximal value of 30%. If this factor is set a higher priority than the factor related to the partial moment from suspicion links, the algorithm proceed more in a decreasing ordering of the test by their respective costs than a diagnosis. The parameter denoted *o* in each factor and placed within the brackets of each logarithm is an offset, it permits to avoid non numeric results in the evaluation of each factor (because some ratios can be equal to zero causing the log to be non numeric). The global result of this algorithm can be compared to an approximation of gradient optimization, were at each step the candidates are re-ordered depending on the different factors and bounded between curves with a logarithmic behaviour. The previous paragraph (section 2.2.2.1) is concerned with the search of an optimal function point of this meta-heuristic.
Figure III-16 resumes what types of innovations are combined into the meta-heuristic diagnosis algorithm. The first type of innovation is related to product/process. Here the implementation of the meta-heuristics changes the guided fault finding process, but still improves the product. The meta-heuristic is an incremental innovation, because it sill works with the same information as SIDIS Enterprise's algorithm but it enhances the performance if other types of information are add to the knowledge structures. This algorithms proposal adds new competences, the modeling of causalities between components, which impact the architecture. Finally, there is also a type of component architecture due to the fact that the guided fault finding module needs to be re-coded. These different types of innovation are important, because SIDIS Enterprise is a commercial product. And the types of innovations have a great impact on the diffusion of the final product. The typology of the technology adoption cycle is depicted in Figure III-17 [Everett, 2003]. The impact of the choice of a backward compatible innovation or an incremental enhancement permits to ameliorate the diffusion of the new product avoiding a too large the chasm in Figure III-17. This argument is particularly important for the commercial success of the product, which is one factor to keep in mind. Another particularity is that the incremental innovations allow clients to eventually invest in the new product for the gain in terms of performance and to enhance their own competences of the after-sale world. Crossing the chasm is one of the most difficult tasks for the commercial success of a new technology. One market strategy of the new proposed algorithm could be to distribute the new product to client of SIDIS Enterprise until it spreads to more pragmatic customers.

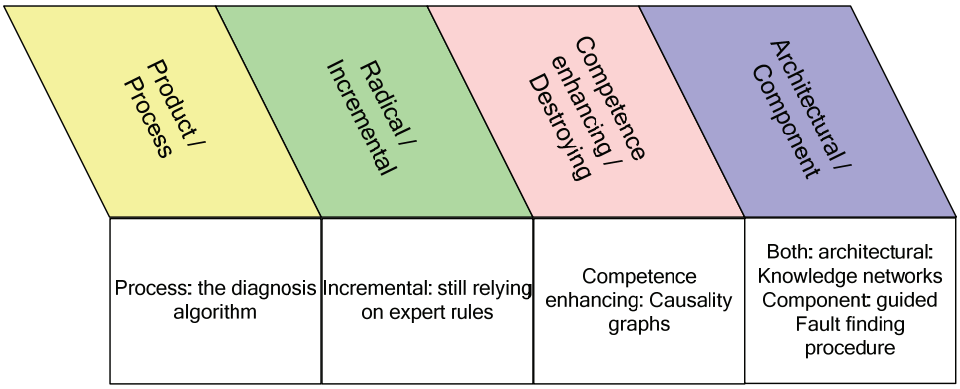

Figure III-16: Types of innovations

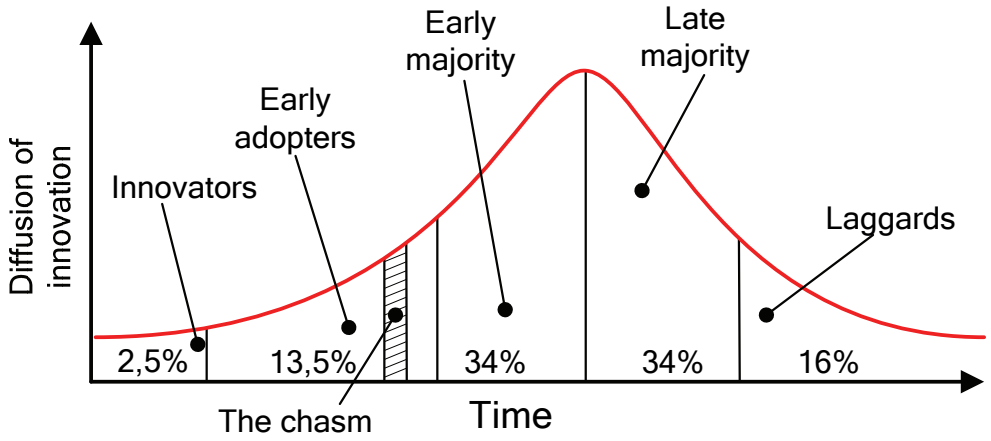

Figure III-17: Cross the chasm

2.3 A hybrid heuristic and model based strategy

One of the aims of this section is to focus on the implementation of causal relations in order to bridge the chasm between expert system and model based diagnosis to ensure a commercial success of the new proposed product. The parameters will be integrated in a different way into SIDIS Enterprise in this section and detailed description as well as the consequences will be discussed. The focus on causal graphs proofed to be efficiently on the global performance of the guided fault finding. Moreover, they are often given in FMEA documents in the automotive industry for the elaboration of the models for diagnosis. Thus, adding them into the SIDIS Enterprise knowledge network will only slightly increase the maintenance costs for a possible benefit in terms of performance.

2.3.1 Overview

In Figure III-18 a simple model is depicted, with suspected diagnostic nodes (in dark) and unsuspected ones (empty). The causal relationships in the graph (dotted arrows in Figure III-18) build an oriented causal graph representing the dependencies between causes and consequences of a failure. Causal relations can point out electrical, mechanical or thermal connections based on signal variables or states, e.g. tension, force, torque, speed or temperature. Note that these relations have to be available at every level.

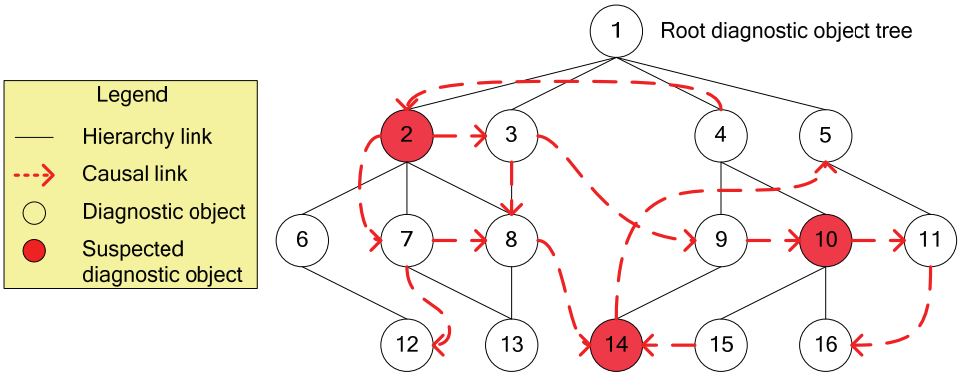

Figure III-18: Simplified model

These oriented causal graphs (OCG) can have either a qualitative or a quantitative weighting:

- **Qualitative**: Causal reference signals and states are not associated with accurate but fuzzy (e.g. very small, small, middle, high, very high) values to map the strength of dependencies between two components.

- **Quantitative**: Causal reference signals and states are associated with numbers, which can describe a large scale of importance. Moreover, it is better adapted to the comparison between two links.

In this section the causal links are considered as quantitative, because the weight coefficient will be determined with the help of a feedback engine detailed in section 3. The algorithm navigates between the suspected diagnostic objects. In the current method (cf. chapter III section 1.1), the ranking of all visited objects is based on their suspicion moments and their associated costs. With the proposed approach in this section the algorithm searches incoming causal links and evaluates their importance for each visited object. The final ranking becomes then a function of both. The principle is quite simple. The system navigates through the tree and visits every suspected object (dark nodes in Figure III-18). For each candidate the algorithm searches its causes (e.g. for candidate 14, the causes are nodes 8 and 15). In order to give more or less importance to the cause and to have a larger view on its environment, the algorithm looks then for possible previous causes of the cause. The process details and the formalizations of the proposed approach follow in the subsequent sections.

2.3.2 Formalization

2.3.2.1 Principle

The diagnostic objects are stored into a tree structure as represented by the continuous lines in Figure III-18. The integration of causal links (dashed arrows) implies that the graph becomes oriented. Such links are drawn between a cause (start node) and an effect or consequence (end node) and are transversal. As described in the previous section, the algorithm looks then for possible previous causes of the cause (e.g. for the cause 8, previous causes are nodes 3 and 7). In other terms, the system takes into account information up to two ranks of causes. For each object a *causal moment* (SM_α) is then evaluated. It is important to notice that this oriented causal graph creates a model, which predicts what can happen within the elements in the system.
Lastly, a *final moment* SM_f will be evaluated as a function of the suspicious and causal moments (respectively SM_h and SM_α); these moments will then serve as basis for the ranking. Consequently, the current system will just be slightly affected by the causal relation; the new algorithm will proceed between the calculation of the SM_hs and the establishment of the ranking. In a first attempt, both suspicious and causal moments will be separately evaluated since the suspicious one is symptom-related whereas the causal one is object-related.
The proposed development consists of implementing two fundamental equations for the candidate generation. The first one deals with the evaluation of the causal moment whereas the second deals with the final moment.

2.3.2.1.1 Evaluation of the causal moment

Because the system should emphasize the elements which are the most suspected, resp. whose causes play an important role, the idea (as in section 2.2) is to use the information theory. In particular, the information entropy given in (Eq. III-24) and which is based on the Shannon entropy [Shannon, 1948].

$$H(X) = -\sum_{i=1}^{n} p(x_i).\log_b\ p(x_i)$$

(Eq. III-24)

Where:

- X is a finite discrete probability distribution with n outputs.
- $p(x_i)$ the probability of output x_i .
- b the logarithm base (here b=2).

This entropy $H(X)$ measures the uncertainty of a distribution: the lower the entropy, the lower the uncertainty and therefore, the better the choice. In our case the notions of uncertainty or information

are related to the notion of causality of an element. The Shannon entropy has been later generalized by [Rényi, 1961] to obtain the Renyi entropies also called entropies of order α in (Eq. III-25). They are defined for $\alpha \geq 0$ but $\alpha \neq 1$ as (where H_α converges to the Shannon entropy as α tends to 1):

$$H_\alpha(X) = \frac{1}{1-\alpha}.\log_2\left(\sum_{i=1}^{n} p(x_i)^\alpha\right)$$

(Eq. III-25)

With the same parameter definition as in (Eq. III-24) and:

- α the order of the entropy.

(Eq. III-26) is then used for the evaluation of the causal moment:

$$SM_\alpha = \left|\frac{1}{1-\alpha}\log_2\left(o+\frac{Level(node)}{Depth(tree)}\right)^\alpha + \frac{1}{1-\alpha}\log_2\left(o+\frac{\sum_i e_i GC_i}{\sum_k GC_k}\right)^\alpha\right|$$

$$\begin{cases} q_1 = \left(o+\frac{Level(node)}{Depth(tree)}\right)^\alpha & r_1 = \frac{1}{1-\alpha}\log_2(q_1) \\ q_2 = \left(o+\frac{\sum_i e_i GC_i}{\sum_k GC_k}\right)^\alpha & r_2 = \frac{1}{1-\alpha}\log_2(q_2) \end{cases}$$

(Eq. III-26)

With :

- $SM\alpha$: the causal suspicion moment of a node.
- *Level(node):* the depth in the diagnostic object tree of the considered node.
- *Depth(tree):* the depth of the diagnostic object tree.
- α the order of the entropy.
- GC_i: the weight of the causal link indexed by i.
- e_i: a boolean value equal to 1 if the causal successors of the node is already suspected and 0 else.
- *o:* an offset equal to 1.

The first member of (Eq. III-26) (called factor r_1) corresponds to the precision and contributes to the information gain in a similar manner as in the meta-heuristic approach (cf. section 2.2.1.2). The higher the position of a node, the less information it contains and the lower its entropy in the tree. This factor benefits to precise fault localization compared to candidates, which have a lower level (or objects that are nearer to the root of the tree). The denominator of the ratio contributes to a normalization of the level of the nodes in the tree, so that the ratio is comprised between 0 and 1. On the other hand, the second member of (Eq. III-26) (called factor r_2) deals with the impact of the previous causal ancestors (like backward chaining, similar to the meta-heuristic approach described in section 2.2.1.5); a cause with three active heavy weighted previous causes has definitely not the same importance as one with one light weighted previous cause. Here, GC_i represents the weight of the previous causes and e_i equals 1 if the previous cause is active, 0 otherwise. If no previous cause is active, then the expression equals 0 (thanks to the offset).

2.3.2.1.2 Evaluation of the life time

As discussed the lifetime parameter of one component plays an important role for the diagnosis (see sections 2.2.1.4 and 2.2.2.2). Previously, the influence of the lifetime has been modeled by a logarithmic function which contributes to the suspicion moment of a candidate. The usual curve of the lifetime of an equipment is depicted in Figure III-19, where three different periods can be recognized:

- Youth period, premature defects.

- Maturity period, failure rate globally constant.
- Wear lifespan, wear defects.

During the youth period, the failure rate diminishes. The probability of a defect follows the same tendency. This period corresponds to production quality defects or assemblage defects.

The maturity period corresponds to an approximate constant behaviour of the failure rate over the time. The probability of a failure is independent of the number of hours the equipment works (random defects). This period is often taken as reference for electronic equipments.

At the end of the maturity period, the probability of a breakdown increases with the number of working hours of the considered equipment. The older the equipment is, the higher the probability of a failure. This type of behaviour is a characteristic of systems that underly wear or other progressive deterioration (e.g. oxidation) causing the failure rate to grow.

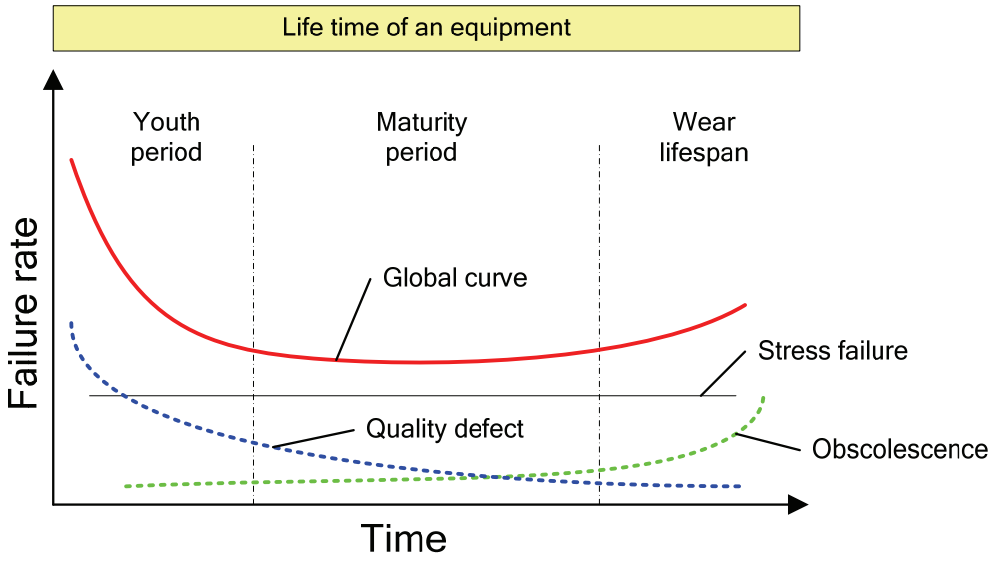

Figure III-19: Global curve of the lifetime of an equipment

The part of the curve which is significant in the life time curve is the second and last period. It is well known that for electronic equipments the youth period of equipment (often due to quality defects) are relatively small. The modeling of this part of the curve is done with a threshold function as described in (Eq. III-27) called Heaviside function. This function is equal to zero for negative values of the argument and to 1 otherwise.

$$PM_{LT} = Heaviside(LT(vehicle) - LT(node))$$

(Eq. III-27)

With:

- PM_{LT}: the partial suspicion moment of a diagnostic object node induced by the life time behavior.
- *LT(vehicle):* the *km* reading of the vehicle.
- *LT(node):* the life time in *km* of the corresponding node.

Here the parameter is the difference between the current km reading of the vehicle and the prior life time of a considered component. In order to avoid failures, if a component has been replaced, the km reading of the vehicle needs to be changed by the difference between the current km reading and the km value when the component had been changed. This information could be accessed with the help of a central database containing the vehicles historic.

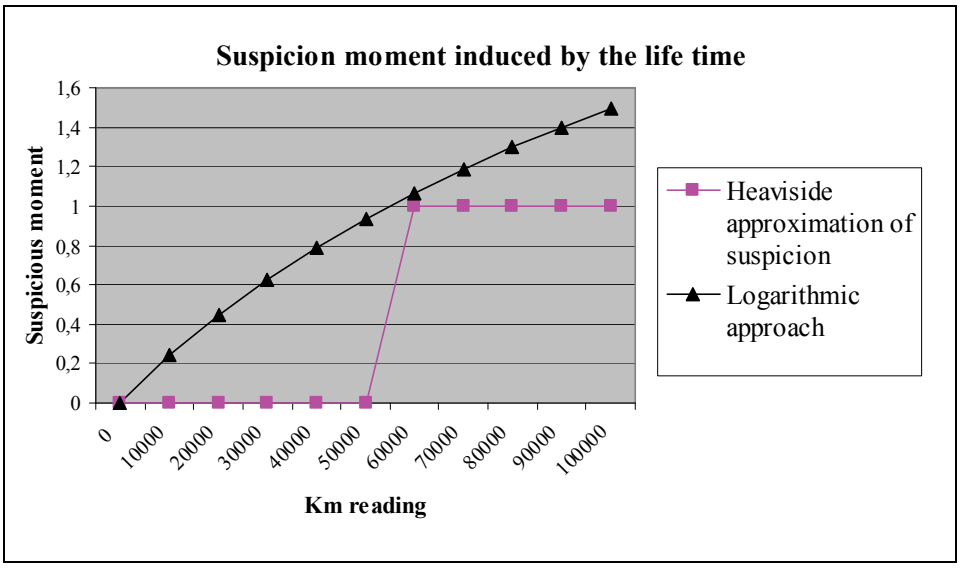

Figure III-20: Suspicious moment induced by life time

Figure III-20 shows the evolution of the suspicion of a component with a life time of 55.000km with the Heaviside approximation function (square curve) and with the meta-heuristic approximation using a logarithmic function (triangle curve). Instead of a continuous increase of the suspicion moment as given by the meta-heuristic function (see section 2.2.1.4), the Heaviside approximation acts as a threshold and fired a suspicion only if the prior life-time of a component is exceeded. This approximation is therefore more suited due to the fact that it has an upper bound and imitates more likely the general behavior of the lifetime periods of electronic equipments. The next section examines how to combine this factor with the causal suspicion.

2.3.2.1.3 Evaluation of the final moment

The two moments (SM_h obtained by the current algorithm of SIDIS Enterprise and SM_α computed with (Eq. III-26)) obtained in the previous steps must now be put together in order to get a final moment for each object called SM_f. A first solution consists of a weighted sum of both moments with β a coefficient defined in [0; 1] as in (Eq. III-28). However, (Eq. III-28) can be generalized as proposed in (Eq. III-29) (with γ positive) so that it can be adapted to some object's specifications.

$$SM_f = \beta SM_h + (1-\beta) SM_\alpha + c_{LT} PM_{LT}$$

(Eq. III-28)

$$SM_f = \beta^\gamma SM_h + (1-\beta)^\gamma SM_\alpha + c_{LT} PM_{LT}$$

(Eq. III-29)

With:

- SM_h : the heuristic suspicion moment computed thanks to the evaluation of the expert rules by the current algorithm of SIDIS Enterprise.
- SM_α : the causal suspicion moment .
- β: A real number comprised in the interval [0; 1].
- α the order of the entropy.
- γ: A real positive number.
- PM_{LT}: the partial suspicion moment of a diagnostic object node induced by the life time behavior.
- c_{LT}: the relative influence of the life time suspicion moment.

2.3.2.1.4 Discussion

In this paragraph the influence of the different factors is discussed except the life time factor which is been described in section 2.3.2.1.2. The influence of this factor can be set through the value of the coefficient c_{LT}, which is discussed later in section 2.3.3.1. On one hand, setting the denominator of the ration in factor r_1 as the depth of the tree is unfair since every branch does not reach the maximal depth. A node which is at a local maximum represents the best fault localization whereas by using the depth of the tree, the node is supposed to contain more information or possibilities than it actually has and is consequently not considered to its full potential by the algorithm. Therefore, it could have been wise to implement a method which evaluates the depth as a local maximum. Such an algorithm distinguishes different sub trees and could be based on *Breadth-first* search algorithms or on the works from [Gessel, Sagan, 1996], [Ruml, 2002] on *Depth-first* search algorithms or with the help of the method developed by [Gessel, 1995] to count the number of graphs. However, these options permit to make a global comparison between all nodes, which is particularly important during the calculation of the prime test agenda, which role of is to cut the tree in different search areas. The left side of Figure III-21 contains a graph of the behavior of the factor r_1 for α varying between 0.1 and 0.9 and for nodes of different levels with a maximal depth of the tree equal to 5. The right side of Figure III-21 contains the behavior of the factor r_1 in function of the level of the nodes for different values of α. As predicted the entropy contained in a node grows with higher values of α and for a fixed α and an increasing level of the node the entropy has a quasi linear behavior.

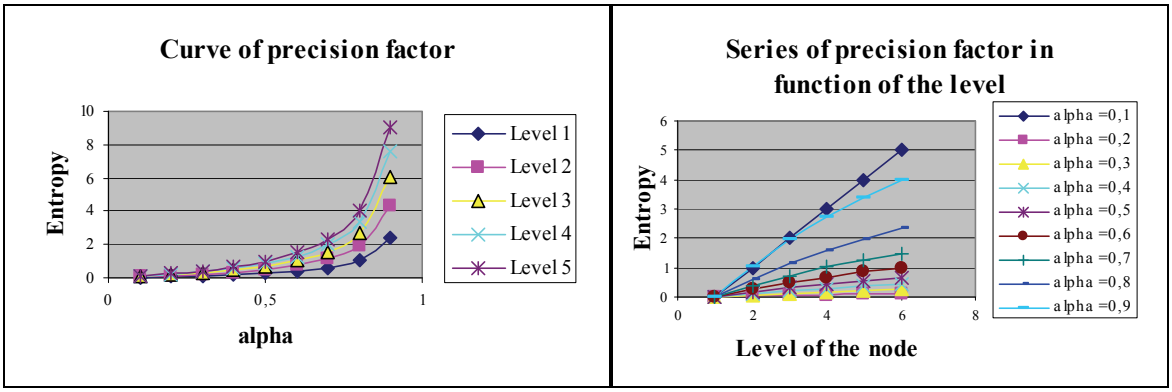

Figure III-21: Precision factor behavior

On the other hand, the second factor r_2 has much more variations. Indeed, if a node has necessarily a given position in a tree, an object may not have previous cause or active previous cause. In this case the factor will be equal to 0. On the opposite, if the only previous cause is active, the factor will have a high value depending of the order of the entropy, which makes sense, because only one active cause is a certainty: that the origin of the causal link is the cause of the failure of the suspected component. This scenario can be shown on the right side of Figure III-22, where for all different values of α the entropy has a maximum for 1 active cause. The case where a node posseses multiple active and inactive causal ancestors, divides the distribution of the suspicion depending of the weights of the incoming causal links. In other words it calculates a probability of failure. The left side of Figure III-22 contains a chart where depending of α, the behavior of the causal factor is plotted for a node with one unique active cause (highest curve) and then two nodes with one active cause and respectively one and two inactive causes.

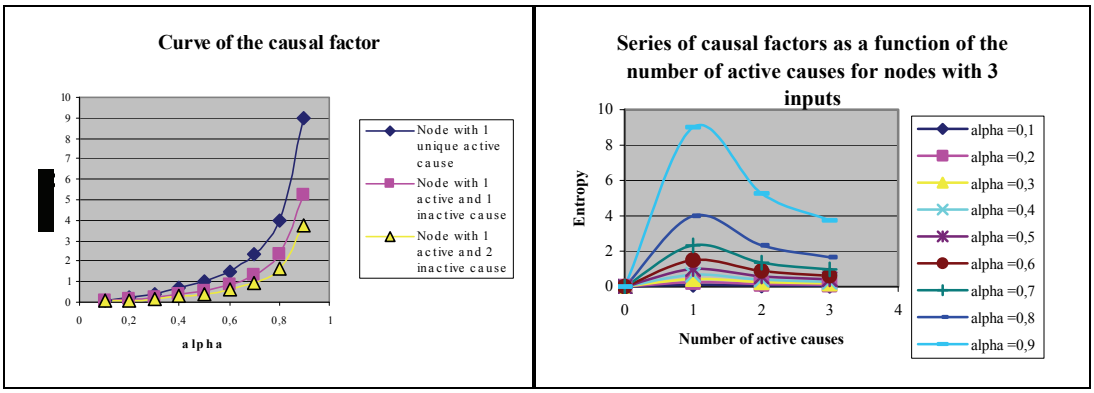

Figure III-22: Causal factor behaviour

Finally, the equations of the calculation of the final moment need to be examined. The first one (Eq. III-28) is a simple balance between the heuristic moment and the causal moment with a parameter β according more or less importance to the one or the other. But the other formula from (Eq. III-29) is more complicated. Figure III-23 depicted the graph of the final moment depending on β from 0 to 1 and γ from 0 to 10 for a heuristic moment of 50% and a causal moment of 50%.

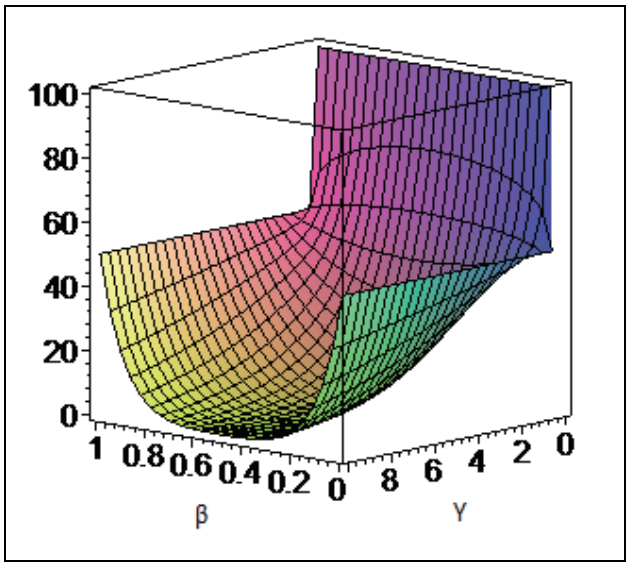

Figure III-23: Final moment behaviour

The shape of the final moment is a U-curve except for small values of γ. The smaller β, the higher the importance of the causal moment and if β tends to 1 then the heuristic moments are privileged and the causal moment is neglected. The parameter γ defines the decrease from one moment to the other with a symmetry axis in the plane β =0.5 (if heuristic and causal moment are different then the U shape is asymmetric). Globally, the important intervals for the curve are located on both extremities of the interval [0,1] for β and for γ from 2 to 6. The reason is that in this area the U shape is more likely a parable until it becomes a U. Thus, these properties permit to change the importance of the causal evaluation or the heuristic one. For a better comprehension of the interaction of these parameters section 2.3.2.2 will detail an example.

2.3.2.2 Example

This section quantifies how much the final results are affected by the parameters. The tests are performed on a modern vehicle, which has about 2000 diagnostic objects and whose vehicle tests (test performed by the technician to get all the fault codes generated by the ECUs) last approx. 20 to 30 minutes. However, because of the time required for a vehicle test and the restricted number of objects

in the external database, only the perceived symptom "*Engine does not start*" has been selected since it suspects 14 objects, which are depicted in the sub tree of Figure III-24, on the left side. The subsequent section investigates deeper the influence of the three parameters α, β and γ. The hierarchical relations between the objects are depicted on the left side and the causal relationships and their weights on the right side of Figure III-24. The complete details of the model used in this example are given in appendix A section 3.4.1.

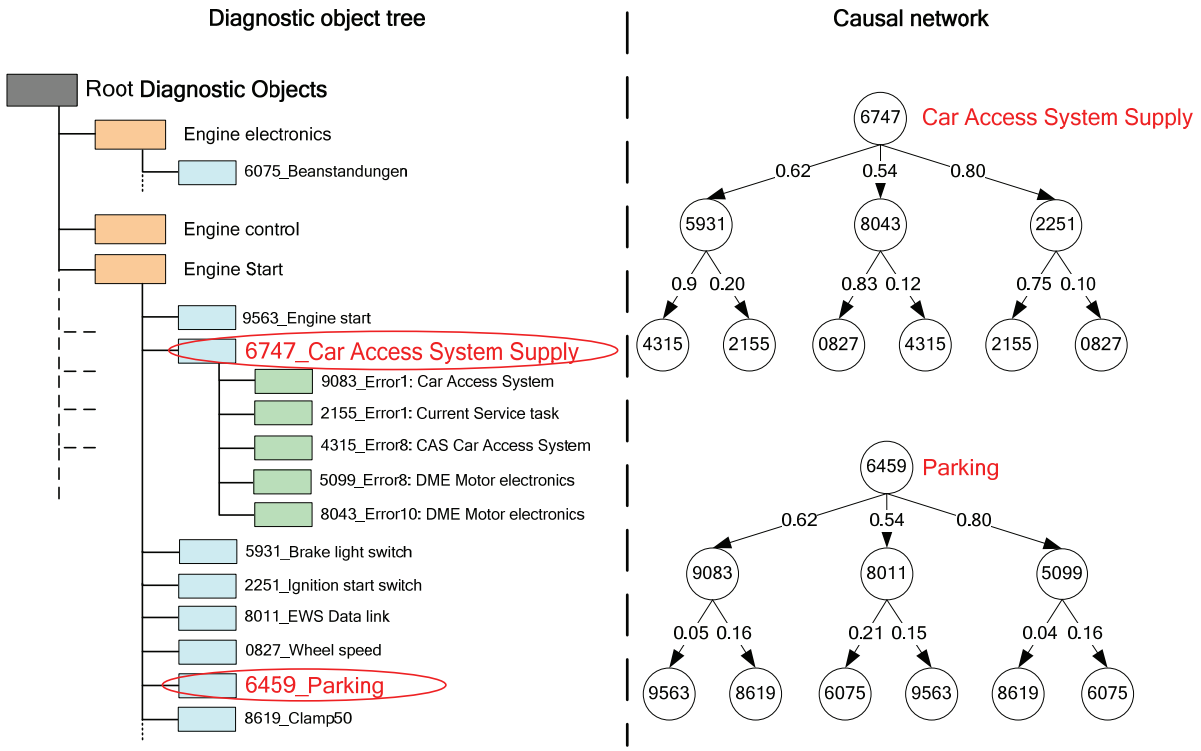

Figure III-24: A simplified scenario

Figure III-25 shows the evolution of the causal moment for different values of α. As predicted by the theory in section 2.3.2.1.4, the moment increases exponentially with α and the δ curve (which is the Euclidian norm of the vector composed by the causal moment of the objects of the prime test agenda) increases as well with the same shape, meaning that the higher α, the higher the discrimination power of the formula.

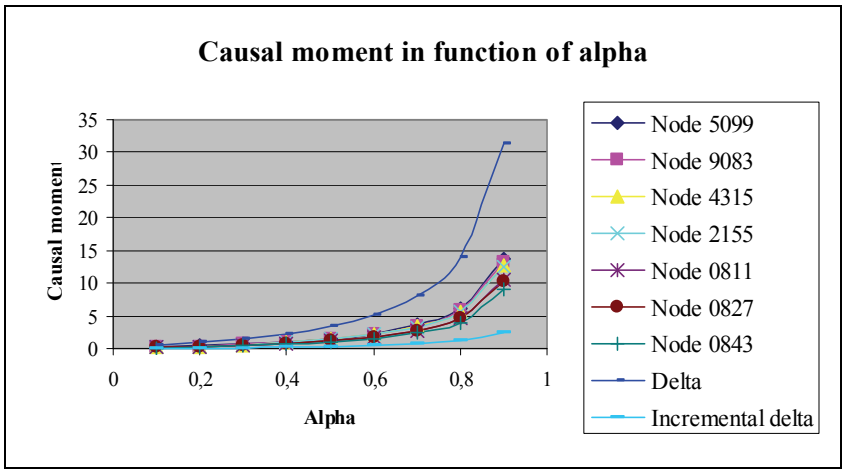

Figure III-25: Evolution of the causal moment in the prime test agenda

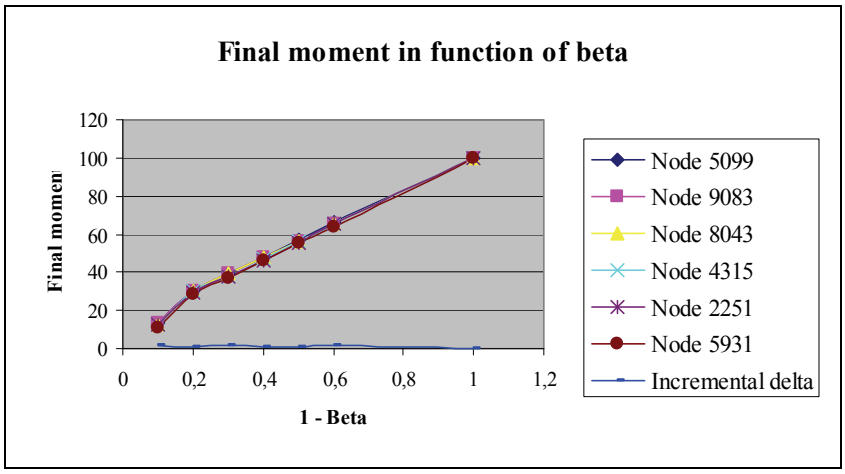

Figure III-26: Evolution of the final moment in the prime test agenda

Figure III-26 shows the evolution of the final moment of the 6 first elements in the prime test agenda for different values of the coefficient β. Both figures (Figure III-26 and Figure III-25) contain a curve called "incremental delta", which is the Euclidian norm of the difference of the moment of two successive objects in the test agenda. This curve gives precious information about the discrimination power of a considered algorithm and its parameters. As depicted, the incremental delta increases with higher values of α and for high values of β (meaning a higher importance of the causal moment). In this example the combined heuristic and causal moment influences the ranking in the prime test agenda. Due to the fact that only one symptom is active, and that all expert rules have the same weights, the incremental delta is equal to zero. Thus, the causal moment influences the ranking, in particular for the intermediate objects in the causal network which have highly weighted active incoming links as nodes 5099 and 9083. It is to notice that also leaves of the diagnostic object tree are privileged in the diagnosis thanks to the factor r_1 in (Eq. III-26). In this particular case where the node 4315 (CAS car access system) is responsible for the breakdown, the combined heuristic and causal search diminished the orbit of the guided fault finding procedure to 1.

2.3.3 Synthesis and examinations of function points

In the following subsections, a deeper investigation of the parameters is done in order to achieve the optimal weight coefficients of the parameters of the final moment: α, β, γ and c_{LT}. The same model as in section 2.2.2 is used for the simulation as well as the same case base of breakdowns, except that the last case is not used, reducing the average values to a cardinal of 14 cases. This is due to the fact that the last case is an evident one with active suspicion rules pointing directly to the parent diagnosis object of the broken one. In order to obtain a better comparison study this case is ignored. The next subsection (cf. 2.3.3.1) is concerned with the examination and the consequences of the variations of the previous mentioned parameters: α, β, γ and c_{LT}. Section 2.3.3.2 summarizes the results and concludes over this new diagnosis approach.

2.3.3.1 Examination of function points of the parameters

The left side of Figure III-27 shows the evolution of the average orbit of the guided fault finding procedure calculated over the 14 cases and for α varying between [0, 0.9]. The minimal value occurs for β =0.4 with an average of 6 tests which is slightly higher than the results obtained through the meta-heuristic diagnosis with a weight of 100% for the suspicion link factors and 40% of the causal relationship evaluation (equal to 5.78 in average). It is interesting to notice that during the simulation the factor α has a low influence on the test agenda (as detailed in the appendix A section 3.4.2), which is also confirmed for extreme variation from 0 to 0.9. The factor β has a much higher influence on the test agendas as depicted on the left side of Figure III-27. The interesting area of the factor β is quite similar as the factor a_5 of the meta-heuristic diagnosis approaches, the interval [0.3, 0.4].

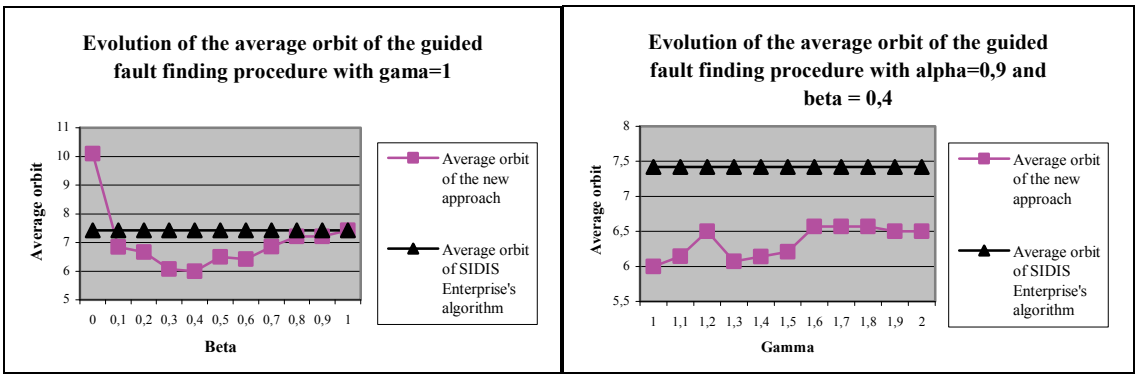

Figure III-27: Evolution of the average orbit of the guided fault finding procedure

The right side of Figure III-27 shows the variations for the average of the orbit of diagnosis sessions for different values of gamma. The standard deviation of the orbit is equal to 0.22, which confirms the small influence of the factor γ. This factor is very specific to the different cases that are considered and the best way to choose a value for this factor is to consider a case-specific value depending on the information amount of the model. The last parameter that is examined in this section is the influence of the life time and the factor c_{LT}. Figure III-28 shows the evolution of the average cardinal of the guided fault finding procedure for different values of the life time factor c_{LT}. As with the meta-heuristic approach the optimal interval if for values between 0.2 and 0.3 with a minimum of the average orbit (5.85) for the lower bound of this interval. Over the value of the upper bound, the suspicion induced by the life-time takes too much importance in the diagnosis engine and finally normal working components checked causing the orbit to increase.

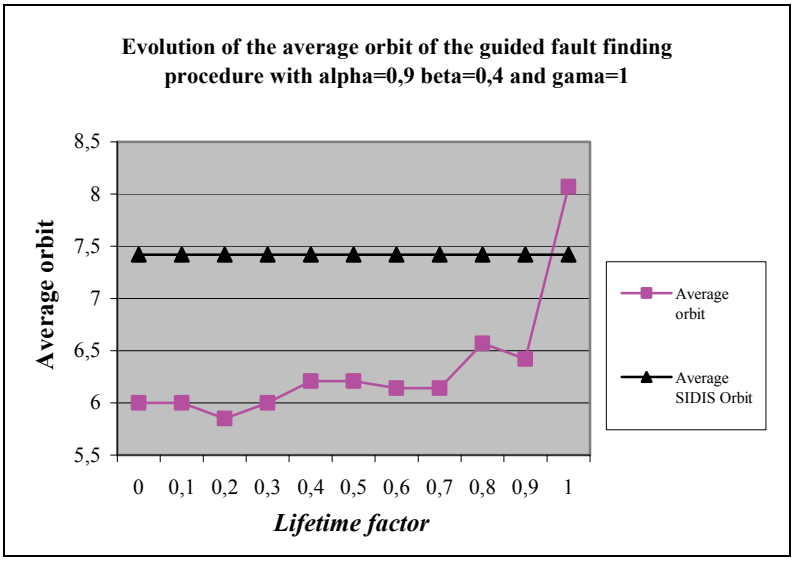

Figure III-28: Evolution of the average orbit with lifetime

As a result the function point of this combined approach for the parameters α, β, γ and C_{LT} are in the intervals $[0.8, 0.9]x[0.3, 0.4]x[1, 1.2]x[0.2, 0.3]$. The tolerance margin is due to the case base and will be deeper investigated in chapter IV.

2.3.3.2 Synthesis and outlook

The presented approach combines two different sources of diagnosis knowledge: on one hand the expert knowledge via the suspicion rules and on the other hand the model based knowledge via the causal links and the life time of the component. The causal links increase the discrimination capacities of the candidates depending on their weight. The interpretation behind a link like A → B is "A may cause B", associated with correct weight coefficients corresponding to probabilities obtained through

frequency analysis these links increase the fault localization procedures with a performance gain of 19%. Moreover, the function used for the suspicion moment of the life time is more realistic than in the meta-heuristic approach. The bathtub curve is approximated with a Heaviside function, which represents more likely the behavior of electronic system's defect rate. As a conclusion the combined parameters of this approach leads to a performance gain of 21%, but this innovations has an impact on SIDIS Enterprise's architecture. There is also a type of component architecture due to the fact that the guided fault finding module needs to be re-coded. Despite these two major innovations, this new approach still remains an incremental improvement, which could enhance the diffusion of the technology, contributing to the commercial success.
The combined approach leads to a bridge between both worlds of heuristic and model based diagnosis closing the gap in the technological adoption cycle depicted in Figure III-17 with an increase of the overall performance of 21% for the orbit of guided fault finding procedures.

3 Inductive learning

3.1 Current Situation

As a major subfield of artificial intelligence, machine learning has gained a broader attention in recent years. Especially the internet makes a variety of information more easily accessible than ever before, creating strong demands for efficient and effective methods to process a large amount of data in both industries and scientist research communities. The application of machine learning methods to large databases is called *Data Mining* or *Knowledge Discovery* [Alpaydin, 2004]. Despite the existence of hundreds of new machine learning algorithms, from a machine learning academic researcher's perspective, the current state of this research is very preliminary. If the ultimate goal is to develop a machine that is capable of learning at the level of the human mind, the journey has only just begun [Silver, 2000]. This section does not explore all areas of artificial intelligence but one step in that field by exploring the outstanding questions in inductive learning in a manner to optimize the diagnosis algorithm:

- Which values have to be learned to optimize the diagnosis algorithm?
- How can an algorithm learn the values of the parameters?
- How can the learned values be balanced with prior knowledge?

The vast majority of standard inductive learning machines are data driven, which means that heavy algorithms are working on the sample data and ignore most of the existing domain knowledge. The reasons why the majority of machine learning techniques ignore domain knowledge vary:

- Researchers try generally to obtain generic algorithms rather than domain-restricted techniques.
- Domain experts often ignore how learning techniques work and thus cannot evaluate or incorporate their knowledge.
- There is no universal representation of domain knowledge (except ontologies which may go in that direction in the future).
- It is difficult to encode knowledge in a computer and find a way that algorithms take the valuable part of the domain knowledge into account. This ground will cause the investigations to focus on techniques and methods that do not need knowledge.

At the same time, in certain domains, the researchers in machine learning often have little idea of certain existing knowledge. In the Figure III-29, three points stand for three training examples. In the case of standard machine learning, a smooth curve is learned to go across three points (upper box of Figure III-29). But if some domain knowledge exists, e.g. gradients at each point, the resultant curve needs not only to go across the points, but also takes into account the gradients. Thus, the resultant curve (lower domain of Figure III-29) differs very obviously from the previous one.

In the problem described in Figure III-29, the domain knowledge, i.e. the gradient, could be treated as some additional constraints apart from the location of the data points. Under the optimization theory,

these gradients could be represented as additional constraints. In some machine learning algorithms, the loss functions, e.g. least squares error, can be modified to contain those additional components to penalize the violation to these constraints. Therefore, through these additional constraints with certain weights, the final result will be influenced by this particular domain knowledge.

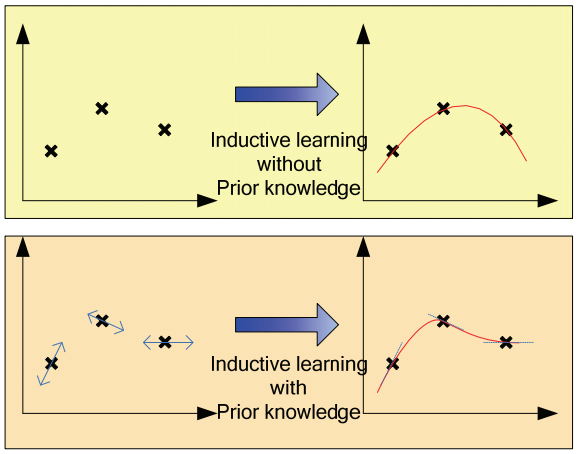

Figure III-29: Learning with and without prior knowledge

A precise definition of machine learning has been given by [Plant, Murell, 2007]:
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."
As a branch of machine learning, inductive machine learning is an inference process consisting of two major steps:

- A learning system L constructs a hypothesis space H from a set of training examples (or observations) S, for example a set of training examples consisting of input-output pairs $S = \{(x_i, c(x_i))\}$, and the inductive bias B predefined by the learning system and the task T.

- The learning system then searches through the hypothesis space to converge to an optimal hypothesis f consistent with training examples and also performing well over other unseen observations.

The inductive learning process can be seen as a process of function approximation, in other words, an inductive learning system generally has a capability of constructing a rather complex model $f(x)$ to approximate any continuous or discontinuous unknown target function $c(x)$, i.e. $f(x) \rightarrow c(x)$, with sufficient training examples S. As detailed in the previous sections the parameters that require be learning or adjusting through machine learning are listed bellow:

- Optimization of the perceived symptom search engine.
- Learning or calibration of the test costs.
- Learning of the life time of the components.
- Optimization of the weights coefficients of suspicion rules and of causal links.
- Learning dependencies between components.
- Learning test coverage and test confidence.

The subsequent sections explore some usual methods of learning algorithms that could contribute to reach the objective of this thesis under the constraints that they can be implemented at the lowest cost. The first explored method is the Bayesian network, because they are known to have the strong advantage to intuitively explain dependencies for a human. The second investigated method is decision trees for the reason that they provide a white box model, where a result can be easily replicated through simple mathematical rules. The last subsection is dedicated to the feedback of the perceived symptom search engine in natural language. All the 3 methods have in common that they can be run within the architecture schema depicted in Figure III-30. In order to save and prevent

maintenance costs the data preparation for the learning algorithms should occur as far as possible in the workshop system. The results are written in an XML protocol, which is sent to a feedback evaluation server. This server realizes the evaluation for the parameters listed previously and can take a configuration file into account, which can for example contain threshold values for the acceptance or the rejection of a hypothesis or a minimal number of protocols for certain objects. The XML protocols are read and depending of the prepared data, stored in datasets (the protocol could also be automatically deleted, once the values have been copied). The implemented export procedure starts the evaluation of the data stored in the datasets, depending of the configuration file and exports the learned results into an XML file. This last one is compatible with the standard file schema used by the data import assistant of SIDIS Enterprise, which is used to re-import the data in the authoring system. Here, the manufacturer can automatically import the new values in the AS servers (like an incremental update) or import the data into one (or more) standalone server. The goal here is to eventually have a data post-production test or verification (by domain experts, or by tester on vehicles). This step allows to delete eventual irrelevant links or parameters and create a new publication package, which can be set on the update server, or create a new XML file, which can be imported on the AS servers of the manufacturer. During this thesis, a prototype was developed, which regroups the different investigated modules, an external feedback evaluation server as depicted in Figure III-30 was not developed.

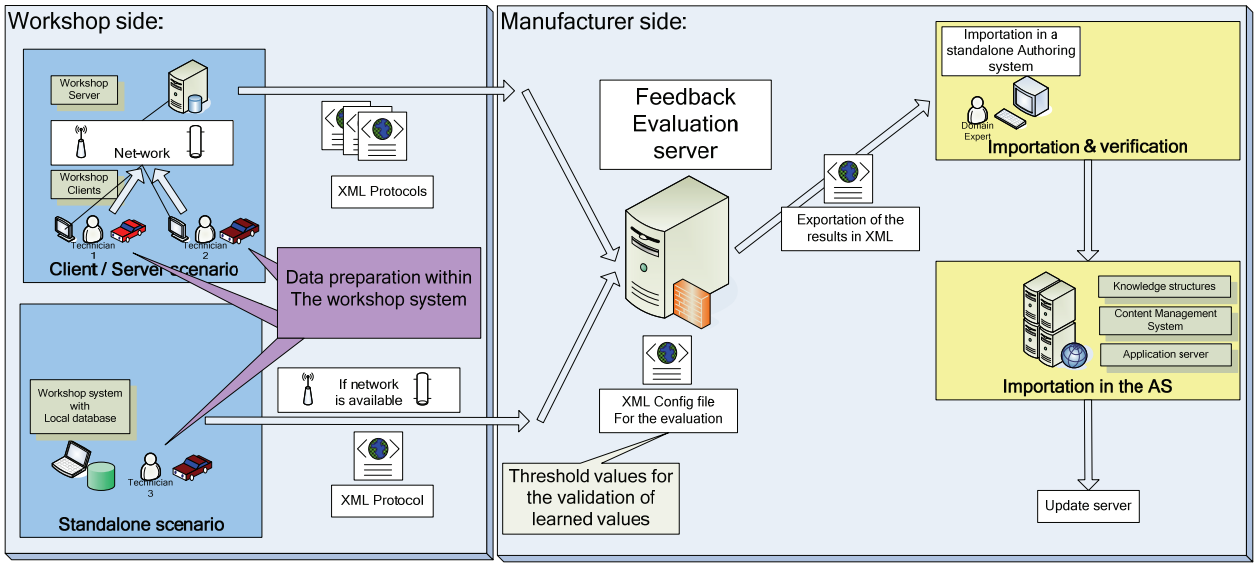

Figure III-30: Architecture of the feedback engine

3.1.1 Bayesian networks

A Bayesian network or belief network is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a Directed Acyclic Graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between components failure modes and their symptoms. If the symptoms are known the probabilities of the failures modes can be computed via forward chaining in the belief network. This sort of approach fits particularly well with SIDIS Enterprise's structure because the arcs of a Bayesian network can be considered as the suspicion rules or the causal relationships between components except that here the diagnosis knowledge is modeled through forests of trees. Formally, Bayesian networks are graphs whose nodes represent variables and edges represent conditional independencies between variables. In order to illustrate the Bayesian network, the strategy will be applied on an abstract example based on the knowledge structure of SIDIS Enterprise. In Figure III-31 a diagnostic object tree is depicted with 3 symptoms, which represent the beginning of a guided fault diagnosis session. The nodes *n2* and *n5* answers 'NotOK', but nodes *n3* and *n4* answers 'OK' despite the fact that they are candidates. The guided fault finding procedure of this example begins with the node *n4*, then *n3*, *n2* and *n5*, which are ranked first in each corresponding test agenda.

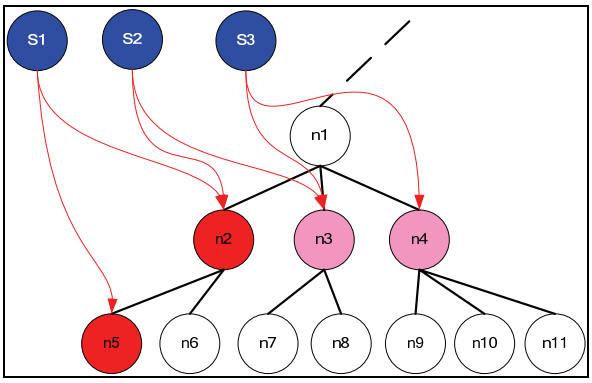

Figure III-31: Example for Bayesian networks

In this example, it is supposed that only default values are affected to the different weight parameters and no causal links have been modelled. Bayesian networks make sense to learn the following posterior probabilities:

- Optimization of the weights coefficients of suspicion rules, symptoms and of causal links.
- Learning dependencies, suspicion rules between components.
- Test coverage and test confidence.

The rank of a suspicion rule can be easily optimized with the posterior probabilities as already observed in [Jaeger, 2004]. For a given symptom or fault code a suspicion rule is taken. Based on all feedback protocols and the results of the associated test to the incriminated diagnostic object by the rule a new rank can be calculated based on:

$$R = \frac{\sum_{i=1}^{k} P_{NotOK\,i}}{\sum_{i=1}^{k} P_{NotOK\,i} + \sum_{i=1}^{k} P_{OK\,i}}$$

(Eq. III-30)

with:

- R: New rank of the considered suspicion rule.
- k: Total number of feedbacks containing the considered suspicion rule.
- P_{notOK}: number (1 or 0) for ‚NotOK' answers for the tests associated to the component targeted by the rule.
- P_{OK}: number (1 or 0) for ‚OK' answers for the tests associated to the component targeted by the rule.

Considering that the previous example delivers a feedback protocol which is analyzed, in particular the Symptom S2 and the rules n°1 pointing to node *n2* and rule n°2 pointing to node *n3* with both a default rank of 1. The details of the calculation of the new rank for both rules is given by:

$$R_1 = \frac{\sum_{i=1}^{k} P_{NotOK\,i}}{\sum_{i=1}^{k} P_{NotOK\,i} + \sum_{i=1}^{k} P_{OK\,i}} \underset{[k=1]}{=} \frac{P_{NotOK}\{n_2\}}{P_{NotOK}\{n_2\} + P_{OK}\{n_2\}} = \frac{1}{1+0} = 1$$

$$R_2 = \frac{P_{NotOK}\{n_3\}}{P_{NotOK}\{n_3\} + P_{OK}\{n_3\}} = \frac{0}{1} = 0$$

(Eq. III-31)

It must be outlined that if the model in the example is designed only to diagnosis this one case, then the second rule has a rank equal to 0, because it is considered as useless. This property can be very

useful in case of models, which are too complicated to maintain. The statistical analysis can provide to clear the model of all useless rules which do not help the diagnosis engine. On the other hand the rank of the first rule is equal to one, which is normal, because it leads to the defect component. The same reasoning can be applied to adapt the values of the test confidence and the test coverage (which will be implemented in the global framework and detailed in chapter IV). The weights of the symptoms can be adjusted with the same formula except that the classifier 'NotOK' and 'OK' are replaced by 'acute' and 'in-acute'. This evaluation between acute and in-acute is detailed on an example in chapter IV section 2.2.4 as well as considerations about the structure of a protocol.

For the learning of structures the Bayesian network considers that all possible suspicion rules are present in the model and also all causal links with a weight of 0. The protocol sent to the central server will then evaluate the probability of acuteness of the link or the causal relationship. This is called the naive hypothesis where the search space is known to be **super-exponential,** but it has the strong advantage to help the authors in the end. Other optimized structure learning algorithms forget some arcs or place the arcs at the wrong nodes. A complete synthesis over the performance of structure learning Bayesian methods can be found in [Francois, Leray, 2004]. Despite the possible automatic learning method and its implementation in the authoring system, the results must be analyzed carefully. The discovered rules correspond to hardcoded diagnosis knowledge and therefore to a physical reality. Their definitive validation should always be realized by an expert. An acceptance limit could also be configured for example in terms of number of protocols. A rejection limit should also be defined (due to the naïve hypothesis) for example a causal link that had a weight of 0.0001 can correspond to a case that occurs 1 time for 10.000 feedbacks! A reasonable acceptability limit for the weight is 20%, because it is around this limit that links have an impact of the test agendas. But depending on the size and the politic applied in a car manufacturer after-sales network (e.g. the protocol can be sent optionally or the protocol must be sent to the feedback server), that limit should be configurable. In the general case an acceptability limit around 10% less of the lowest weight is enough to keep the relevant links for the diagnosis.

3.1.2 Inductive decision trees

Decision tree learning, used in machine learning, uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value. More descriptive names for such tree models are classification trees or regression trees. In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making. This section deals with decision trees in data mining especially for the learning of the test costs and the life time of the component.

Basically, the goal of a decision tree is to create a model that predicts the value of a variable based on several input variables. Each interior node corresponds to one of the input variables, with edge to children for each possible value of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf. In order to illustrate the principle of decision trees an example is examined for the determination of the life time of a component.

If for a diagnosis session the tested components and the km reading of the vehicle is sent to a central server very simple mathematical operators on the occurrence permits to build the decision tree from Table III-7:

Case number	Test of COMP i answer	km
1	OK	15000
2	OK	20000
3	OK	17000

4	OK	18000
5	NotOK	17.000
6	NotOK	16000
7	NotOK	14000
8	NotOK	12000
9	NotOK	9000

Table III-7: Protocol table

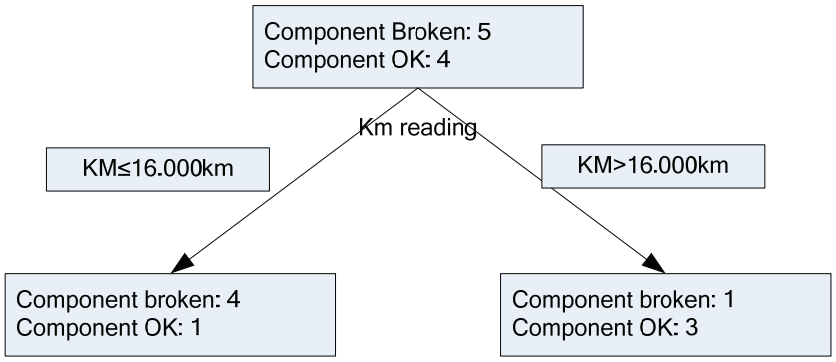

Figure III-32: Decision tree example

These frequencies permit to construct the hypothesis table (see Table III-8) with the implicit hypothesis that if the km reading of the vehicle is higher than the supposed life time, then the test should reply 'NotOK'.

Hypothesis:	TRUE	FALSE
LT≤ 16.000km	4	1
LT >16.000km	1	3

Table III-8: Hypothesis table

One way to control the quality of the hypothesis is to compute an indicator called Giny impurity given in (Eq. III-32). The value of this indicator reaches zero when all cases in a node reache a single label.

$$I_G = 1 - \sum_{i=1}^{m} f_i^2$$

(Eq. III-32)

with:

- m: the number of values a variable can take (here m=2 for true or false).
- f_i: the fraction of item labeled with the value i in the dataset.

In the dataset given in example it is possible to compute the Giny impurity for a hypothesis that the component had a life time of 16.000km. The number of occurrences where the hypothesis is true (means the km reading is superior or equal to 16.000km and the test of the component answers 'notOK' AND the number of occurrence where the km reading is inferior to 16.000km with a component's test answer of 'OK') is equal to 3 and 6 when the hypothesis is false. The value is 1-(3/9)²-(6/9)² equal to 0,44 which means that the data are quasi equitably categorized in both labels (a value of 0,5 would mean that the data is divided into both labels equitably). But the information given by this coefficient just concerns the distribution into the labels, it is not an indicator, which allows to confirm or reject a hypothesis. For example if for 2 different datasets of 10 cases concerning the same component, the first one has 100% occurrence for a life time hypothesis of 10.000km and the second 0% occurrence for the same hypothesis then both datasets will have a Giny impurity of zero despite the fact that both hypothesis are completely opposite.

To confirm an hypothesis, the posterior probability for the hypothesis to be true must be at least equal to 90% and the Giny impurity lower than 0.2. The problem of the decision tree is to construct an

optimal decision tree with the best hypothesis and the best classifiers at the root of the tree. This problem is known to be NP-Complete (Non-deterministic Polynomial time). But despite this difficulty a very simple dichotomy procedure can be used here. As a starting point the average value of the km reading of the dataset can be taken and 3 different leaves of the tree can be computed: average value minus standard deviation, average value, average value plus standard deviation. If no leaf is adequate with the previous criteria, then a dichotomy is made between both leaves that deliver the best percentage for the hypothesis equal to true (or if the score are increasing then a new leaf is created with the hypothesis average value plus 2 times the standard deviation or vice versa). This minimizes the finding of an optimal tree with an algorithm of a complexity of $o(n\log(n))$ with n the number of times the dichotomy is applied. The same techniques is applied for the test cost whereby 2 values are learned: the first one for a test reply 'OK', the second one for a test reply 'notOK'. This distinction has to be made, because the time varies depending on test results which are logic, because the test logic is to investigate deeper and deeper all possible failure modes. Moreover, the dichotomy procedure doesn't need to be applied for the learning of the test cost, because the test cost do not vary as widely as the km reading of a vehicle. Nevertheless, to protocol the durability of the tests a timer needs to be started at each execution of the test and paused for example if the test is waiting for a user action (the user can for example be currently working on another vehicle). These solutions permit to avoid that protocol of test costs contains the time for the technician to take his lunch!

3.1.3 Search engine feedback

This section explores the question how to obtain a benefit in terms of precision with the feedback of the perceived symptom search engine. The basic idea relies on the fact that during a diagnosis session, as user enters a request, the engine delivers some perceived symptoms and the user selects the relevant symptoms, and all these information could be sent to a central server. The same notations as in chapter II section 2 are applied here.
One type of feedback that can be exploited for the search engine of perceived symptoms is the popularity, which had been studied in [Manning, Raghavan, 2005]. If a term indexed by i in the base of terms is used in a request and if the user selects the symptom indexed by the variable j (whereby symptom j is supposed to contain the term i) then the popularity of this word is incremented. The computing of the similarity is then obtained by an exponential linearization of both factors: the weight and the normalized popularity. This last step is made in order to balance theoretical symptom discrimination and popularity on the other side.

- **Example :**

Considering the matrix TF (term frequency), df (document frequency) and PO (popularity) and the request term frequency vector given in (Eq. III-33):

$$TFQ = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad TF = \begin{pmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad PO = \begin{pmatrix} 3 & 0 & 3 & 1 \\ 0 & 2 & 2 & 4 \\ 0 & 4 & 0 & 0 \\ 3 & 1 & 1 & 0 \end{pmatrix} \quad df = \begin{pmatrix} 3 \\ 3 \\ 1 \\ 2 \end{pmatrix}$$

(Eq. III-33)

This results in the weight matrixes given in (Eq. III-34) with WQ the request weight vector and We the global weight vector obtained through the formula *tf.idf*.

$$WQ = \begin{pmatrix} \log_{10}\left(\frac{4}{3}\right) \\ 0 \\ \log_{10}(4) \\ \log_{10}(2) \end{pmatrix} \quad We = \begin{pmatrix} \log_{10}\left(\frac{4}{3}\right) & 0 & 2\log_{10}\left(\frac{4}{3}\right) & \log_{10}\left(\frac{4}{3}\right) \\ 0 & \log_{10}\left(\frac{4}{3}\right) & \log_{10}\left(\frac{4}{3}\right) & 2\log_{10}\left(\frac{4}{3}\right) \\ 0 & 2\log_{10}(4) & 0 & 0 \\ \log_{10}(2) & \log_{10}(2) & 0 & 0 \end{pmatrix}$$

(Eq. III-34)

The following similarities are obtained between the request and the symptoms:

$$\begin{aligned} Sim_1 &= 0.476070471 \\ Sim_2 &= 0.954968001 \\ Sim_3 &= 0.163227121 \\ Sim_4 &= 0.08161356 \end{aligned}$$

The following similarities are obtained between the request weight and the popularity weight matrix:

$$\begin{aligned} Sim'_1 &= 0.816496581 \\ Sim'_2 &= 0.629940788 \\ Sim'_3 &= 0.6172134 \\ Sim'_4 &= 0.140028008 \end{aligned}$$

Finally, after an exponential linearization with a factor α equal to 0.5 the global similarity can be computed by:

$$\begin{aligned} SIM(R,d_1) = \alpha.Sim_1 + (1-\alpha)Sim'_1 &= 0.646283526 \\ SIM(R,d_2) &= 0.792454395 \\ SIM(R,d_3) &= 0.39022026 \\ SIM(R,d_4) &= 0.110820784 \end{aligned}$$

(Eq. III-35)

Taking into account the similarity of the documents'increase of diminish the similarity depending on the number of received feedback and the terms (or words), which are used commonly. Nevertheless, this calculation does not respect the equity principle of the symptoms, which means that for a non null popularity matrix and an α superior to zero, some symptoms had already a greater chance to appear in the list of the results. If a request is used a keyword that appears in two symptoms, and if one of these symptoms was already incremented by the feedbacks then the most popular will have a prior probability of appearance in the results which is the usual similarity multiplied by α plus 1- α. The number of feedbacks does not play a role once a symptom had been updated because the chosen similarity metric based on the cosine formula only takes the angle between two vectors into account and not their length. A demonstration is given in (Eq. III-36) for a request weight vector called *y*, a symptom vector called *x* and a positive number of feedbacks for the symptom *x* noted λ (superior to zero). This demonstration relies on the bilinearity of the standard Euclidian scalar product and the positive scalability of the Euclidian norm with a positive multiplicative factor. The similarity issue from the feedbacks is then limited after the first feedback. This feature is important to respect the equity principle of appearance of two similar symptoms, where the first was updated by 10000 times and the second only 1 time.

$$Sim(y,\lambda x) = \frac{\langle y \mid \lambda x \rangle}{\|y\|\|\lambda x\|} = \frac{\lambda \langle y \mid x \rangle}{|\lambda|\|y\|\|x\|} = \frac{\langle y \mid x \rangle}{\|y\|\|x\|} = Sim(y,x)$$

(Eq. III-36)

If for the previous example the user selects the first symptom, then the terms of the request will directly increment the popularity matrix and more precisely the first column of this matrix. The evolution of the popularity matrix before $PO(t)$ and after $PO(t+1)$ the addition of the feedback as well as the request term frequency vector are given by:

$$TFQ = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad + \quad PO(t) = \begin{pmatrix} 3 & 0 & 3 & 1 \\ 0 & 2 & 2 & 4 \\ 0 & 4 & 0 & 0 \\ 3 & 1 & 1 & 0 \end{pmatrix} \quad \Rightarrow PO(t+1) = \begin{pmatrix} 4 & 0 & 3 & 1 \\ 0 & 2 & 2 & 4 \\ 1 & 4 & 0 & 0 \\ 4 & 1 & 1 & 0 \end{pmatrix}$$

(Eq. III-37)

This idea relies on the principle to confound the popularity of a symptom with the relevance (depending on the key terms). Despite the fact that this method does not respect the principle of equity, it may have a positive impact on the precision of the search engine as reported in the literature. This feedback method can also be adapted to the correspondence graph approach, where the graph adjacent matrix is incremented by the popularity.

3.2 Outlook

3.2.1 Synthesis

The previous sections introduced basic machine learning and data mining methods. The considered parameters that have been involved proved already their importance and their possible gain in terms of performance for the diagnosis in section 3.1.1and 3.1.2. The important values have been identified as being:

- Optimization of the perceived symptom search engine: popularity matrix.
- Test cost, coverage and confidence.
- Life time of component.
- Optimization of the weights coefficients of suspicion rules and of causal links.
- Learning dependencies between components and suspicion rules.

How the algorithm learns the values of the parameters depends on the selected method. For the weights of the symptoms, causal links, suspicion rules, test coverage and test confidence the workshop system computes the Bayesian probabilities. At the end of the diagnosis session based on the vehicle's characteristics the path of the guided fault finding procedure is accessible in memory and if an acute causal link was detected then the vehicle's version, the class of the object (causal link), the source and targeted object's ID (unique ID from the database) are written in the protocol as well as the un-acute ones. Once the protocol is sent to the evaluation server, it has just to compute the posterior probabilities of the object (here a causal link) and validate or reject this new object (e.g. if the weight is lower than 0.0001, then the evaluation server can ignore the object) by writing it in the feedback export file. This can be re-imported in the authoring system or validated by an expert first.

The life-time and test cost are learned with the help of decision trees. The life-time is probably the most problem causing parameter in this section. The protocol must for all tested objects contain the test ID the vehicle's version, the km reading and the test result. The evaluation server could then construct the decision tree or hypothesis table (as Table III-8) with the 3 hypotheses: life time equals

- Average minus standard deviation
- Average value
- Average plus standard deviation

And if the best case doesn't fit, a new leaf is added average plus 2 times the standard deviation (or average minus 2 times the standard deviation). A dichotomy is then realized on the continuous variable of the hypothesis: the km stand, which adds a new leaf to the decision tree until a leaf is created that meets the acceptance criteria (90% of the data sets are conform with the hypothesis and the Giny impurity indicator is under 0.2). The tests costs are easier to learn and follow the same data treatment.

At last the symptom search engine is optimized according to a method taken from the literature in [Manning, Raghavan, 2005], with on one side the consideration of the discrimination power of the stems (the root of the words) which is put in balance with the popularity of the symptoms. These methods change the equity principle of the finding of symptoms, but with a value of α inferior or equal to 0.7 the balance still remains acceptable and the gap between a very popular symptom and a very unpopular cannot exceed 30%. In order to collect this information, the diagnosis search engine must collect the request term frequency vector (and the language), and the selected symptoms by the technician. This could be done by saving only the positions and non-null values of the vectors and the IDs of the selected symptoms. The feedback evaluation server have to create a popularity matrix as described in section 3.1.3 by the addition of the non null values of the request term frequency vector for the column corresponding to the selected symptoms.

Finally, this section presents basic machine learning, discovering methods which can help to reach the objective of increasing the performance of the diagnosis sessions and reducing the maintenance cost. The algorithm used to achieve these results are as seen extremely simple because they are specifically adapted to each variable and needs only basic mathematical operation (computing an average value and a standard deviation in the worst case). The post development costs of the models are reduced and optimized with a sufficient number of protocols using as much as possible the current architecture of SIDIS Enterprise. The next section briefly discusses the choices made for the learning methods.

3.2.2 Discussion

The solutions presented in the previous subsections may appear to be the basics of automatic learning and data-mining methods, but they have powerful advantages in this application. The question why these methods have been selected can be explained by the sequent list of reasons:

- Parameters to be learned: section 3.1.1and 3.1.2 presents shortly the impact of the different parameters related to the diagnosis knowledge and their influences on the guided fault finding procedures. This analysis shows the importance of having accurate test confidence and coverage values, because a wrong test answer causes the orbit of the session to increase dramatically. The second point is the importance of the test cost for the 'heuristic optimization' of the test agenda. At last the importance of an adequate symptom weight was also outlined. These 4 parameters can already have an impact of 21% of the performance on the guided fault finding procedure. In section 2.3, with both presented approaches new parameters have to be learned: the life time of the component and the causal relationships, which can contribute to a gain of 19% of the performance. All these parameters are very specific. It makes no sense to take an abstract learning algorithm, which permits to learn all the values by an adequate configuration. Moreover, an abstract algorithm needs to define metrics to limit the search space, which is likely to be super exponential for example for the suspicion links and causal relationships. The advantage of the Bayesian networks is that they require little data preparation and they are quite simple to understand and interpret. Other techniques often require data normalisation or more rarely dummy variables. Considering the set of parameters to learn or adapt very specific methods achieve better results as abstract algorithms with encoded prior knowledge.

- Data preparation and re-use: This is more likely the highest advantage of the selected methods. The data preparation can be done during the preparation of the feedback protocol. It is just needed to reefer to the object by the class name, the ID, and an already fixed classifier. Once this data is prepared, written in the protocol, it can be sent to the central server as an (encrypted) XML file. The central server has only the task to read the file and increment or add the occurrences of the objects contained in the protocol. This means that absolutely no data interpretation is made, only very basic mathematics which require very few hardware resources (except storage space, depending on the size of the after sale network of the manufacturer). The results of the frequency analysis should be exported as an XML file with

a fully configurable schema. This feature should allow re-importing that feedback data directly in the authoring system with the import tool which is part of SIDIS Enterprise. The simplicity of this solution in terms of interpretation, mathematical operations, implementation and compuational complexity makes it probably one of the best possible learning methods for SIDIS Enterprise's return of experience evaluation. Moreover, the data preparation is directly assured by the workshop system, which avoids manufacturers to invest in a farm of server for the feedback evaluation. The evaluation is only based on the computation of posterior probabilities and the server does not need to interpret the data and thus does not need the knowledge structures of SIDIS Enterprise to be installed on it.

- Finally, it is possible to validate the discovered relationships corresponding to diagnostic knowledge by statistical tests. The evaluation server could be implemented with a student test for example and export the results only if they pass the tests. But these adaptations are specific to each manufacturer, this is why the validation by a human operator increases the reliability of the discovered knowledge. Moreover, the chosen methods are robust; the assumptions cannot violate the physical model because the methods are specifically adapted to each parameter. At last the decision trees and Bayesian networks perform well with large data in a short time (especially decision trees).

The benefit of the new parameters or the adjustment of the weights are already demonstrated in previous simulations about the diagnosis algorithm in section 2. In terms of innovation, the feedback evaluation contains a lot of little incremental innovation distributed into: the protocol module, the service program engine (implementation of a function that permits to measure the time to perform a test without taking into account the waiting time for an user action) and finally the evaluation server, which is probably the most important part. Despite these changes, the benefit of the module as well as the simplicity of the evaluation makes it easily comprehensible which is of major importance for a commercial success.

4 Conclusion

This chapter presents the core work of this thesis. The beginning tackles some intuitive ideas to improve SIDIS Enterprise's diagnosis capabilities without any change. The simulations prove that simple concept taken from the model based diagnostic world and adapted to SIDIS Enterprise can increase the performance of the guided fault finding procedure. This section explores also the dramatic influence of test with a bad weighted confidence or coverage. Some other metrics are also proposed to compute the rank for the objects in the test agenda. And finally the information maximization concept are used in order to ask the technician if a set of selected perceived symptoms are observed within a set of symptoms which is susceptible to speed up the diagnosis or increases the discrimination power between the candidates. All these small improvements lead to diminishing of the average orbit of the guided fault finding procedure of 45%. But in average the guided fault finding procedures is still over 150% higher than the optimal diagnosis path of each procedure.
In order to reach the objectives of this thesis section 2 investigates two different diagnosis approaches with the major difference that this time SIDIS Enterprise's information model needs to be changed. The first one is based on a meta-heuristic and lead to an average performance gain of 9% (without the improvement presented in section 1). The idea behind this approach is to emulate a multi-criteria optimization based on the costs and the diagnostic knowledge (suspicion rules, causal links, precision, etc.). The research of a real multi-criteria optimization based on a mathematic modelling of the algorithm as a dynamic system is an impasse. The second proposed approach is a combination of a traditional expert system and model based diagnostic. A search of the function points of this strategy permits to obtain a gain of performance of 21% (still on the same case base and without the improvement of section 1). This approach delivers better results than the first one and is also more realistic. The reasons lies in the choice of the equation:

- The integration of the life time as a Heaviside function, which is known in the literature to emulate the realistic behaviour of electronic devices.
- The integration of the precision, which benefits to nodes that are deeper in the tree.
- The causal networks which model the dependencies between the components which as described in chapter I are particularly important in modern vehicles.

Despite the fact that this approach needs some changes in the information model and the diagnosis module of SIDIS Enterprise the costs still remain acceptable in comparison with the time saved on each diagnosis session.

Finally, this thesis presents the necessary algorithm to learn and/or optimize the values of the different parameters that are required by the second developed diagnosis engine. Always following the bottom line: minimizing the changes or the innovations, in order to avoid a disruptive technological advancement, which has heavy consequences of the commercial adoption of the final product. The examined methods are selected as function of the objective to limit the cost and the changes and moreover to be easily understood. The intuitive ideas combined with the second diagnosis approach and the feedback evaluation engine leads to a global framework for the car diagnosis as depicted in Figure III-33.

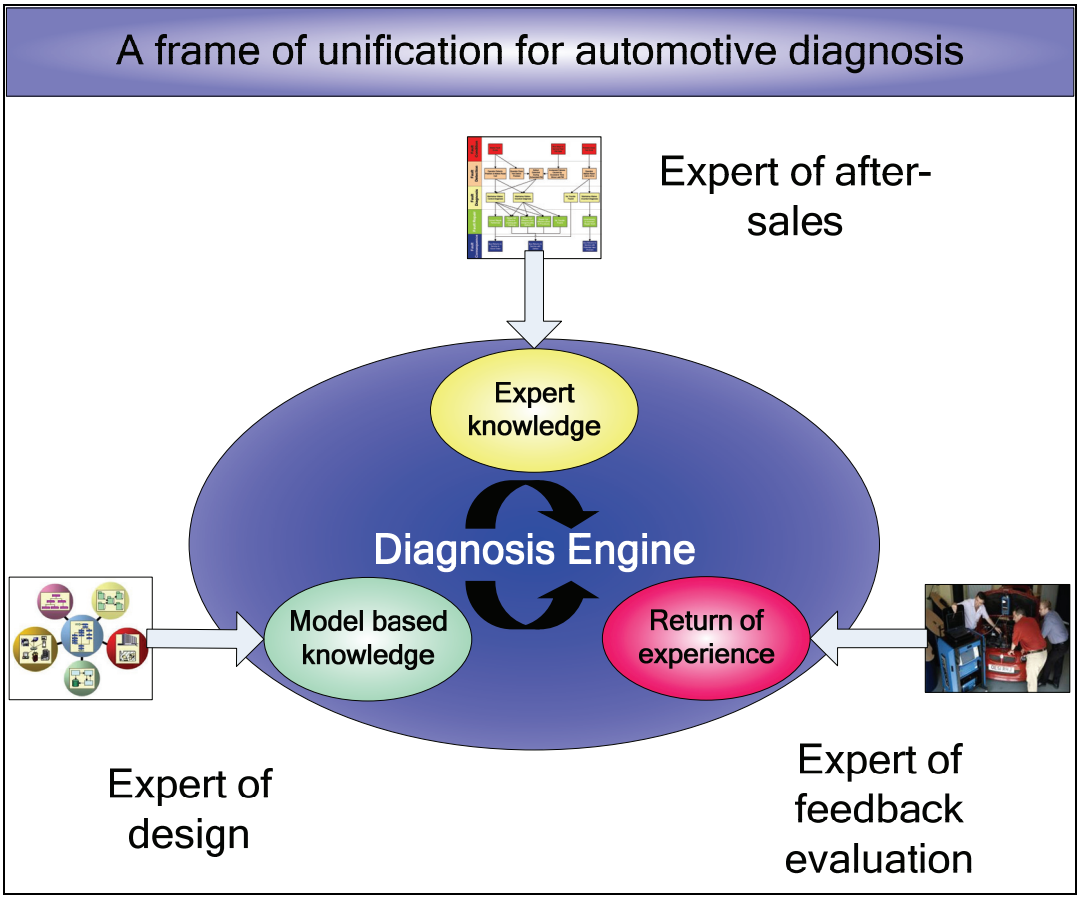

Figure III-33: A frame of unification for automotive diagnosis

The next chapter is concerned with the verification and validation of this global framework with 3 vehicle models of a low, medium and luxury –class car.

Chapter IV: A global framework for the diagnosis

1 Introduction

This chapter examines the four modules presented in the previous chapters. The chapter focuses on the verification of:

- The perceived symptom retrieval engine (see chapter II section 2).
- The ODX file processing engine (see chapter II section 3).
- The combined diagnostic engine (see chapter III section 2.3).
- The feedback evaluation module (see chapter III section 3).

with the support of 3 vehicles models obtained by a car manufacturer. The details about the models of these vehicles are given in Appendix B sections 2, 3 and 4. The vehicles used for the validation and verification of the global framework belong to the low, middle and luxury class and will be denoted vehicle A, B and C. Each business case is described in a separate section (chapter IV section 3.2 to 3.4). The first part of this chapter (section 2) relates briefly to the developed prototype used to run the simulations of the diagnosis session. Then the chapter IV section 3 reports about the business cases for the vehicles A, B, C and concludes with a discussion about the values of the parameters. Due to the confidential character of ODX files only a motivational example of a fictive ECU had been related in chapter IV section 3.1. Nevertheless this ODX case relates the faced difficulties by the import module in a real situation. In some business cases the multiple-fault of breakdowns is handled. Hence, the impact of the feedback is also studied as well as the analysis of the search of perceived symptoms in natural language. As reference values the results provided by the current diagnostic engine of SIDIS Enterprise are given, as well as the meta-heuristic approach (see chapter III section 2.2). For all three vehicles the cases are composed with the help of an expert in order to match as much as possible real situations. The last section (chapter IV section 4) summarizes the obtained results, the gain in terms of performance, the impact of the different parameters and the benefits of this global framework for the after-sale network of a car manufacturer.

2 Developed prototype

The developed prototype in the scope of this dissertation includes the perceived symptom retrieval engine, the diagnostic engine and the feedback evaluation engine. The ODX file processing module is set apart and implemented within SIDIS Enterprise. The reason for these choices are that this module is strongly connected to the authoring system and needs to search in the ECU trees for the comparison of old and new ECU features. The other reason is that during the period of research of this dissertation, SIDIS Enterprise was still under development and the following features are missing:

- Test costs, coverage, uncertainty.
- Life time.
- Causal links.

The more important argument is that at that time, the vehicle's data were not complete and functional. The migration project consisting in the development of an automated import module for the vehicle data was still under development at that time causing the database of SIDIS Enterprise to be quasi empty.

These reasons lead to the externalization of the developed prototype with the advantage of being more flexible. A special modus had been implemented that simulates each step of the diagnosis sessions in a separate graphical user interface, which is particularly well suited for demonstration of the impact of the feedback evaluation engine. Moreover, the developed prototype has flexible import-export mechanisms in XML format or directly into Microsoft Excel, which is more comfortable for the edition of data than in the current version of the AS of SIDIS Enterprise.

The programming language of this prototype was C# as SIDIS Enterprise. This language is platform independent and simplifies the Component Object Model (COM) calls, which were used for the data import and export in Microsoft Excel. All standard field names (e.g. class types, user controls etc.)

used in chapter IV section 2 refer to the implemented classes in the .NET Framework version 3.5. The details about the attributes and classes of the prototype are give in Appendix B section 1.

2.1 Overview

2.1.1 Graphical User Interface

The user interface of the prototype is depicted in Figure IV-1 with a legend explaining all functional areas. The interface contains:

- 3 tree view controls: for the edition of the perceived symptoms, diagnostic objects and ECU fault text. Each tree view has a specific context menu which allows to edit the trees with functions as: add node, add child node or delete node, etc…
- 3 data grid controls: for the edition of suspicion and causal links and one to show the attributes of a selected node.
- 1 list box control: this shows the user the performed actions as for example: file opened, file saved, export to excel completed, etc…
- 1 Menu bar: with all the commands.
- 1 Toolbar: for quick access to the main functionalities like open, save, demonstrator etc…
- 1 Status bar: to show the user information about the current executed process or selected object. A progress bar may appear for time-consuming functionalities.

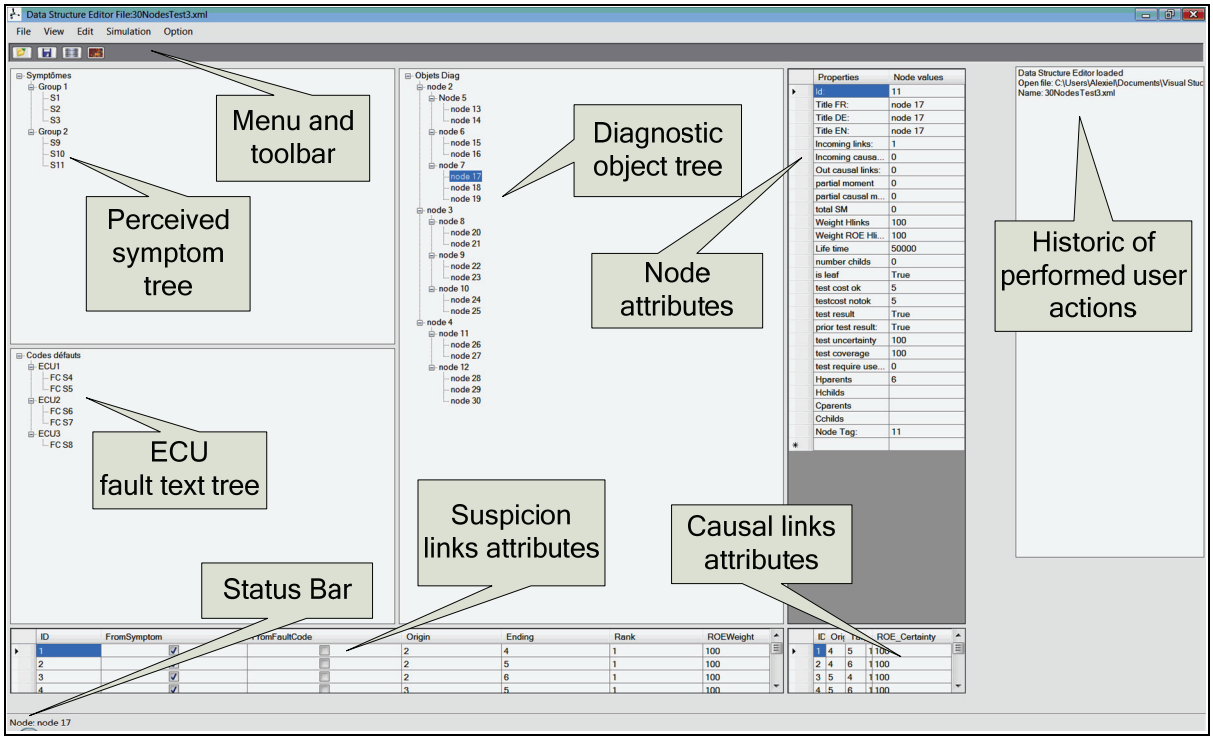

Figure IV-1: Screenshot of the main window of the prototype

Under the file menu the following commands are available: Open, Save, Import, Export, Clear Project and Close. Under the View menu, it is possible to show the correspondence graph and the link between the concept nodes and symptom nodes. It is also possible to show the dataset (cf. chapter IV section 2.2.2) for the language resources. The Edit Menu strip contains the command, which allows to edit the trees, suspicion links, causal links and business cases in separate windows. The simulation Menu contains the commands: load cases, start demonstrator and start automatic simulation. Finally, the language Menu allows to change the current language (between French, German and English), generates the language resources (e.g. BoT, TF matrix, etc…) in the current selected language and generates the language resources for all languages.

2.1.2 Main software components

The main components of the prototypes are depicted in Figure IV-2. The main window (cf. Figure IV-1) contains the class Form1 and the settings of the application. When functions like the editing of trees or rules are executed then special corresponding forms are instantiated. The diagnostic demonstrator is also a windows form, which allows the simulation of a diagnosis session in the guided fault finding modus. For the design of this module some special user controls are created as the SymptomControl, DiagNode, etc…

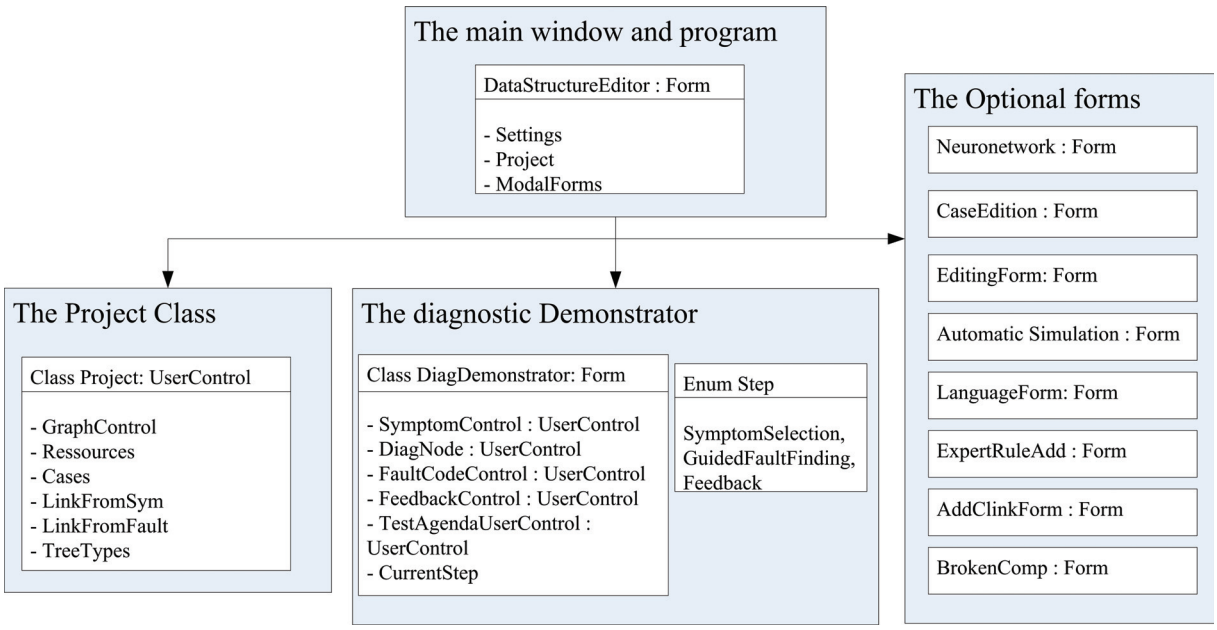

Figure IV-2: Main Software components and associations

The last component is the project class which contains all the vehicle data, user, controls, and the different engines listed in chapter IV section 1. The next sections detail this component because it is the core of the diagnosis engine. The other components are not further described, because they constitute only optional features which are not entering in the scope of chapter IV.

2.2 Presentation of the main component of the developed prototype

This section details the main components and workflow in the Project class. Four essentials points are commented: the datasets, the perceived symptom search engine, the diagnostic engine and the feedback engine which constitutes the main part of the global framework for the automotive diagnosis. Chapter IV section 2.3 concludes this presentation with a summary and a discussion of the differences between this prototype and its feature implementation in SIDIS Enterprise.

2.2.1 Overview

The project class inherits the standard User Control class because it contains different controls. Moreover, the class inherits some standard methods fired when some events occurred like *Click*, *DoubleClick*, *Scroll*, etc. and it inherits some important properties like the list of Parent or Child Controls, Parent Form, etc. The principal fields of the class are given in Table B-1. The contained user controls are: 3 tree views for the perceived symptoms, fault code text and diagnostic object tree and 3 data grids for the suspicion rules, causal links and the last for one is an information provider which details the attributes of a selected node in a tree view. The last important component is a dataset, which contains collections of tables where each column represents a particular variable (cf. chapter IV section 2.2.2). The Project class contains also a context menu, which appears when a right click on a

tree view control occurs. Finally, some general fields as the current language, path and file name, etc. are also defined and detailed in Figure IV-3.

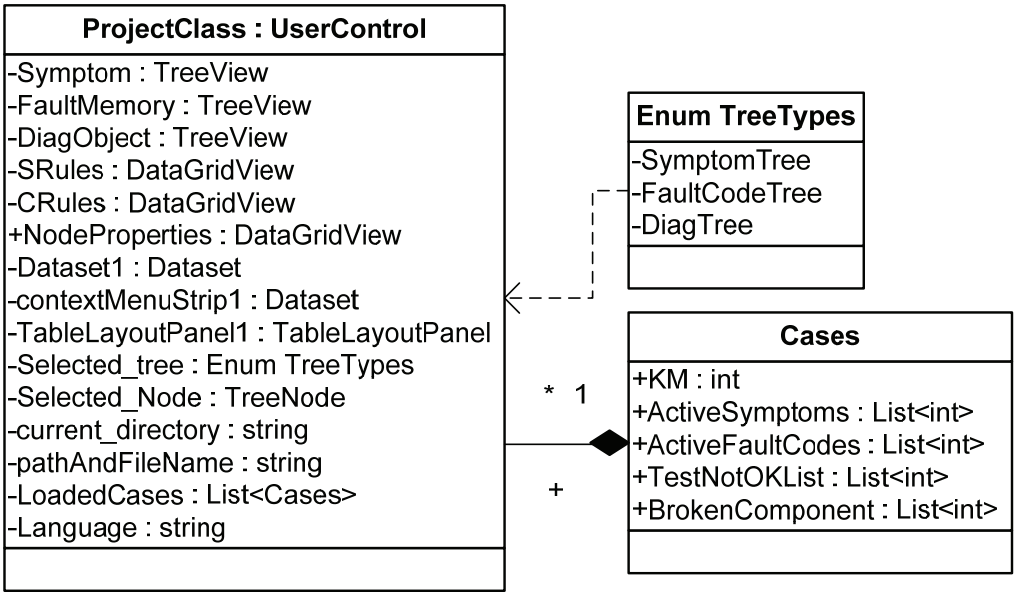

Figure IV-3: Major fields in the project class and associations

One special field is the list of business cases called *LoadedCases*. Each element of this list contains a case for the simulation of a guided fault finding procedure. A case is defined by a name and the *km* reading of the vehicle. Furthermore, the class contains the following four lists:

- ***ActiveSymptoms***: a list of integers containing the identifiers of the entire active perceived symptom in that case.
- ***ActiveFaultCodes***: a list of integers containing the identifiers of all active fault code in that case.
- ***TestNotOkList***: a list of integers containing all the identifiers of the tests that answers notOK
- ***BrokenComponent***: a list of integers containing the entire identifiers of the broken components

The class contains an empty constructor and methods to read from or write into an XML file.

2.2.2 The Dataset

As mentioned previously the *dataset1* contains a collection of 8 tables with all the knowledge structures and vehicle data. The first and second tables are related to the perceived symptoms and the fault codes. The third table concerns the diagnostic objects. The data tables 4 and 5 describe the suspicion and causal links. Finally, the data tables 6, 7 and 8 are related to the language resources.

2.2.2.1 Perceived symptoms and fault code data tables

The data tables describing the perceived symptoms and the fautl codes have the same structures (fields and attributes) defined and commented in Figure IV-4.

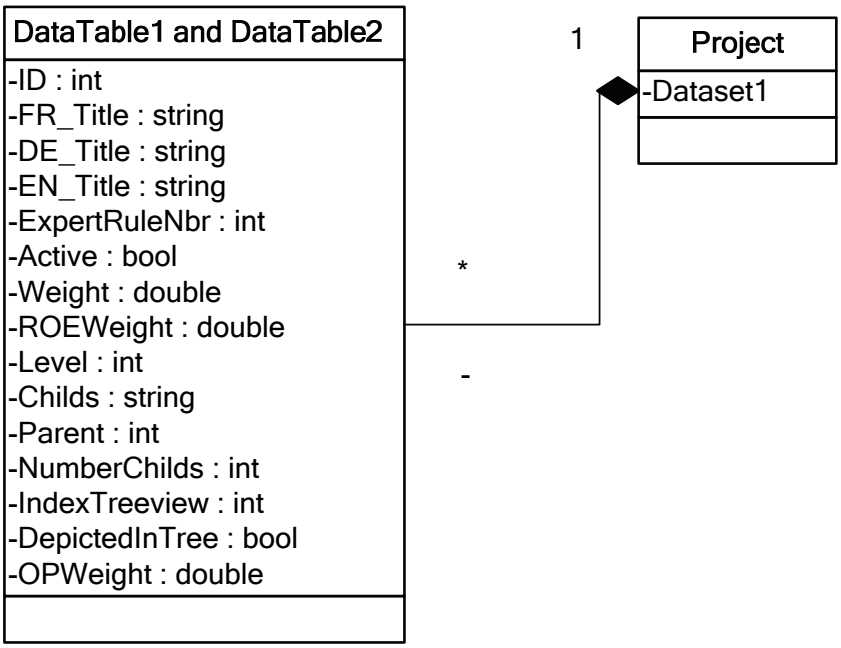

Figure IV-4: Data columns of first and second data tables from *dataset1*

Despite the normal fields like identifier, title and the fields related to the position in the tree, there are two important columns: the weight obtained by the feedback engine and the operational weight. The reason is that depending of the chosen symptom weighting method; the computed weight is calculated and written in the column OPWeight. The diagnosis engine just takes this column into account for the candidate generation.

2.2.2.2 The diagnostic object data table

The third table concerns the diagnostic objects and their associated tests, given in Figure IV-5.

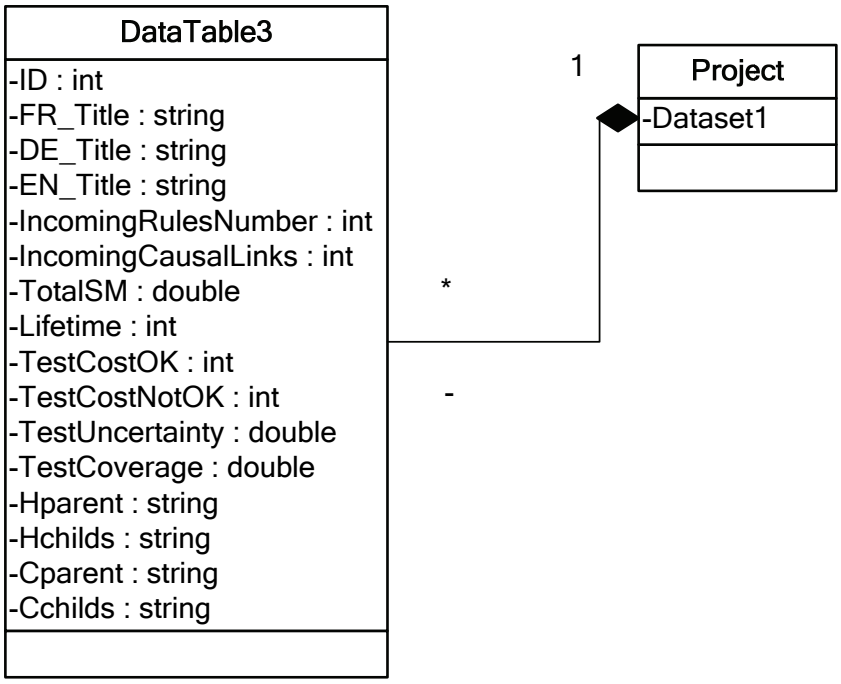

Figure IV-5: Data columns of third data table from *dataset1*

The diagnostic object table contains similar columns as for the perceived symptoms and the fault code (e.g. titles and positions in the tree). The difference is that each node in the table contains the weight of the hierarchy link to its parent node and the identifiers of the causal ancestors/successors nodes as well as fields for the suspicious moments, which are used by the diagnostic engine. The information of each test associated to a component is also defined from the column indexed by 15 to 21. The prior test result column stands for the a priori test answer during a simulation and the test result is the posterior answer given by the test (the difference here is specific to a special mode of the simulator which introduces a randomized uncertainty to the test answer, but this mode is suppressed in the release version of the prototype).

2.2.2.3 The data tables for the links

The data table n°4 and n°5 contain the definition of the suspicion links and the causal links. Their description is given in Figure IV-6.

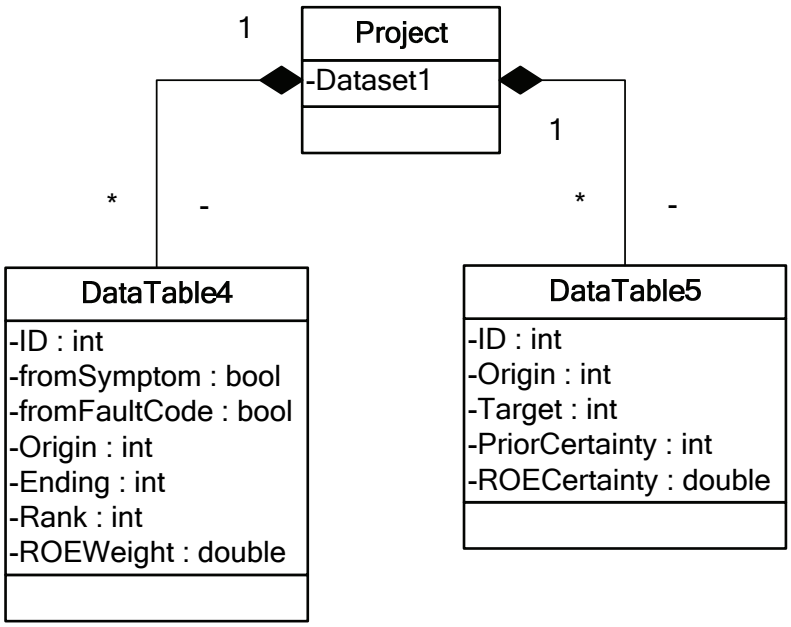

Figure IV-6: Data columns of suspicion and causal links data tables from *dataset1*

Both links have a normal weight or rank and a second one with the prefix “ROE” for return of experience. The suspicion links have two boolean columns to precise if the link is starting from a perceived symptom or a fault code, then in row indexed by 3 the ID of the targeted diagnostic object is defined.

2.2.2.4 The data tables for the language resources

The last data tables are related to the language resources of the perceived symptoms for the retrieval engine and given in Figure IV-15 for all fields related to the engine reported in chapter II section 2.2 without stemming. For the values with stemming the same column types are defined except that they have the prefix “STE_” for the title.

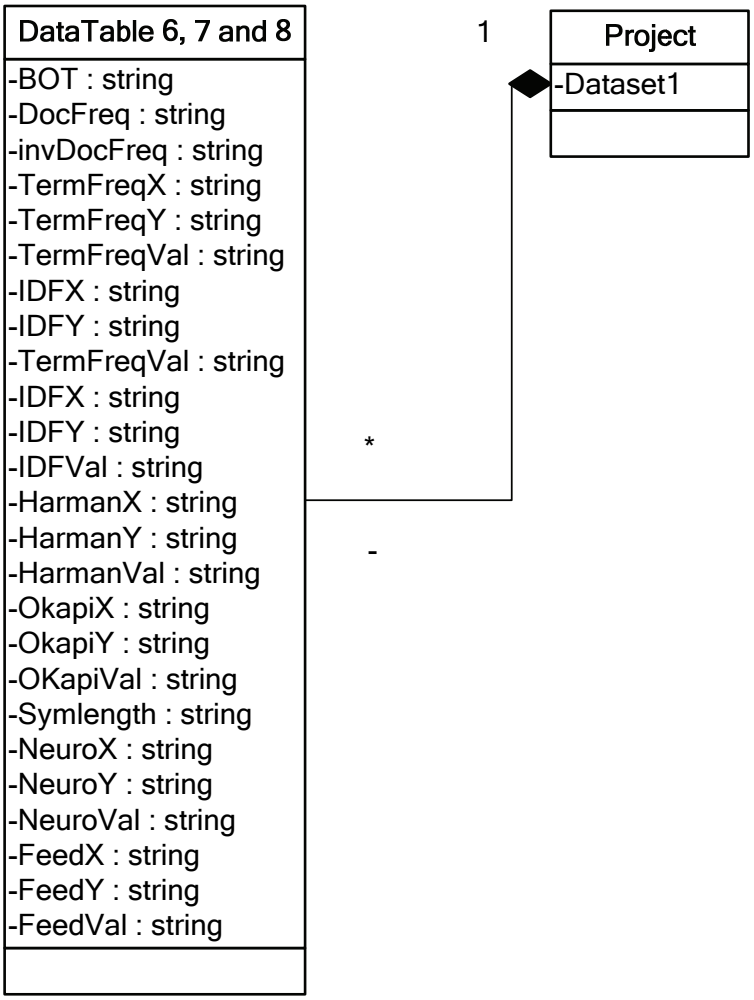

Figure IV-7: Extract of data columns of the language data table from *dataset1*

All the information for the language retrieval engine is present in the tables. For performance and space reasons only non null values of the matrices are saved in the tables. For the fields of the Croft weight matrix only values that are different of the Croft constant are saved. This data table can be filled separately for each language or for all languages but the process for the construction of the base of terms, filtering, term frequency counting, etc… are critical in terms of performance. Thus, once the values are calculated, they are saved within the project file to avoid the re-execution of these time consuming methods.

2.2.2.5 Synthesis

The choice of using a data set with a collection of tables rather than a local data base like Oracle rely on the simplicity of use and reliability. The data set can be saved and refilled with a standardized XML writer or reader. Would the prototype use a database system, then the standalone modus could not be run anymore. Only PC with a dedicated pre-installed database server could be used for running the prototype. Moreover, there is no need of the programming of a query adapter for requests upon the database. Finally the data tables contain a search functionality with different types of parameters, which is very time efficient e.g. for the search of a diagnostic object by its ID. For the tables 1 to 5 the column called ID is set as the primary key and auto-increments its value for each new entry. The search by the key of a row can be directly called by a standard functionality, which is already implemented in .NET. Furthermore, the data tables are very flexible, if new data needs to be modeled, the data tables just needs to be extended by the corresponding column without any changes in the

basic function as open or save to XML file. For more convenience of the data edition in the prototype it is possible to export the 5 first data tables which correspond to the diagnostic knowledge into Microsoft Excel Worksheets and re-import them into the prototype.

2.2.3 The perceived symptom search engine

The perceived symptom search engine is implemented in the project class in order to have access to the data tables. The main fields used by the engine are specified in Figure IV-8, whereby all matrices have equivalent fields with the prefix "STE_" which means they are constructed with a stemmed base of terms.

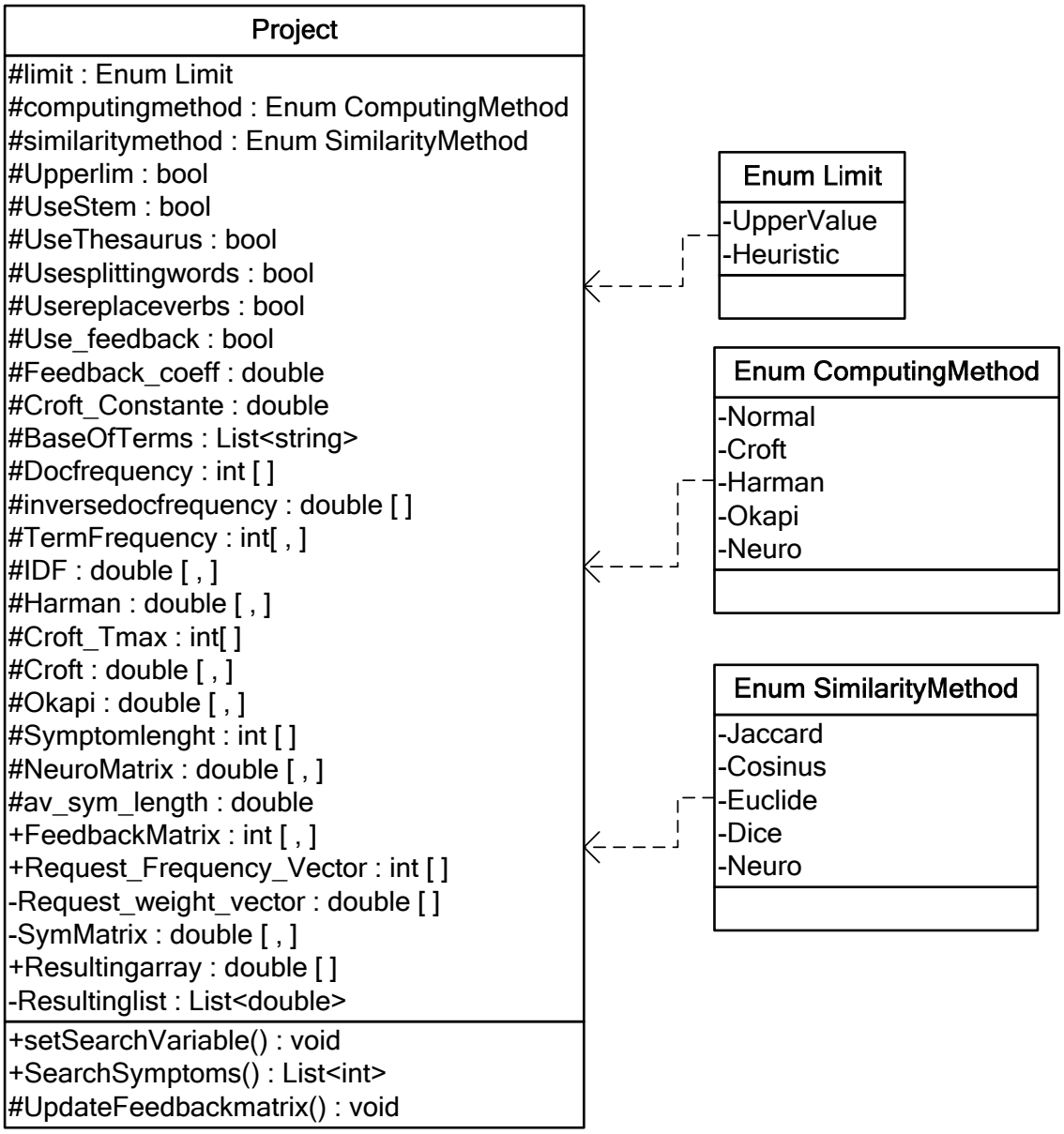

Figure IV-8: Main fields and methods of the perceived symptom retrieval engine

The tree principal methods of the engine are used to set all the optional parameters (method *setSearchVariable*), search the symptom (method *SearchSymptoms*) and update the popularity matrix (*UpdateFeedbackmatrix*). The simplified workflow of the search engine is depicted in the centered area of Figure IV-9 with the information exchange between each process step and the fields or data tables.

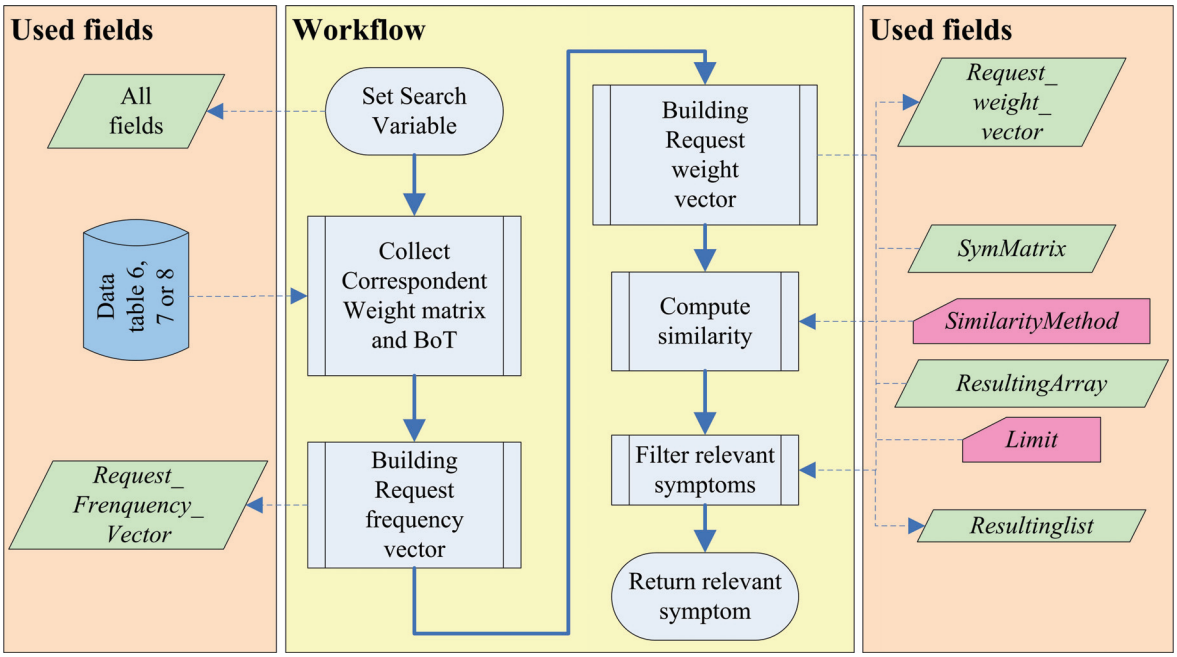

Figure IV-9: Workflow of the search engine

The first step of the algorithm consists in fixing the value of all the optional parameters via the method *setSearchVariable*. Then, depending on the selected weight formula, the correspondent weight matrix (e.g. Okapi, Croft, Harman) is built by the reading of the language data tables. The term frequency vector is built with the base of term as reference (stemmed or not). The next step consists in the construction of the request weight vector with the corresponding weighting method (e.g. *tf.idf*, Harman, Croft, Okapi). The similarity between the request weight vector and the symptom weight matrix is finally computed depending on the selected formula (e.g. Dice, Jaccard, Euclide, cosinus, etc…). At last, the relevant symptoms are sorted depending on the selected limit type. If the simple acceptability threshold is selected, then all symptoms with a similarity score over the upper bound are returned. If the heuristic metric is selected then variable acceptability threshold (described in chapter II section 2.3.2) is applied for the filtering of the relevant symptoms. Finally, a list of integers containing the IDs of the relevant symptoms is returned. One encountered problem during the test phase with the engine is an out of memory exception, which is fired, if too much searches are performed with different weighting formulas (e.g. *tf.idf*, Harman, Okapi, Croft and with the correspondence graph) causing all the matrix fields being set to matrixes of dimensions 2500x700. To avoid this problem, the unused fields are re-initialized in order to free memory resources enabling a research with other parameters. The main search methods are defined with the access modifiers public, because this module is only used in the demonstrator mode (and not in the automatic simulation mode), which needs to have access to these methods.
For the specific feedback evaluation of this engine, the variable *Feedback_coeff* is set to the value of the parameter α defined in (Eq. III-25) (cf. chapter III section 3.1.3). If the feedback is used, the term frequency vector is added to the columns of the popularity matrix corresponding to the definitive selected symptoms via the *UpdateFeedbackmatrix* procedure.

2.2.4 The diagnostic engine

The fields defined and used by the diagnostic engines are listed in Figure IV-10. The important variables are the different lists of IDs of the tests performed during the diagnosis session like the OKTested or NotOKTested lists. The performance indicators like the orbit are also declared as general variables.

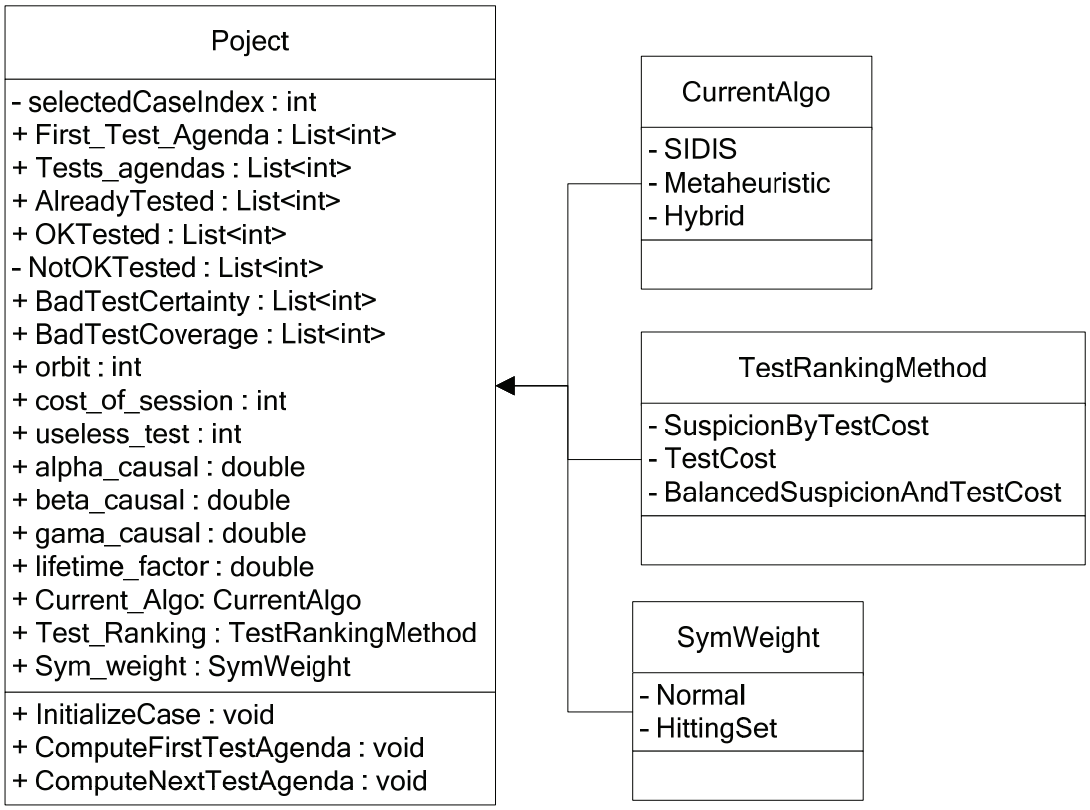

Figure IV-10: Main fields and methods of the diagnostic engine

Three custom enumerations (index by 18, 19 and 20 in Table B-7) allow to switch between the implemented diagnosis algorithms, the test ranking methods, the symptom and fault code weight calculation. The workflow of the guided fault finding procedure is depicted in the center of Figure IV-11 as well as the main used fields, enumerations and data tables. Here, the 5 first data tables are used, which corresponds to the perceived symptoms, fault codes, diagnostic objects, suspicion rules and causal links. The first step of the simulation of a guided fault finding procedure consists in fixing the values of all the fields. A special window allows the user to select all the variables for the simulation (e.g. the diagnosis algorithm to use, the symptom weighting method to use, etc.). The graphical user interface of this window contains combo boxes and shows the user the corresponding formulas in picture boxes. Once the user clicks on the acceptance button a global check of the values of the parameters is made (a message box appears if wrong values are detected). Then the information of the selected case is read and in particular the prior test result is set to the corresponding value in the data table 3 (for the diagnostic objects). The weight of all active symptoms and fault codes is set or computed (e.g. if the hitting set method is selected for the weight calculation, cf. chapter III section 1.2.1). The suspicious moment is then calculated depending on the selected algorithm used for the simulation. The choices are: SIDIS Enterprise's algorithm (cf. chapter III section 1.1.1), the meta-heuristic approach (cf. chapter III section 2.2) and the hybrid heuristic and model based strategy (cf. chapter III section 2.3). Then, the test agenda is computed depending on the ranking metric. The implemented metric are: the suspicion moment, the balanced suspicion moment with ratio maximal test cost by the test costs, see (Eq. III-9) and the usual metric from SIDIS Enteprise's algorithm, see (Eq. III-2). After this step, the tests are sorted by their rank. For the test cost the arithmetic average value between the OK answer and notOK answer is taken. In the automatic simulation mode the user can select if in each test agenda it is automatically the first ranked test that is excecuted or the second or the third, etc… In the demonstrator mode, the rank of the test selected by the user is returned to the *ComputeNextTestAgenda* method. Naturally if in the automatic simulation mode, the user has selected to perform always the test in the 7^{th} position of the agendas and if the agenda contains less than 7 tests then the last one is performed in order to avoid unexcpected exceptions. In the case that the test agenda contains no elements and if the broken components have not been discovered, the child elements of the already tested elements are ranked (by the selected test ranking metric) in a new test agenda. This

solution is used in order to find out tests that have a wrong coverage and/or confidence value. Finally, if the list of tests answering notOK correspond to the list of broken components defined in the class case, then the diagnosis session is finished, else the suspicious moment calculation is performed again (depending of the obtained test result). For security reason an additional termination criteria is implemented in the test and refine loop (between the calculation of two test agendas). The criteria is if the list of tested components during the diagnosis session equals the list of all suspected objects of the first calculation of the suspicious moment **and** all their childs then the diagnosis session is finished. This auxiliary termination criterion provides the security that the simulator does not perform an infinite loop during the test and refine cycles.

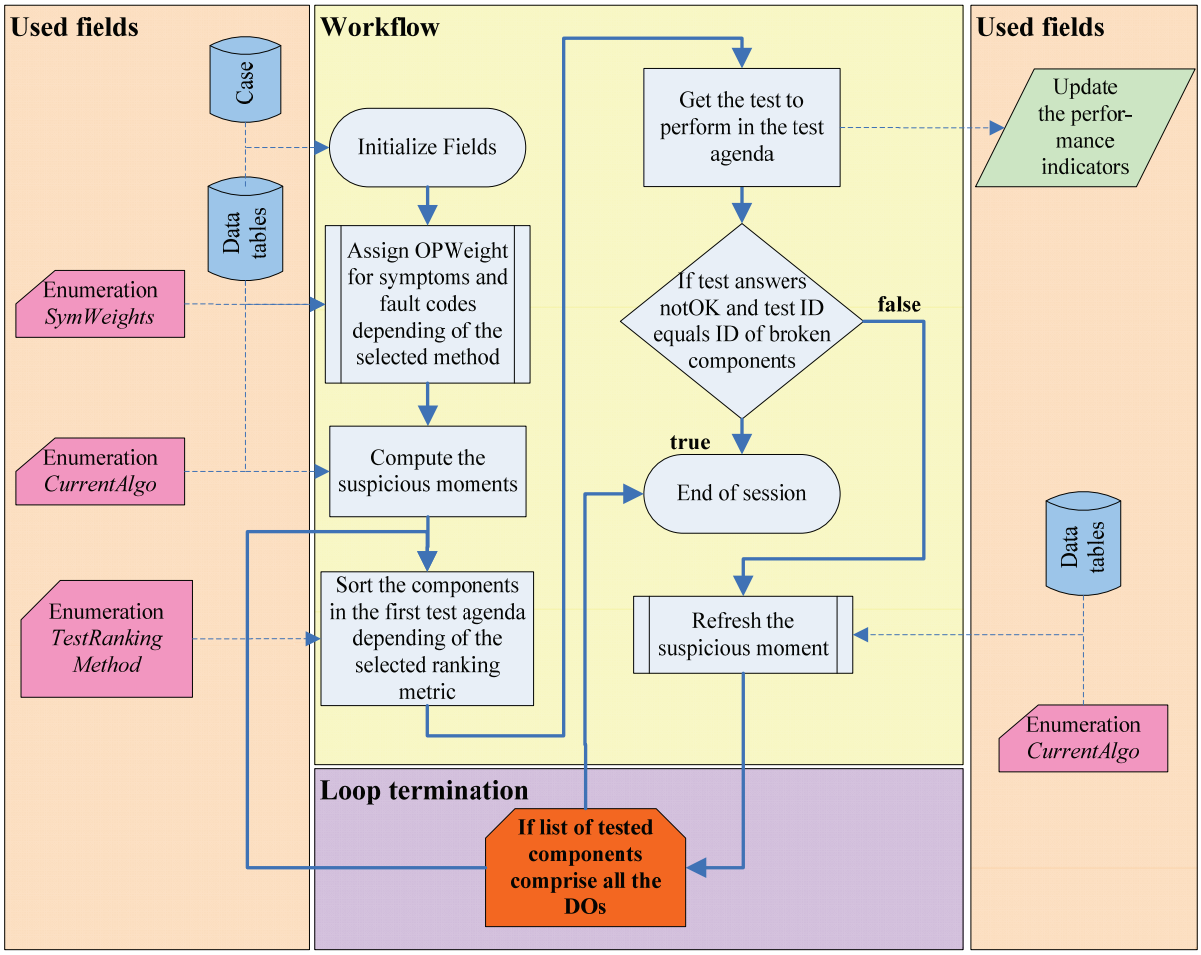

Figure IV-11: Workflow of the guided fault finding procedure

It is to notice that the automatic simulation of guided diagnostic session can be run through the whole list of cases loaded in the project class. This option is specially developed to see the impact of the evaluation of the feedback engine. Moreover, at the end of the diagnosis session a global check is performed over the tested objects and if a configuration is found with a node answering OK and a child node of the node answering notOK, the list *BadTestCoverage* is immediately updated as well as the confidence.

2.2.5 The feedback engine

Figure IV-12 contains the main fields and methods of the feedback engine. The important parameter is specially the *NumberOfIterations* before the new weight parameters, suspicion links, etc… are evaluated and stored in the data tables.

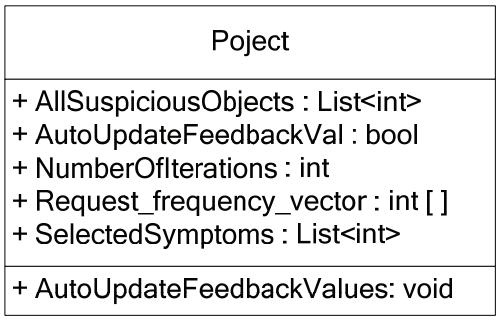

Figure IV-12: Main fields and methods of the feedback engine

The workflow of the AutoUpdateFeedbackvalues method is depicted in Figure IV-13 as well as the information exchange between the different fields, tables and each step of the procedure. Once a diagnosis session is finished all the important information are stored in a temporary data table. For example with the values adjusted by the Bayesian networks, it is the occurrences OK and notOK that are stored for the suspicion rules. For the parameters learned with the decision trees as the lifetime for example, it is the component's ID, the km reading of the case and the test answer.

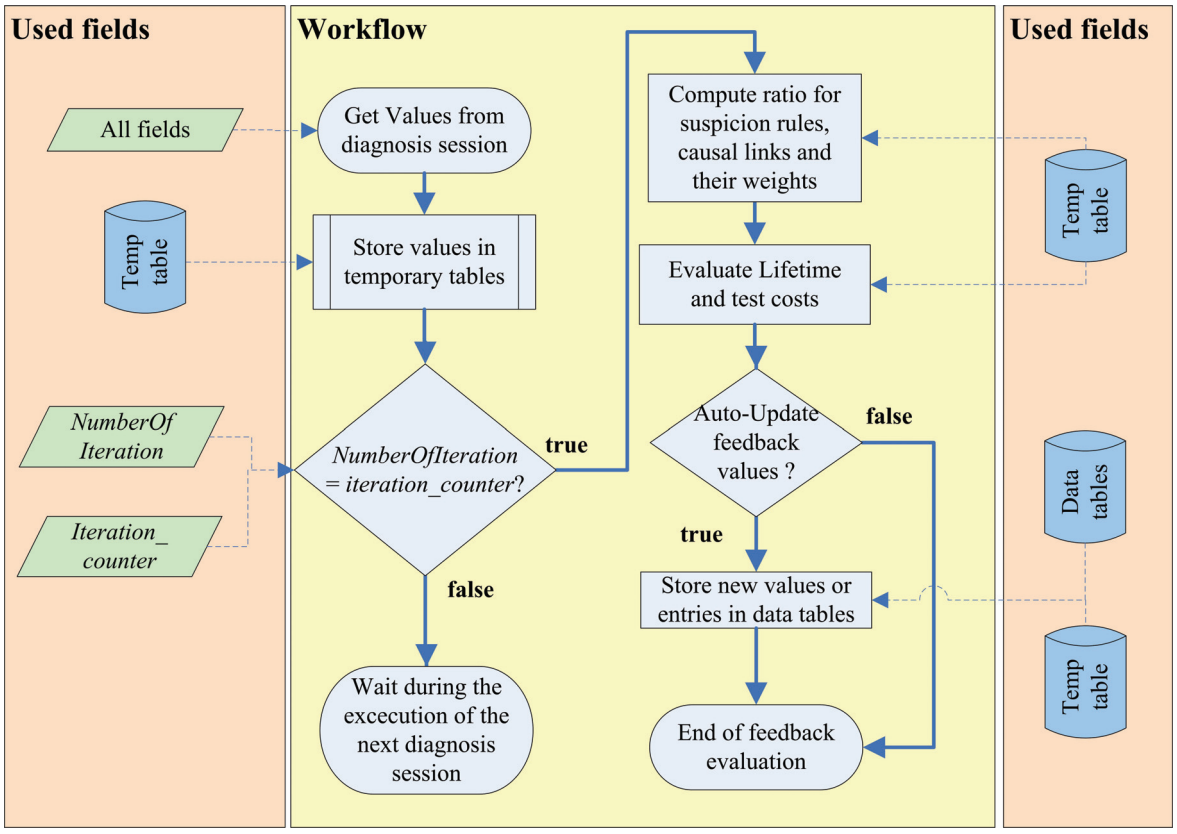

Figure IV-13: Workflow of the feedback engine

It is to notice that the feedback engine updates the popularity matrix of the perceived symptom search engine only when the demonstrator mode is run. The reason is that this specific feedback step needs on one side the retrieved symptoms by the search engine and on the other side the subset of definitive selected symptoms by the technician for the initialization of the diagnostic engine. Thus, only a human operator can make the delta in the relevance judgment, which is minimized through the performed feedback engine, which explains why this feature is only relevant and implemented in the demonstrator mode.

2.3 Discussion about the implementation in SIDIS Enterprise

As previously mentioned the developed prototype includes the perceived symptom retrieval engine, the diagnostic engine and the feedback evaluation engine. The features of this prototype were not directly implemented into SIDIS Enterprise, because the product was still under development during the dissertation and the vehicle's data as well as some specific weight coefficient were not complete and functional or implemented. Moreover, an automatic simulation mode of a guided fault finding procedure would have been very difficult to implement into SIDIS Enterprise and an external simulator that emulates the vehicles behavior (specific ECU answers during the testing) would have been necessary. These reasons have lead to the externalization of the prototype for the simulation of the guided fault finding procedures. Furthermore, the software provides far more flexibility under these conditions, allowing for example to change the data structures and the comparison of results obtained with different algorithms (of the perceived symptom search engine and the diagnostic engine).
The next subsections discuss the details and recommendation for the implementation of the feature of the prototype into SIDIS Enterprise.

2.3.1 Perceived symptom search engine

For the implementation of the features provided by this prototype into SIDIS Enterprise with the lowest cost and reasonable time performance, it is to remember that the architecture depicted in Figure III-30 is considered as the best solution. The publication process of the authoring system must include the following steps: the creation of the base of terms, the creation of the weight and inverse document frequency matrix (and eventually a dictionary for the equivalence between symptom IDs and column index of the matrix), the compilation of the linguistic resource files (the resources used in the prototype are given in Appendix A section 2.1). All these steps should be compiled in language specific files and distributed to the WSs. For space saving, it has already been mentioned that the weight matrix should be sent in a format where only the positions and coefficient of non null values. Chapter II section 2.3.2 already reports that 97.2% of the coefficients of the weight matrix are null, they can consequently be called sparse matrix. Thus sending only the non null values reduces the size of the update consequently. One practical method to implement this search engine in SIDIS Enterprise would be to create a dedicated functional module in SIDIS Enterprise, which is then integrated in the standard guided fault finding procedure. This would allow a practical configuration of the module (e.g. if a manufacturer wants to implement its specific similarity function) in the AS by a software expert and in the WS, the functional module can access its language resources through the resource manager and send the important data for the feedback engine to the protocol engine of SIDIS Enterprise (cf. chapter I section 2.6.3). A last consideration has to be taken into account for this module: in the cases that the perceived symptoms are dependent on the vehicle variant, only the retrieved symptoms that are relevant for the handled vehicle variants have to be proposed to the technician.

2.3.2 Diagnostic engine and preparation of the protocol

The implementation of the diagnostic engine relying on the description given in chapter III section 2.3 can be established very simply. The causal links need to be implemented in the information model, but due to the presence of the general *Link* class, a simple inheritance with the fields: origin and targeted node (as type diagnostic objects), the weight (as type double) and the vehicle variant need to be specified. There are no particular difficulties for the changes that have to be implemented in the current diagnostic algorithm, because the structure is still the same except that the causal moment and lifetime has to be evaluated according to (Eq. III-26) and (Eq. III-29).
In order to benefit fully of the performance improvements of the feedback engine, the following data needs to be written in the protocols which are sent to the central server (cf. the architecture schema depicted in Figure III-30):

- The km reading of the vehicle.
- The term frequency vector of the request (only coordinates and position of non null values to save space in the protocol and time in the data transmission).

- The IDs of the selected symptoms by the technician: in order to increment the popularity of the relevant symptoms by a human relevance judgment.
- The occurrences of suspicion rules (and causal links) targeting tests with OK answers and notOK answers.
- The same as previous for the occurrence of acute and not acute symptoms or fault codes.
- The broken component or Least Replaceable Unit (LRU).
- All the performed test IDs, answers and costs.
- The list of tests with an identified wrong coverage or confidence factor.

The vehicle variant must be specified for all these mentioned objects. One problem for the measurement of the tests costs is that some tests require a user action and the technician can for example take a break at this moment. But these specific test, which requires user actions, uses graphical forms, which requires another specific service of SIDIS Enterprise WS. Thus, thanks to the call of these services the time of the tests can be measured by a timer and paused or restarted whenever a graphical user interface is created or closed. Finally, at the end of the test the measured time by the timer can be returned to the protocol engine.

The protocol structure plays no central role, but for the time-optimization of the feedback evaluation engine a similar structure as presented in Figure IV-14 for a given vehicle variant is strongly recommended. Moreover, the constitution of a specific library as well as the compression, decompression algorithm in the WS and on the central server could limit the time of the data transfer for relative minor implementation cost.

```
<?xml version="1.0" encoding="utf-8"?>
<FeedbackData>
<!--Section km reading -->
<KMvalue>55000</KMvalue>
<!--Section SymptomSearchEngine-->
 <PerceivedSymptomSearchEngine>
 <Language>en-EN</Language>
 <BaseOfTerm version="1.2.3"></BaseOfTerm>
 <Request>Engine does not start</Request>
 <RequestFreqVector PositionX1="15" CoordinateValue1="1"
PositionX2="112" CoordinateValue2="1"></RequestFreqVector>
  <SelectedSymptoms>
   <ID>16</ID>
   <ID>17</ID>
   <ID>112</ID>
  </SelectedSymptoms>
 </PerceivedSymptomSearchEngine>
 <!--Section Diagnostic Sections-->
 <DiagSession>
  <SuspicionRules>
   <Rule ID="115" TargetAnswer="OK"></Rule>
   <Rule ID="116" TargetAnswer="OK"></Rule>
   <Rule ID="117" TargetAnswer="notOK"></Rule>
   <Rule ID="118" TargetAnswer="notOK"></Rule>
  </SuspicionRules>
  <CausalLinks>
   <Clink ID="112" TargetAnswer="NotOK"></Clink>
  </CausalLinks>
  <NewLinks>
   <SuspicionRules>
    <Rule FromSymptom="TRUE" FromID="2237" TargetID="6678"></Rule>
   </SuspicionRules>
   <CausalLink>
    <Clink From="6678" To="2256"></Clink>
   </CausalLink>
  </NewLinks>
  <!--Section test cost-->
  <TestCost>
  <Test ID="1223" Answer="OK" Time="5"></Test>
  <Test ID="6548" Answer="NotOK" Time="7"></Test>
  </TestCost>
  <!--Section broken Component-->
  <BrokenComponent ID="6678"></BrokenComponent>
  <!--Secction test confidence and coverage-->
  <ListBadTestCoverage>1223</ListBadTestCoverage>
  <ListBadTestConfidence></ListBadTestConfidence>
 </DiagSession>
</FeedbackData>
```

Figure IV-14: Recommended diagnosis session protocol structure

The protocol structure depicted in Figure IV-14 presents several advantages. One of them is that the data can immediately be stored with an XML reader into a data table (or data base depending of the size of the after sale network of the car manufacturer and the excepted number of protocols received by day). It has to be remembered that the preparation of the data is done at the end of the diagnosis session because this process would take too much time on the feedback evaluation server. Moreover,

at the end of the diagnostic session, the data can be interpreted, because the diagnostic knowledge structure has been filtered to match the handled vehicle variant. The data interpretation consists especially in what is called acute and in-acute links for the suspicion rules and causal links. If for example the configuration depicted in Figure IV-15 occurrs without causal links during a diagnosis session:

- Symptom S1 and S2 active at the beginning of the session.
- Then the node n2 is ranked first in the agenda then tested and answers notOK.
- Then the node n7 is tested and answers OK.
- Then node n6 is tested and answers OK.
- Then node 5 is tested and answers notOK.
- Then node n14 is tested and finally is detected as the broken component.

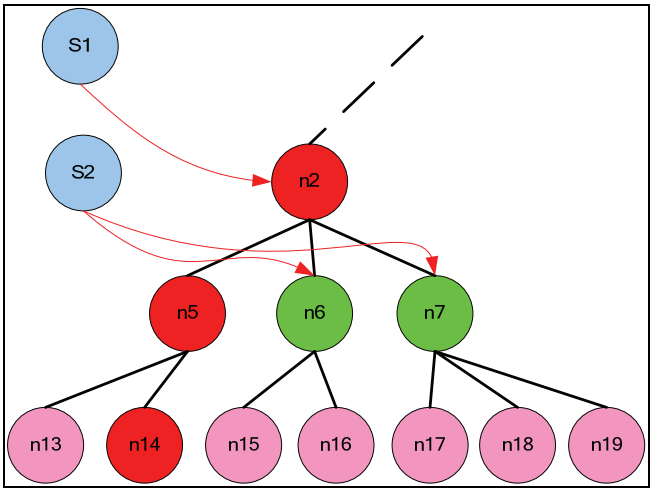

Figure IV-15: Example for the data preparation

The interpretation made by the diagnostic engine on this example in the evaluation of the acute and in-acute links relies on the shortest path that leads to the broken component. Here, the link from S1 to node n2 will be considered as acute because the link targets an object that is on that path. The links originating from S2 are considered as in-acute because they lead to the execution of useless tests. For the discovering of new suspicion rules, due to the fact that the naïve variant of the Bayesian network is taken (cf. chapter III section 3.1.1) the links inexistent in the model are considered:

- Suspicion link from S1 to n5 as acute
- Suspicion link from S1 to n6 in-acute
- Suspicion link from S1 to n7 in-acute
- Suspicion link from S1 to n13 in-acute
- Suspicion link from S1 to n14 acute

The same applies for the discovering of causal links, where the cause is identified as the component modeled by node n14. Therefore, the interpretation engine will consider all possible causal links between the diagnostic objects and all the suspicious objects from the first calculation of the suspicious moment. The causal link from n14 to n6 and n7 in this example will be considered as acute and all other as in-acute.

2.3.3 The feedback evaluation server

The feedback evaluation server has to compute the ratios from the frequencies of occurrences of all the the protocols and re-inject them into the authoring system. In order to avoid the creation of links with very low weights, the feedback evaluation has also to filter the data. The proposed filter criteria should be configurable in the case that a car manufacturer wants to use his threshold values and/or specific statistical indicators. The Figure IV-16 shows the recommended process of the implementation of the feedback feature (with an example for causal links) for SIDIS Enterprise, which minimizes the costs.

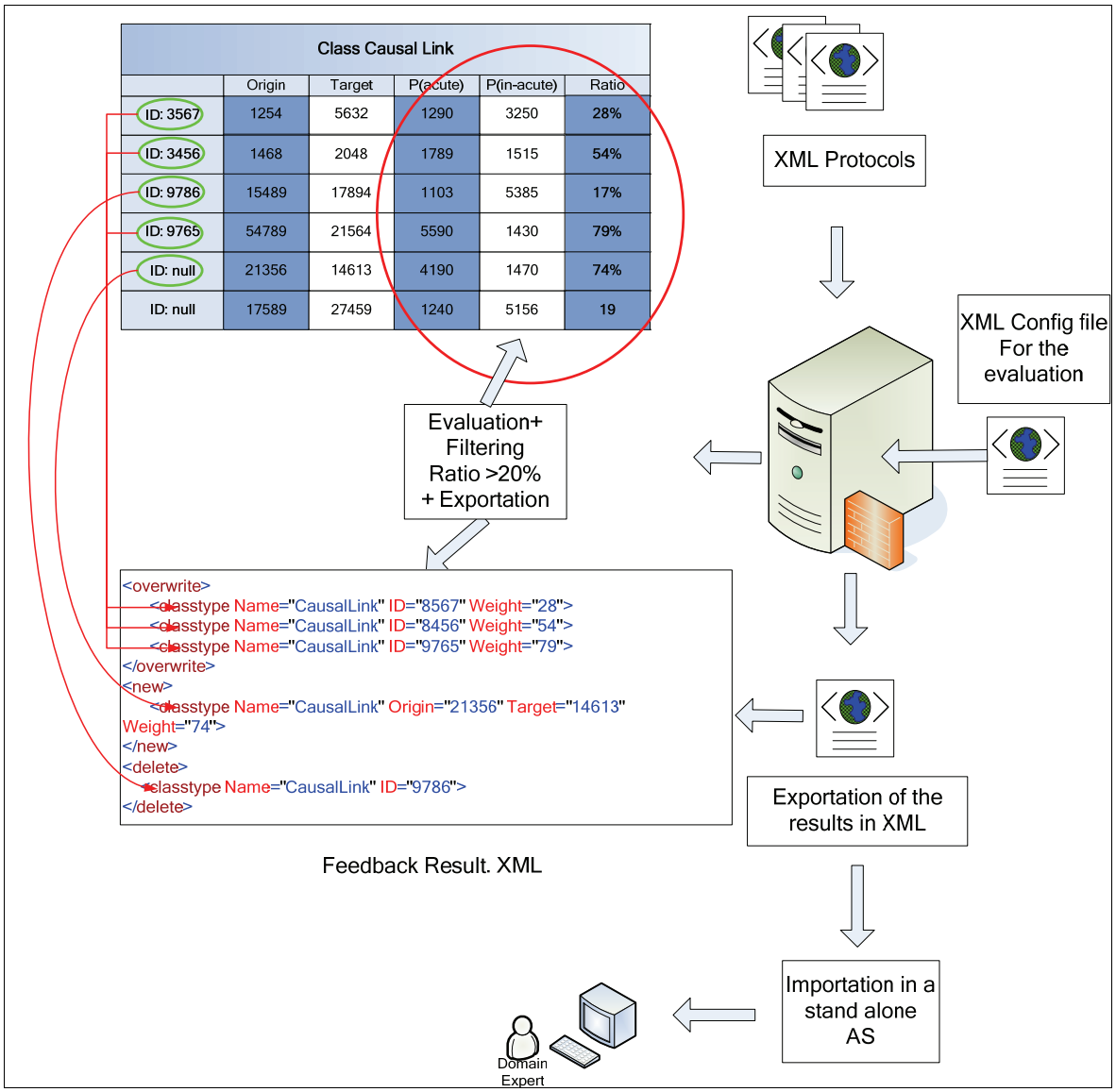

Figure IV-16: Example for the data preparation

It is to notice that existing links already have an ID whereby new links have no ID as depicted in the frequency table on Figure IV-16. For a threshold ratio of 20%, the three first links see their weight updated. Thus they are exported in the section "overwrite" of the Feedback Result file. The link with the ID 9786 had a ratio that is under the threshold value, causing the export engine to write it in the "delete section". The next to last entry in the table contains a new link with a ratio of 74%, which means that it is written in the section "new" of the result file. Finally, the last entry of the table contains a non existent link but with a ratio inferior to the threshold value causing him to be ignored.
This architecture allows realizing only a simple evaluation of the frequencies and basic mathematical operations for the calculation of the ratio on the feedback server. This minimizes the space and hardware requirements of the feedback server, which is an important argument for a car manufacturer. The structure of the resulting feedback file with the tree sections: overwrite, new and delete is also recommended for the reason that it is possible to use the data import tool of SIDIS Enterprise for the injection of the results of the feedback engine into SIDIS Enterprise's AS. The XML configuration file of the feedback server should contain the exact XML schema for the structure of the result file (which is compatible with SIDIS Enterprise's import module) and also at least the criteria of chapter III section 3.1.1and 3.1.2 (ratio criteria, Giny impurity factor and correct hypothesis percentage). An extendable algorithm should be developed for this engine in the case that some car manufacturers prefer to use their own statistical acceptance metrics.

The last step after the creation of the result file consists in the importation of the new objects into the database of the AS. It is recommended to execute this import on a stand alone AS server, where the new values, objects are validated and eventually tested by an expert.

3 Use Cases

3.1 An ODX File import

3.1.1 Presentation of the motivational example

In this section, a simple example, which illustrates the capabilities of the developed ODX import system is presented. This example bases on the import of the description of a fictive new ECU called CDC_1DIN in a database where the description of the fictive ECU XHUAGW is already linked with the related functional information. In our test base, a series of nine DIAG-COMM objects are to be interpreted and five fault codes are to be put into relationship with their functional counterpart.
In this experimental case some objects belonging to the new ECU are strictly similar to those of the existing one. However, in order to demonstrate the intelligent capabilities of the system and to simulate a real situation, some data have been slightly modified. The example in Figure IV-17 illustrates that the functional job read sensor list is going to be executed by the DiagComm *JobGetSensorList* when communicating with the ECU XHUAGW.

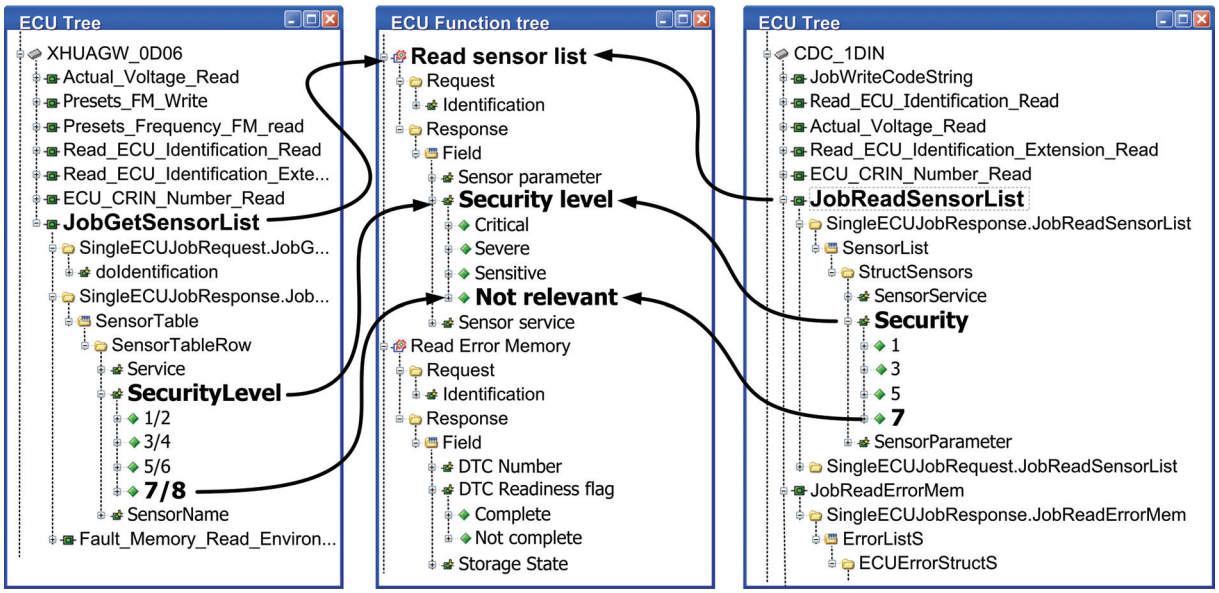

Figure IV-17: The tree extracts from the AS with the ODX description of the ECUs

The response is made of an ODX table and a row containing the three parameters *service*, *securityLevel* and *sensorName*. When analyzing the new ECU CDC_1DIN, our multi-agent system suggested that the unknown DiagComm JobReadSensorList should correspond to the function Read sensor list and that its unknown parameter Security might correspond to the functional parameter Security Level. Even the returned values are interpreted in a functional way, so that the signification of a result 7/8 is automatically found when reading the database. Note, that the value 7 returned by the other ECU will have exactly the same signification on the functional level. Once these suggestions have been accepted by the author, the description of the new ECU can be imported in the AS and so complete the case base. Once it is done, a functional communication with the ECU CDC_1DIN can be started using the functional description and the ECU-specific information for the vehicle communication. The completed case base is also available to the multi-agent system for further ECU imports.

3.1.2 Data processing by the agent FFMatcher

This section describes the processing results of the multi-agent system for the Fixed Function *ReadSensorList*. Figure IV-18 describes the output of the analysis of the Fixed Function, as done by the agent FFMatcher. The parent node of the XML-file is the Fixed Function's description. It is identified through its CMS-ID (identifier in the content management system of SIDIS Enterprise), which is the unique object ID within the AS (zone A in Figure IV-18). The second attribute gives the Title of the node in order to present the result to the author and no description for the FaultCode attribute is present in the node as we deal with a FixedFunction and not with a Fault Code.

```
<AnalysisResults>
 <FixedFunction CMSID="310173067" FFTitle="Read sensor list" FaultCode="">
  <FFParameter CMSID="311203467" ParamTitle="Security level">
   <NewODXDiagComm ID="LT3_JobGetRoutineList" ShortName="JobGetRoutineList">
    <ASDiagComm CMSID="201" DiagCommGlobalMatching ="59,8518518518519" >
    </ASDiagComm>
   </NewODXDiagComm>
   <NewODXDiagComm ID="LT3_JobReadSensorList" ShortName="JobReadSensorList">
        ...
   </NewODXDiagComm>
   <NewODXDiagComm ID="LT3_JobGetActorList" ShortName="JobGetActorList">
        ...
   </NewODXDiagComm>
  </FFParameter>
  <FFParameter CMSID="310364555" ParamTitle="Sensor parameter">
        ...
  </FFParameter>
  <FFParameter CMSID="311206027" ParamTitle="Sensor service">
        ...
  </FFParameter>
  <FFParameter CMSID="310055691" ParamTitle="Identification">
        ...
  </FFParameter>
 </FixedFunction>
</AnalysisResults>
```

A

B

C

Figure IV-18: Extract from the analysis results of the Fixed Function "Read sensor list"

The child nodes are the FF Parameter nodes which are stored in the AS. Here, the nodes SecurityLevel, SensorParameter, SensorService and Identification are found, which are the four functional parameters the Fixed Function is made of (zone in Figure IV-18). Using this XML representation makes it possible to look for the suitable ODX-DiagComm objects for each FFParameter separately. This implies that several ODX-DiagComms can be necessary to execute one Fixed Function and makes it necessary to look for the more suitable ODX-DiagComm for each FF-Parameter separately, whereby the grouping of parameters belonging to a same ODX-DiagComm is being made further on by the agent FFL-Matcher. Figure IV-18 also shows the detail for the analysis of the FF-Parameter *SecurityLevel*, which could a priori be sent back by one of the three ODX-DiagComms *JobGetRoutineList*, *JobReadSensorList* or *JobGetActorList*, because they are the closest to the Fixed Function *ReadSensorList* in the sense defined in (Eq. II-17), which is computed by considering following DiagComm features:

- DiagComm identification data:
  - Short Name.
  - Long Name.
  - OID.
- Associated functional data:
  - Referenced Functional Classes:
    - Short Name.
    - Long Name.

- ID.
- Diagnostic class.

- Request and response structure:
  - Various parameter types.
  - Various field types.
  - Multiplexed message.
  - Env Data objects.

The comparison result which is obtained is stored in the protocol file for the most interesting results in the node *DiagCommGlobalMatching*. At the same time, the CMSID of DiagComm from the case base, which leads to this identification is stored in order not to use unnecessary further resources for the ODX parameter identification by the agent FF Parameter Matcher.

3.1.3 Parameter and value matching

The agent FFParameter Matcher puts the functional parameters in relationship with their best ODX-specific counterparts outgoing from the work of the agent FFMatcher presented in Figure IV-18.

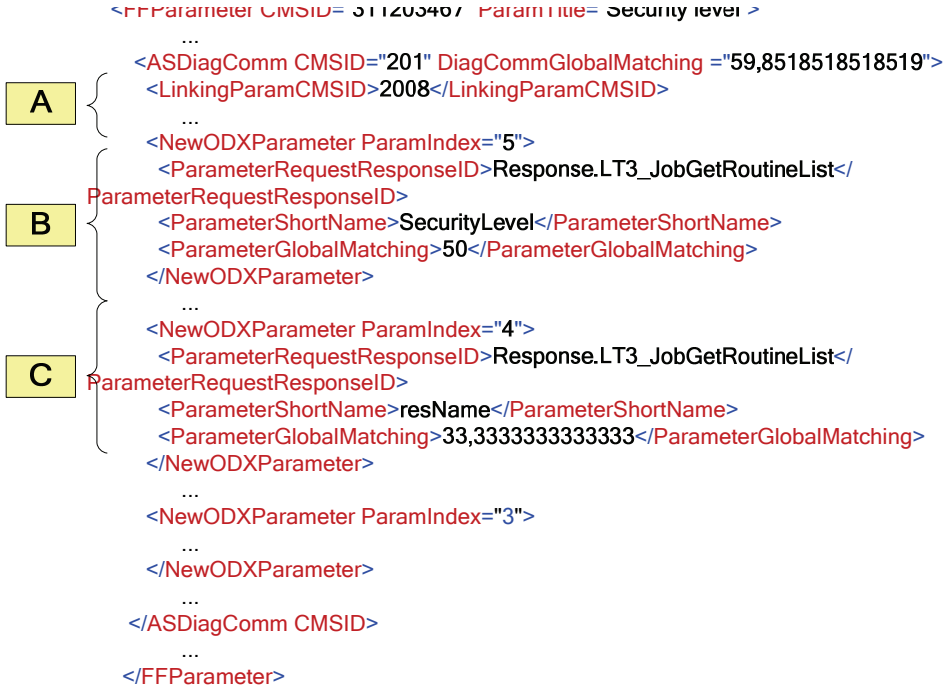

Figure IV-19: Extract from the analysis results of the functional parameter SecurityLevel

Outgoing from the Fixed Function's CMS ID, the FF Parameter and the suspected DiagComm from the case base, the agent FFParameter Matcher makes requests to the agents AS data miner and ODX data miner in order to get further details for the parameter analysis. The role of the agent AS data miner is to follow the relation links and to retrieve the relevant cases from the case base and the agent ODX DataMiner retrieves the information to be compared from the ODX file, which contain:

- Parameter identification data:
  - Short-Name.
  - Long Name.
- Associated functional data:
  - Semantic.

Once the information is gathered together, a request is sent to the agent StringComparator to evaluate the nearness of the considered parameters (e.g. zone B and C in Figure IV-19). Further on, it is very important to make sure that cases that do not make sense are avoided in order not to face an unsolvable problem at the value linking step.

Figure IV-19 represents the analysis results, which indicate the key elements for the best matches as well as a ParameterGlobalMatching indicator, which represents the global matching between the

functional parameter and the new ODX parameter. As one ODX-DiagComm object can be put into relationship with several responses, it is important to store the response, in which the interested ODX parameter has been found and to remember its short name in order to make it possible for the author to compare the relevancy of the results obtained.
The data processing from the agent value matcher is somehow more complicated, because it has to take the value linking type between the functional parameter and the ODX parameter into account to determine the data processing algorithm. We are faced with three different cases:

- Having **values from type EqualsValue on both ODX and functional side** implies that the algorithm has to return one to one relationships between functional values and ODX-specific values. In that case, a correspondence matrix is created and the best match in the whole matrix is selected for the first value link proposal. The corresponding row and column are not considered anymore for the further links since there is a bijection between the functional and the ODX-specific values in this specific case. The selection of the parameters is then repeated until either all the functional parameters or all the ODX-specific parameters have their pendant.
- Dealing with a **value from type AllValues on the functional side** implies that all the values on the ODX specific side should be linked to it. In accordance with the filtering mechanism of the agent FF-ParameterMatcher, this case should happen only in a request with values from type AllValues on the ODX side or in responses.
- Dealing with a **value from type AllValues on the ODX side** implies that all functional values should be linked to it. In accordance with the filtering mechanism of the agent FF-ParameterMatcher, this case should happen only in requests or in responses with a functional value from type AllValues.

As the parameter linking algorithm used by the agent FF-Parameter Matcher ensures that only compatible parameters are proposed, the agent ValueMatcher should not face any particular linking problem for the values. The result which is obtained is saved as a series of links between functional values and ODX values and their matching rate.

3.1.4 Agents FFL Matcher and Global supervisor

The agents FFLMatcher and GlobalSupervisor play the central role to functionalize the ODX-file as they respectively select the best matches found by the agents described until now and point the search in the optimal direction.
As the agent FFMatcher does not provide a definitive functionalization proposal but the most interesting alternatives, the first task of the agent FFLMatcher is to determine the best solution to get the counterpart from a fixed function. As no bijection is supposed in the general case, the agent can have to associate one fixed function with several DiagComms. To do so, it takes several elements into account

- The computation of a global match per couple (DiagComm, FFParameter)
- The maximum number of DiagComms linked to a fixed function in the case base
- A global relevancy of each DiagComm evaluated by the agent FFMatcher

By evaluating the different possibilities and picking the best out, a unique proposal for the ODX pendant of a fixed function can be made. If necessary, FFLMatcher makes further requests to competent agents in order to refine or complete the analysis. Once the final decision is taken concerning each object, the information is transmitted to the agent global supervisor which determines the best search strategy.
Building a coherent search strategy demands that the agent global supervisor takes two elements into consideration. The first one is that the computing resources (time) is limited and the second is the accuracy of the results. In order to increase the efficiency in the given time, the agent can decide for each analysis step if he wants to explore new possibilities (diversification strategy) or if he wants to deepen the search in a direction he considers promising. In the first case, a fixed function will be selected randomly and evaluated whereas in the second case, the fixed function will choose according to the ranking established among the ECUs, followed by the ranking established between the fixed function Lists.

Figure IV-20: The final XML analysis report

To model the limited resources available for the agent, it is given a predefined amount of energy, which is depends on the average number of fixed functions in the FFLs considered. This energy is used to make the search progress and the agent takes this evolution into account to determine the search strategy: the more energy remains the higher is the probability to use a diversification strategy. The global search process can be divided into three parts:

- **Initialization**: a random FixedFunction is analyzed for each ECU considered
- **Iteration**: FixedFunctions are analyzed until the energy is consumed, according to the chosen exploration strategy
- **Finalization**: in order to make a coherent reporting, the two most promising ECUs are analyzed completely

The analysis results are presented in Figure IV-20 and can be imported in the AS using the import module to obtain the links presented in Figure IV-17 between the new ECU and the function tree.

3.1.5 Synthesis

This motivational example of two fictive ECUs shows the step performed by the automatic import module of ODX files for a new ECU called CDC_1DIN. The first step consists in analyzing the knowledge already entered in SIDIS Enterprise's database meaning here the ECU XHUAGW and its links with the related functional information (Read Sensor List). Then, the matching score of the parameters and value are computed and sent to the FFLMatcher and the GlobalSupervisor which filter the best matches. With this module an author can save around 45 minutes for each ODX file. But to be more effective a user-friendly interface of the module should be designed where for each proposed link the similarity score is written as well as the origin of the link and the targeted node in the function tree. The actions button must at least comport the function accept and reject the proposed link. The possible performance gain of this module with the hypothesis that an expert takes two hours to import manually an ODX file (as in the current situation) allows to save 1.25 working hours for a vehicle with 80 ECUs. This feature can allow to reach an economy of 37.5% of the costs to fill the ECU knowledge structures in the database for a vehicle with 80 ECUs. Thus, the realized economy is very significant considering the fact that the trends of more innovation in electronics are integrated in modern vehicles.

3.2 Vehicle A: single and multiple fault cases

Vehicle A belongs to the low class of cars. It was built in 2001 and the model encompasses more than 500 diagnostic objects nodes, 2000 perceived symptoms, 500 fault codes and 300 expert rules. Only the relevant details about the model and the case are given in Appendix B section 2. The 3 cases taken as example are quite usual for a low class vehicle. The first one is like a single fault case as the second one, but this one contains a test that delivers a wrong test answer. The third contains a multiple fault case were subcomponents of the ABS and ESP do not work. It must be outlined that the original of this vehicle contains only 25 perceived symptoms, but the base was filled with data from another manufacturer in order to test the perceived search engine within a more critical environment (over 2000 symptoms). The main focus on this study concerns the search of perceived symptoms and the multiple fault case (which is implemented as two sub-cases). Despite the fact that the feedback engine is not strongly stressed over these cases, the details of the creation of new links will be explained.

3.2.1 Search of symptoms

The detailed results of the search of the symptoms in the three cases are given in Table IV-1, where both limit types were tested (the simple acceptance limit fixed to a threshold value of 30% and the heuristic variable acceptability limit described in chapter II section 2.3.2) as well as the *tf.idf* (column M1), Harman (column M2), Okapi (column M3) weight formulas and the correspondence graph search method (column M4).

		Number of returned symptoms											
		French				**German**				**English**			
Request	**Limit type**	M1	M2	M3	M4	M1	M2	M3	M4	M1	M2	M3	M4
Engine temperature too high (4 relevant symptoms)	Upper Bound	31	64	45	34	42	55	52	14	35	62	60	84
	Heuristic	2	3	3	6	6	12	9	11	4	6	4	8
Black emissions when engine starts (4 relevant symptoms)	Upper Bound	32	47	38	45	35	64	52	58	35	65	53	60
	Heuristic	4	8	8	12	6	8	8	12	4	8	6	10
Brake lights do not work (1 relevant symptom)	Upper Bound	34	45	42	62	32	68	64	42	35	65	53	60
	Heuristic	0	4	4	12	2	4	4	8	3	9	9	24
Average precision (heuristic acceptance limit)		**67**	**50**	**50**	**36**	**61**	**36**	**40**	**27**	**78**	**43**	**59**	**31**

Table IV-1: Results of the search of perceived symptoms

Due to the number of very similar expressed symptoms, the results provided by the search engine with a fixed threshold value as acceptance limit are too numerous. For example for the first request a lot of symptoms containing the word "temperature" alike: "indicator of water engine temperature", "indicator of exterior temperature", "indicator of oil temperature" are returned. The impact of the heuristic (stepwise variable) acceptability limit is clearly positive, except that in some cases the filtering is too strong and no symptom is returned. The *tf.idf* weight formula provides the best matches in terms of precision with an average value of 69%, which validates the conclusion of chapter II section 2.3.2. The correspondence graph shows deceiving results in these cases with an average precision of 31%, but this can be explained by the high number of concepts to handle, which creates a dispersion effect. Moreover, the 2000 concept nodes of the first layer have been translated by a machine in this experiences but the weight of the arcs between the two layers are kept to their original value. The dispersion effect is more critical if only a keyword is used as a request causing the graph to return the quasi totality of symptoms containing that keyword (e.g. for request with the word engine,

over 100 symptoms are returned alike "water temperature in engine…", "cooling related to the engine…").
With the hypothesis that an inexperienced user wants to search the perceive symptom, it can take him up to 2 minutes to find the relevant symptoms. With the engine based on the *tf.idf* weight formula and the heuristic acceptability limit, this time is reduced to: 10 seconds for the request, 1 and half a second for the retrieval, then around 15 seconds (worst case) to deselect retrieved symptoms that are not relevant. The time economy is therefore around 77% for a search with this module. This average result of 1.6 minutes economized for each session does not take into account the potential gain in terms of performance for the diagnostic because the number of relevant link issues from the retrieved symptoms must be taken into account (but this is evaluated in the next section).
The case "Brake lights do not work" is especially interesting for the impact of the feedback engine. The similarity between the request and the excepted symptom is equal to 0.29 with an empty feedback matrix. If the user takes the time to search the symptom in the tree and to select it, the term frequency vector of the request is added to the column of the feedback matrix corresponding to the relevant symptom. The evolution of the similarity of the relevant symptom in dependence of the number of feedbacks with a linearization factor of 0.5 is depicted in Figure IV-21.

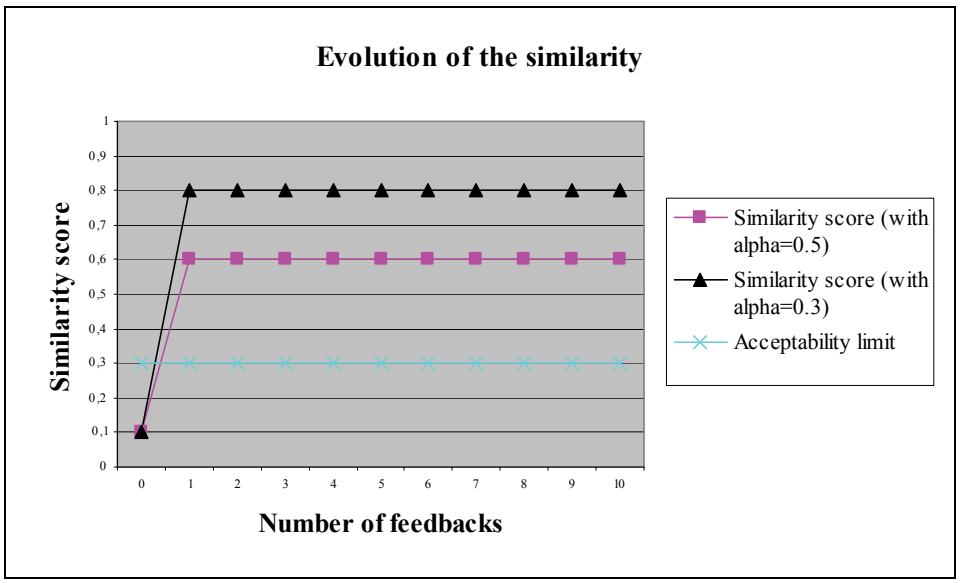

Figure IV-21 : Influence of feedbacks for the perceived symptom search engine

As Figure IV-21 shows, for zero feedbacks the similarity between the request and the symptom is lower than the acceptance limit and lower than in the normal case. The cause is that the similarity between the usual weight matrix and the request weight vector is multiplied by the coefficient α, which is lower than 1 and the result is added to the similarity between the weight vectors of the request and the weighted popularity matrix, which is equal to zero. Actually without any feedbacks the entire coefficients of the popularity matrix are equal to zero. But once the relevant symptoms have been selected, the popularity matrix is updated, and if the same search is performed the obtained similarity score increases by $(1-\alpha)$. This is explained by the fact that the weight vector from the popularity matrix matches exactly the request weight vector causing the cosines between the two vectors to be equal to 1. This case illustrates the positive impact of the popularity matrix for the perceived search engine. Another aspect is the oscillation phenomena of the matrix. For example if for the two first cases the request for the search of the symptoms consists only in the keyword "engine", and that during each session each symptom (noted S1 and S2) is selected alternatively, then the evolution of similarities changes. The detail of the evolution is given in Table IV-2 with a feedback coefficient α equals to 0.5. If the first symptom is selected, then for the second search with the word "engine" the similarity increases by 49% and the symptom S1 will be ranked before S2. If after the second search the symptom S2 is selected then in a third search with the word "engine" the symptom will be ranked first. After the update of both of these symptoms in the feedback matrix, further search does not change the similarity. The reason is that the similarity is constructed by the scalar product of

request and symptom weight vector divided by their norms. If the symptom popularity is updated by the same request vector, its column is just multiplied by a real positive number. The scalar product is a bi-linear form, so the result of $< \lambda x, y >$ is equal to $\lambda < x, y >$. The same applies for the norm: $||\lambda.x|| = |\lambda|.||x||$, and for a positive λ, the ratio of the scalar product by the norm is still the same.

	Empty feedback matrix	→	S1 selected	→	S2 selected	→	S1 re-selected
feedbacks	0		1		2		3
Sim (R, S1)	0.25		0.74958		0.74958		0.74958
Sim (R, S2)	0.315		0.315		0.81517		0.81517

Table IV-2: Oscillation in feedback engine of perceived symptoms

The dispersion effect is then avoided by the selected feedback approach as well as the convergence on the most popularity symptom. Despite the popularity of a symptom, the retrieval algorithm still uses the discrimination factors between the terms to compute the similarity thanks to the property of the cosine formula that measure the angle between two vectors and does not take into account the length. This property confirms that the retrieval engine is stable despite a large number of feedbacks and the similarity measure is limited thanks to the pseudo-linear property of the Euclidian norm and the bilinear form of the usual scalar product.

3.2.2 Guided fault finding

The first guided diagnostic sessions on the cases are simulated with default values for all weight parameters of the different objects. The lifetime of the component is also empty in this first attempt. The test confidence and coverage is set to 100%. The results are depicted in Figure IV-22.

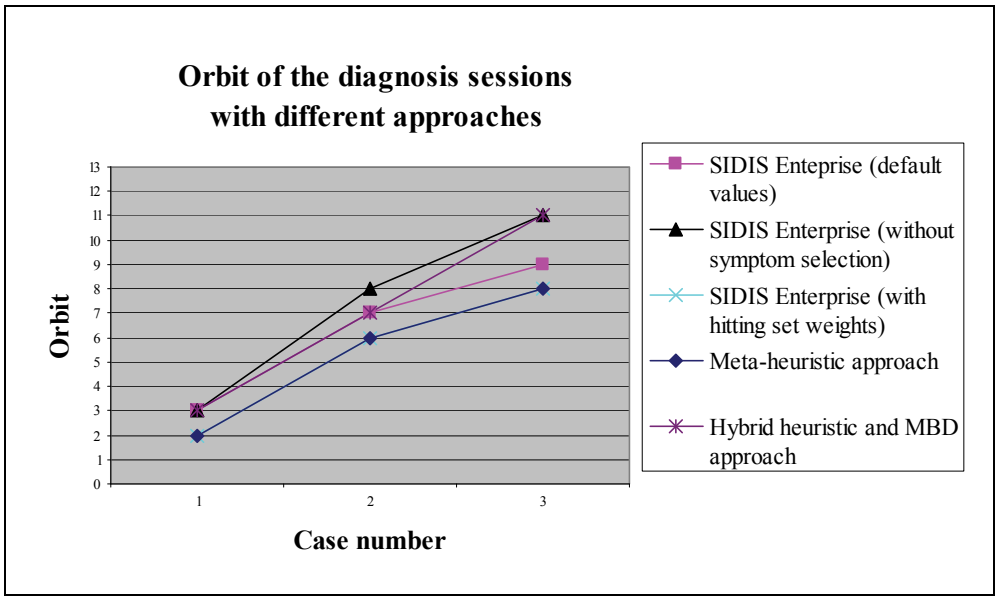

Figure IV-22 : Orbit of the guided fault findings

The first single fault case with a failure in the cooling system is the most trivial case, despite the 5 fault codes that are not associated to rules pointing to the faulty sensors, the average orbit of the guided fault finding procedure is 3 except with the meta-heuristic approach were an orbit of 2 is reached. This result is explained through the logarithmic irradiation of the suspicion rules consequences of the formula in (Eq. III-23). The relevant suspicion rules are elevating the partial suspicious moment of faulty components over 58% instead of 50% with SIDIS Enterprise's algorithm. For this reason the relevant object are ranked higher in the test agenda, reducing the guided fault

finding procedure. Another remark is that if the cases are simulated without the selection of the perceived symptoms, the orbit of the session is higher by an average of 1 test. Thus, the retrieval engine of perceived symptoms plays an important role to speed up the diagnostic sessions.
The second and third cases are more critical, due to he wrong test answer a lot of intermediate candidates are tested first before it comes back to the wrong answer delivered by the ABS test. Again, the meta-heuristic approach shows a good score with one saved test, but again the explanation relies on the irradiation of the suspicion rules (avoiding a test of the ESP).
The third case is more critical with the multiple faults within the ABS and ESP system. Due to the number of disturbing fault codes, which generates irrelevant candidates, the meta-heuristic approach as well as SIDIS Enterprise algorithm (with a hitting set approach for the automatic computation of the weights) shows the best results. But once the first faulty component in the ABS is found and repaired, all methods show similar performance for the localization of the second failure.

As a conclusion, if empty database (no causal links, no lifetime, no test coverage, confidence, etc…) are used, the meta-heuristic approach and SIDIS Enterprise's algorithm show the best performance if the weights of the symptoms are computed by the hitting sets. Moreover, the use of the symptom retrieval engine in these cases allows to save 0.33 test executions in average. The estimated balance to retrieve the symptoms is around 26 seconds and can save up to an average value of 1.65 minutes/session. This result is confirmed by the potential gain for that specific module.

3.2.3 Impact of feedback engine on the diagnosis session

Under the conditions that the weight of the symptoms, the rank of the suspicion rules, causal links, test coverage, confidence, and the lifetime of the component were automatically updated after the 3 cases, Figure IV-23 shows the orbit of the guided fault finding procedure with the updated feedback values.

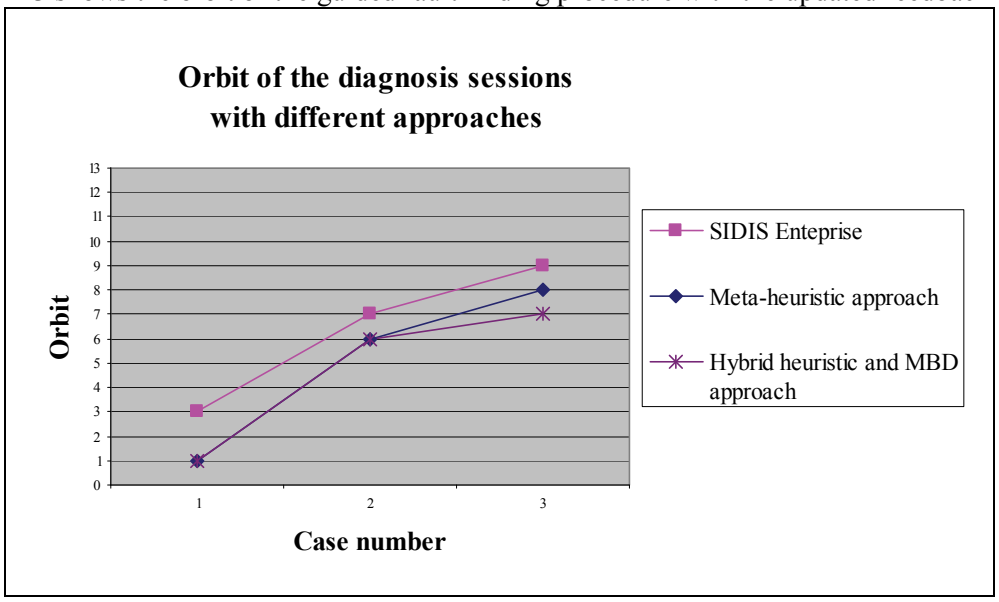

Figure IV-23 : Orbit of the guided fault findings

The details about the updated values and causal links are reported in Appendix B section 2.3. The first case is consequently solved within only 1 test with the hybrid heuristic and causal approach (as with the meta-heuristic approach). The reason is that the ranks of the relevant suspicion rules pointing to the optimal diagnosis path are re-enforced. Besides this the weight of irrelevant fault codes is slightly decreased by 30%. But the main reason of this performance improvement is the creation of the causal links, which contribute to the convergence of the path of the diagnostic session with the optimal path leading to the defect. One drawback here is that the feedback evaluation does not decrease sufficiently the test confidence of the ABS. The result is that still 3 unnecessary tests remain that are performed. Finally, the hybrid approach shows the best results for the multiple fault case where 2 tests are saved compared to SIDIS Enterprise current algorithm.

3.2.4 Synthesis

These cases performed on a low class vehicle prove that the retrieval engine of perceived symptoms allows a potential economy for the diagnosis sessions. Nevertheless, in German, the search engine has a smaller precision than with the other languages (e.g. French and English). The feedback engine of the retrieval engine also proves that an increasing popularity of a symptom does not mean that the similarity diverges. The engine is stable with a growing number of incoming feedbacks and it just benefits lightly to symptoms that have been updated. The weight formula *tf.idf* proves its value in terms of precision compared to the Harman, Okapi and correspondence graph search engine. The guided fault finding modus shows that another irradiation of the suspicion rules can increase the performance (as the results obtained for case 1 with the meta-heuristic or the hitting set method for the weight computations of the symptoms or fault codes). The wrong test answers in the second case have a dramatic consequence on the performance of the diagnostic engines for all methods. With the help of the feedback engine and in particular the contribution of the causal links the hybrid and meta-heuristic strategy allows to achieve an economy of 1 test in this session compared to SIDIS Enterprise's algorithm. Finally, the multiple fault case is also solved with the best performance by the hybrid strategy if causal links are taken into account.

3.3 Vehicle B: verification of the stability of the algorithm

Vehicle B belongs to the middle class cars. It was built in 2004 and the model encompasses more than 1800 diagnostic object nodes, 2000 perceived symptoms, 700 fault codes and 400 expert rules. Only the relevant details about the model and the case are given in Appendix B section 3. The 2 cases taken as examples here study especially the impact of the feedback engine. The cases concern a defect of the sensor for the oil temperature and a defect in the cooling engine, where all interconnected fault codes are sent. It must be outlined that the main focuses on this study concerns the feedback engine because some suspicion rules are evaluated as acute in one case and as not acute in the other case. This study of the oscillation phenomena between both states will verify the stability of the couple feedback and diagnostic engine.

3.3.1 Search of symptoms

The detailed results of the search of the symptoms for both cases are given in Table IV-3 with the same convention as in Table IV-1. The average precision is the highest for the *tf.idf* weight method. Except that in German the results are slightly lower than usual but this is due to the automatic translation of the symptom base: where the word "oil" was translated by the wrong term, causing the discrimination to be based essentially on the word "temperature". This last one appears in numerous symptoms and therefore increases the number of returned and not relevant symptoms. But the lower average precision obtained in German is more due to the nature of the language itself, the stemming algorithm which task is to resolve the morphological variants is too severe in German with a reduction of 9% of the base of terms (in English language the stemmer reduces the BoT by only 3%). This effect is the main cause of the lower precision score in German language.

		Number of returned symptoms											
		French				German				English			
Request	**Limit type**	M1	M2	M3	M4	M1	M2	M3	M4	M1	M2	M3	M4
Wrong display of oil temperature (4 relevant symptoms)	Upper Bound	80	122	93	102	86	131	99	108	94	132	101	110
	Heuristic	4	8	4	26	8	12	12	34	6	8	8	26
ABS display always blinking (1 relevant symptoms)	Upper Bound	61	78	61	80	66	80	69	83	75	86	75	89
	Heuristic	1	4	4	18	2	6	9	12	1	4	4	12

Average precision (heuristic metric)	100	38	63	10	50	25	22	10	83	38	38	12

Table IV-3: Results of the search of perceived symptoms

3.3.2 Guided fault finding

The results of the orbit of the guided diagnosis sessions for both cases are depicted in Figure IV-24. It can be noticed that this time the length of the sessions is more critical than with the first vehicle (reported in chapter IV section 3.2.2). This is related to the number of active fault codes, which is two times higher for this vehicle. Again, these simulations have been realized with empty values for all optional fields (e.g. weight of symptoms, rank of rules, no causal links, no specified lifetime) in order to match the real situation of a car manufacturer (which release the data on the update server with all optional parameters sets to the default values).

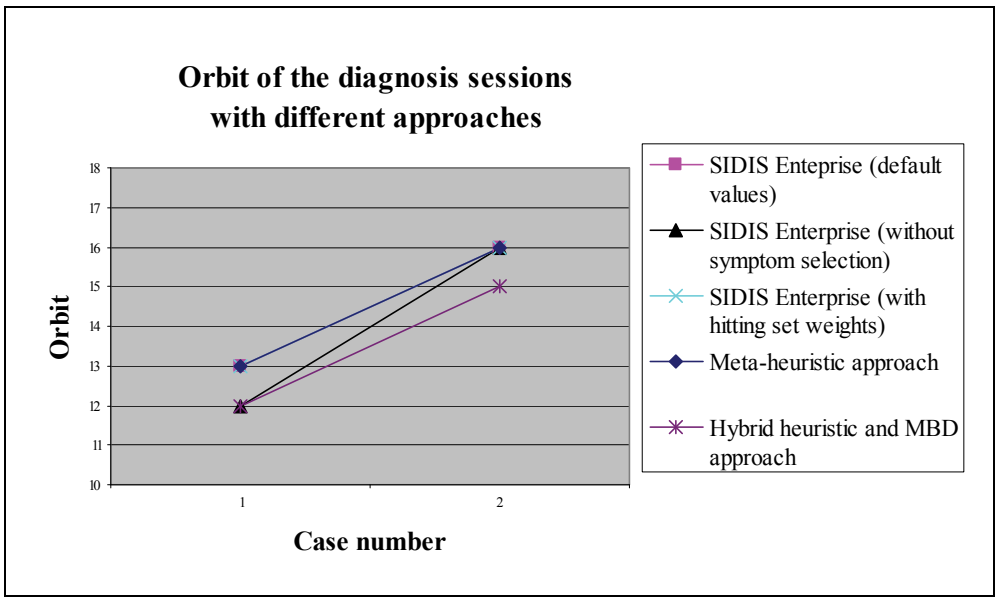

Figure IV-24 : Orbit of the guided fault findings

Due to the high dispersion of some fault codes like “Sensor for cooling system G62” in the diagnostic object tree: “KD1958_cooling fluid regulation”, “F265_19_Operational map for engine cooling system”, “sensor for cooling temperature”, the orbits are comprised between 12 and 16. Here for one time the de-selection of the perceived symptom speeds up the diagnostic session in the first case, because this symptom is linked to a normal functioning component. This explains why the diagnostic session is faster without the symptom, but this example remains exceptional. Due to the reason that quasi all the fault codes point to a similar number of diagnostic objects the hitting set weighting methods do not provide a high enough discrimination power to impact the test ranking providing the same performance as with SIDIS Enterprise’s usual algorithm. The hybrid approach provides the best performance in both cases. Effectively the logarithmic partial moment of the suspicion rules ameliorates the ranking in the test agendas.

3.3.3 Impact of feedback engine on the diagnosis session

Under the conditions that both cases are simulated over 10 iterations whereby all parameters from the feedback engine are automatically updated the orbit of the diagnosis session shrinks immediately after the first iteration. The diagram in Figure IV-25 shows the evolution of the average cardinal (on both cases) for the different approaches. Due to the evaluation after the two first cases, all updates are directly injected in the model. The problem of the consideration of a naive version of the Bayesian networks is that it cleans the model for the diagnostic in a very efficient way. All fault code weights that did not lead to the faulty components are set to zero. Alone this manipulation diminishes

drastically the orbit of the emulated diagnostic sessions. If the cases re-occur the previous values are only confirmed, which explains why the orbit values are constant.

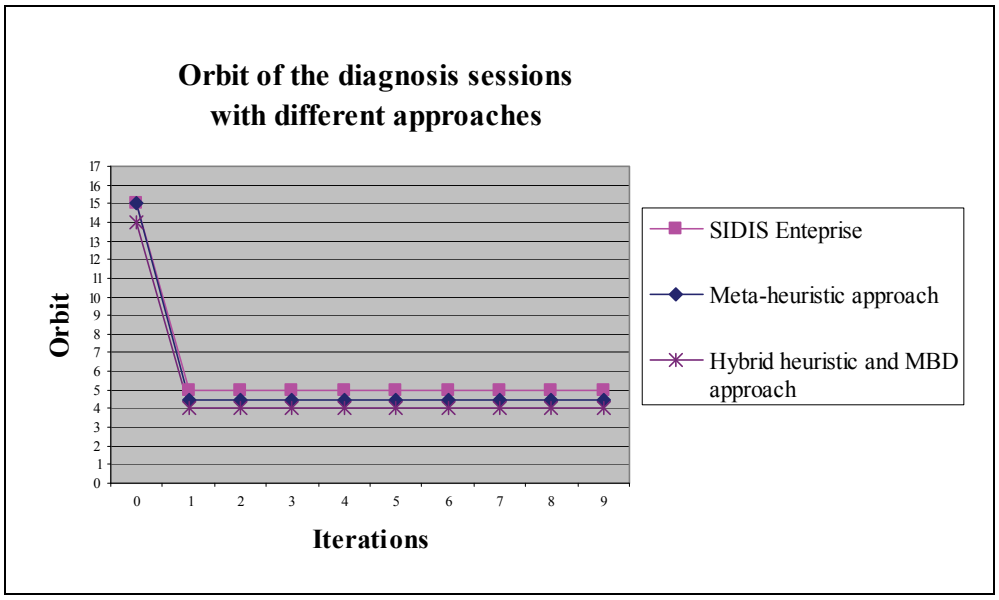

Figure IV-25 : Average orbit of the guided fault findings after feedback evaluation

The hybrid approach shows better results in this case due to the importance of the causal links that appear after the first evaluation of the feedbacks. The fault code "16500_Sensor for cooling fluid implausible signal" has two expert rules R1, R2 pointing alternatively towards the faulty component for case 1 and 2. Both rules are alternatively acute and not acute, which cause their respective ranks to be equal to 1 as described by (Eq. III-30). Despite this oscillation the global behaviour of the hybrid strategy still achieves the best performance. The lifetime of the components is updated in function of the km reading of both cases. After the first iteration all OK tested nodes have a lifetime that is equal to 63000km (because the vehicle in case one had 37000km and in the second case 63000km, which is the highest value to build the hypothesis). Intermediate nodes in the model of the vehicle, which are tested in both cases with answers: OK or NotOK could not be updated because they are classified in both leaves of the tree with the same number of occurrence. Thus the Giny impurity factor is equal to 0.5 over the acceptability limit of 0.2.

3.3.4 Synthesis

As mentioned previously this business case is based on a middle class vehicle built in 2004. The numerous fault codes that appear at the beginning of the diagnosis sessions are typical of a real case situation. The perceived symptom search engine experience proves again that the *tf.idf* formula provides the best precision. The hybrid approach confirms also an improvement for the diagnosis session, but the couple diagnostic engine and feedback raises some problems that could occur in an industrial environment. Actually, if the number of collected feedbacks for a specific vehicle variant are all related to the same faulty behaviour of a component (for example a quality defect of a new series of vehicles) then the naive Bayesian belief network that is basically used to update the vehicle models converge with evidence towards the relevant data of that special case. So despite the valuable contribution in terms of performance depicted in Figure IV-25, the suppression criteria of a suspicion rule, or an updated weight coefficient of zero for a symptom, fault code should always be validated by an expert. If not, the module can suppress hardcoded diagnosis knowledge entered by the authors. The best solution to avoid this case is to wait for a large number of protocols before the evaluation process, which ensures a diversity of faulty behaviours and avoid a too strong convergence. Another solution is during the evaluation to take only in account objects that have at least 10.000 entries from protocols, which is probably the most appropriate solution, because each object (e.g. suspicion rule, symptom weight, causal link) is evaluated separately depending if it has received enough data from feedbacks.

3.4 Vehicle C: verification with critical dispersion of the suspicion

Vehicle C belongs to the luxury class and was built in 2006. The model encompasses 2100 diagnostic objects, 800 fault codes, and around 600 expert rules. The relevant details about the model and the case are given in Appendix B section 4. The 2 cases taken as examples here study especially the impact of the dependencies between the ECUs. Here, two faulty behaviours are supposed: the sensor of the engine speed, which causes the diagnostic session to begin with approximately 60 fault codes (which suspect around 100 diagnostic objects) and the pressure sensor of the brake pedal, which causes arround 50 fault codes. The focus of this study is the couple diagnostic with feedback engine. This time the feedback evaluation takes other non described cases into account than the 2 studied in order to focus on the possible divergence effect of the belief network.

3.4.1 Search of symptoms

The detailed results of the search of the symptoms for both cases are given in Table IV-4 with the same convention as in Table IV-1. The average precision is the highest for the *tf.idf* weight method, which confirms definitively the performance pre-domination of this method. Again the precision is lower in German language. Nevertheless, the precision of the search engine with this method still delivers acceptable results within 1.2-1.5 seconds.

		Number of returned symptoms											
		French				German				English			
Request	**Limit type**	M1	M2	M3	M4	M1	M2	M3	M4	M1	M2	M3	M4
Engine does not start (1 relevant symptoms)	Upper Bound	2	126	80	120	8	87	85	72	2	102	96	72
	Heuristic	1	22	18	34	2	34	8	17	1	26	12	18
Indicator of ABS always on (2 relevant symptoms)	Upper Bound	64	82	66	92	61	74	61	78	66	87	75	94
	Heuristic	3	8	6	17	6	4	6	11	3	9	6	12
Average precision (heuristic metric)		**83**	**15**	**19**	**7,4**	**42**	**26**	**23**	**12**	**83**	**13**	**21**	**11**

Table IV-4: Results of the search of perceived symptoms

3.4.2 Guided fault finding

The results of the orbit of the guided diagnosis sessions for both cases are depicted in Figure IV-26. Both cases are simulated with default values for all weight parameters, which cause SIDIS Enterprise's orbit being very high (13 and 16 tests). In the second case, if SIDIS Enterprise's algorithm is run and the weight of the fault codes and symptoms assigned by the hitting set method, the guided fault finding procedure is lower, providing an economy of 2 tests. The reason is that the relevant fault codes in this case point to less diagnostic objects than the irrelevant ones. Thus, a higher weight is associated to the fault codes that point toward single diagnostic objects through the suspicion rule and the suspicion moment of these objects is higher and impact positively the test ranking. In the first case the meta-heuristic and hybrid approaches show smaller orbits than SIDIS Enterprise, which is explained by the way the suspicion moment of the rules is calculated, which benefits to their ranking in the test agenda (specially for the heuristic metric). The precision factor plays also an important role in hybrid approach because the test ranked first in the test agenda are all leave nodes. In the path of the guided fault finding procedure of SIDIS Enterprise algorithm there are also some intermediate nodes that lead to the faulty component. Finally, in average the hybrid heuristic and MBD based approach shows the best improvement for the automotive guided diagnosis on these cases, but this gain must be put in balance with the feedback evaluation engine, which is detailed in the next section.

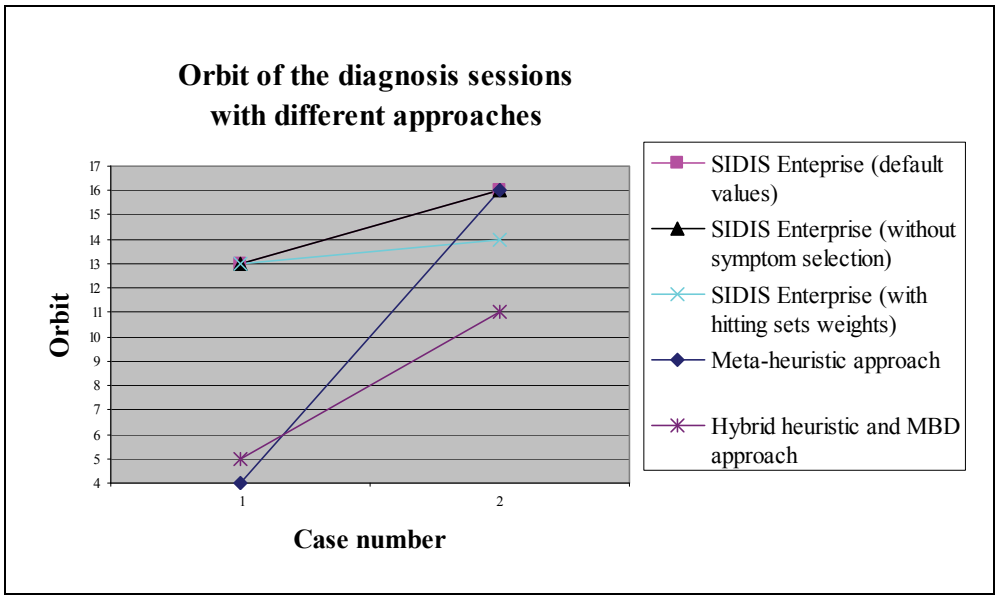

Figure IV-26 : Orbit of the guided fault findings before feedback evaluation

3.4.3 Impact of feedback engine diagnosis session

In this simulation, the weight parameters as well as the lifetime, suspicion and causal rules are updated after the evaluation of the 2 cases presented previously and 8 other cases, which concerns other components in order to study the influence of the dispersion of protocols on the guided fault finding. The results for both cases are depicted in Figure IV-27.

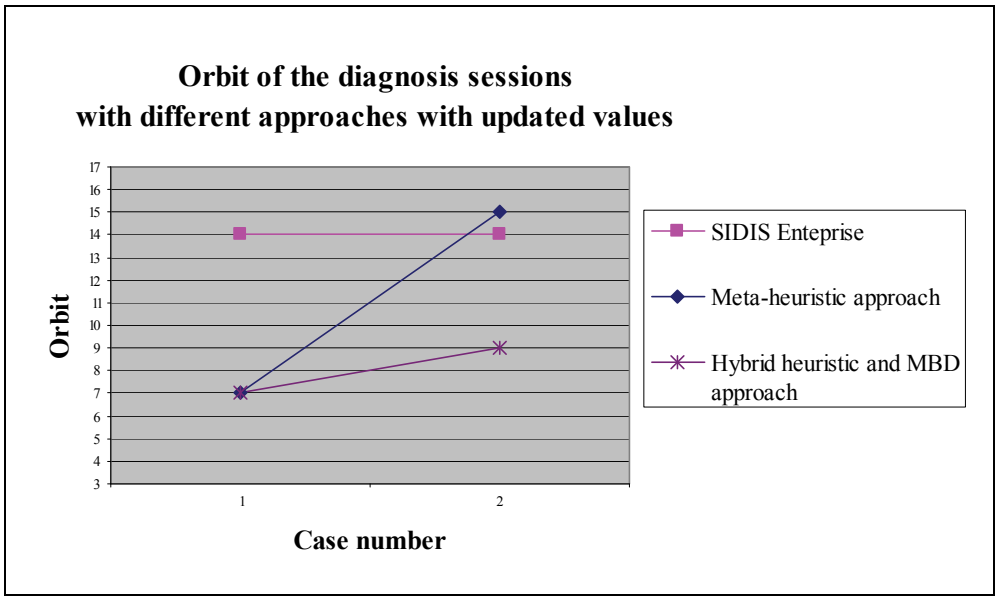

Figure IV-27 : Orbit of the guided fault findings after feedback evaluation

Firstly, the diagnosis sessions performed with SIDIS Enterprise and updated values of the different weight parameters provide in average an economy of 0.5 test. On one side the evaluation of the feedbacks augment the orbit of the first case and on the other side decreases the orbit of the second case. So despite the dispersion of the provided protocols that do not converge on the handled case, SIDIS Enterprise algorithm is still stable and the diagnosis session provides a light performance improvement. For the hybrid approach and the meta-heuristic approach, the creations of the causal links are perturbing the flow of the diagnosis procedure. The hybrid approach shows a higher cardinal for the orbit of the first case because the influence of the causal links plays a higher importance.

Furthermore, the majority of these links concern the evaluation of the other 8 cases that are added for the feedback evaluation. Thus, the algorithm is oriented on a wrong path during the session. Despite the 2 more tests that are performed, the approach has the same orbit in average on both cases before and after the feedback evaluation. For the meta-heuristic approach the impact of the causal factor is higher than in the hybrid approach. With the resulting in-acute causal links that are created from the feedback evaluation, the path of the diagnostic sessions with this approach increases particulary for the first case. The lifetime parameter has only a very low contribution on the suspicion moment of the candidates, but experiences with a higher factor of the lifetime do not contribute to a performance gain.

3.4.4 Synthesis

As mentioned previously this business case is based on a luxury class car built in 2006. The numerous fault codes that appear at the beginning of the diagnosis sessions are typical of a very critical real situation. The faulty behaviours of the cases (sensors for the motor rotation speed and the pressure sensor of the brake pedal) fire between 40-60 fault codes that are linked with over 100 diagnostic objects inside the motor management, ABS, BCS, etc… The perceived symptom search engine experience proves again that the *tf.idf* formula provides the best precision. The hybrid approach confirms also an improvement for the diagnosis session. The couple of the diagnostic engine and feedback evaluation in a high dispersive environment (80% of the cases given to the feedback are not acute for both cases), shows the stability of the heuristic approach despite the divergence caused by the evaluated beliefs of the Bayesian network. Thus, the cases based on this model confirm the stability of the diagnostic and feedback strategies in a critical environment.

3.5 Summary

The parameters used in the reported cases (chapter IV sections 3.2 to 3.4) are listed bellow (notice that two different parameters are denoted α: once for the weight of the popularity matrix and another for the order of the entropy):

- α : the coefficient of the perceived symptom popularity fixed to 0.5, see (Eq. III-35)
- a_0: Weight of the precision factor for the meta-heuristic approach fixed to 10%, see (Eq. III-23)
- a_1: Weight of the test cost factor for the meta-heuristic approach fixed to -20%, see (Eq. III-23)
- a_2: Weight of the life-time factor for the meta-heuristic approach fixed to 10%, see (Eq. III-23)
- a_3: Weight of the partial moment of suspicion rules for the meta-heuristic approach fixed to 100%, see (Eq. III-23)
- a_4: Weight of the partial moment of the causal relationships for the meta-heuristic approach fixed to 50%, see (Eq. III-23)
- α : the order of the entropy for the hybrid approach fixed to 0.9, see (Eq. III-29)
- β: the linearization coefficient between the heuristic and the causal partial suspicion moment in the hybrid approach fixed to 0.4 see (Eq. III-29)
- γ: the exponent of the linearization coefficient in the hybrid approach fixed to 1, see (Eq. III-29)
- c_{LT}: the relative influence of the life time suspicion moment in the hybrid approach fixed to 0.2, see (Eq. III-29)
- Threshold value for the acceptance of the creation of a suspicion rule or a causal link fixed to 20% (cf. Chapter III section 3.1.1)
- Threshold value for Giny impurity factor fixed to 0.2 see (Eq. III-32)
- Threshold value for the validation of a lifetime or test cost hypothesis fixed to 90%

Concerning the perceived symptom search engine, the earch and selection of symptoms for the cases generally prove that a selection of symptoms has a positive influence on the performance of the guided

diagnosis sessions (except for one case of vehicle B). The average performance improvement of the orbit is 0.28 tests (which represents an economy of 1.4 minutes/sessions). Nevertheless, this economy of time must be put in balance with the number of relevant symptoms, because for some manufacturer the perceived symptoms database is not used and for others this knowledge plays a central role. It has to be remembered that in the cases the perceived symptom database is filled with over 2000 entries in order to test the engine in the worst case situation. The weight formula *tf.idf* provides the best results in term of precision (average of 82% in French and English), except in German language with an average precision of 51%. This effect is mainly due to the German stemming algorithm, which is too severe when it comes to parse the symptoms for the construction of the base of terms. The result is that not only morphological variants of words are eliminated but also words with complete different meanings are recognized as the same with the consequences that more unrelated symptoms are returned for a request. The used feedback algorithm based on the incrementation of the popularity matrix also helps to return relevant symptoms. Thanks to the implemented cosine formula for the similarity and the feedback treatment the result computation relies only partially with a limited impact on the popularity and the symptom discrimination for a given request still remains on the informativity content.
The hybrid approach for the guided fault finding modus diminishes the orbit by an average value of 2 tests if the default weight values are taken (which correspond to the real situation of many car manufacturers). The meta-heuristic approach allows only to save 1.72 tests in average compared to SIDIS Enterprise's current algorithm.
After the evaluation performed by the feedback engine and the values of the parameters updated according to the threshold values, the models are changed. The emulated guided fault finding procedures with the updated values lead to an average economy of 1.75 tests with the meta-heuristic approach and of 3 test with the hybrid approach. Besides these results, other parameter variations are realized especially with the hybrid approach and the parameters β, γ, but the results were disappointing (between $\pm$ 3 tests/guided fault finding procedure).

The overall performance improvement is confirmed through this use case analysis in different types of environments: single and multiple faults, wrong test answers, feedback convergence behaviour, critical faults with high impact through ECU dependencies and divergent feedback entries. Nevertheless, some questions will be addressed in the next section in particular for the feedback engine (cf. chapter IV section 4.1).

4 Discussion and outlook

4.1 Discussion

As it is mentioned the *tf.idf* formula for the weight calculation within the search module of the perceived symptoms shows the best performance in terms of precision. Nevertheless, the precision is lower in German language because the stemming algorithm is not adapted for this language. The other weight formulas: Harman in (Eq. II-6) and Okapi in (Eq. II-8) provide lower scores than *tf.idf* , the reason relies on the discriminative calculation of the informativity. In the Harman formula the weight follows a pseudo-linear line for increasing term frequencies with the effect that the difference is quasi step-wise between the symptoms. This effect is also observed for the Okapi formula. Furthermore, both formulas use a normalization factor for the weight calculation, which is set by the length of the symptoms (in number of relevant words). This normalization procedure could be adequate when it comes to the search of documents with variable length in order to take into account the fact that long documents have higher term frequencies. The normalization within the perceived symptom database has for effect that the intervals between the weights of terms are smaller and the norms between the symptom vectors follow the same trends. Finally, the result is that more symptoms are returned with a negative influence on the precision. For example a request like "indicator of exterior temperature does not work" returns symptoms with "indicator of oil temperature…", "indicator of engine temperature", and "indicator of water temperature in cooling system…" etc… The Harman and Okapi formula are not adequate for the perceived symptom database; they may be more specific for the indexation of

large documents or electronic books. The logarithmic behavior of the *tf.idf* formula provides the best discriminative power, which is confirmed by the case studies. A search with the word "display" alone returns a lot of symptoms like "display of ABS…", "display of temperature", "display of ESP…", then with the help of the heuristic acceptability limit described in (Eq. II-16) only the symptoms containing the minimal number of words will be filtered. Because the less words they contain, the nearer they are to the weight request vector, which causes the cosine of the angle between request and symptom (the similarity) to be high. But this is only the case for requests containing an unique keyword like "display" or "engine". If the request contains at least 3 keywords like "display exterior temperature" then the correct group of the symptom tree is expanded. The usage of the heuristic acceptability limit is also confirmed by this study. For the design of a user-friendly interface of the module for the search of the perceived symptom some recommendations issue from a comparative study of software usability evaluation [Azarian, Siadat, 2007] allows to lists the following points:

- The search module should construct the weight matrix and load the base of terms when it starts (to save time).
- A field (e.g. a textbox) should be on the center of the screen with an explicative text like: "enter your request or keyword".
- If the request does not contain words that are in the thesaurus or the base of terms then an information message should be schown which explains why the research cannot be successful.
- If a research delivers some symptoms then the tree should be schown on the GUI (Graphical User Interface) and the resulting symptoms should be expanded and have a special backcolour compared to the ones that are not in the results.
- If possible, the words of the request which are used in the returned symptoms should be written in bold in order to explain to the users why these results have been delivered.
- Finally, for a simple selection, a checkbox should be associated to each node in the perceived symptom tree and the user can validate (or not) the symptom delivered by the search engine. The default value of the box should be unchecked because as seen the search engine have the tendency to deliver too much symptoms; obliging the technician to de-select all returned symptoms will not benefit to the global time of a diagnosis session.
- The user should have the possibility to re-do a search if he wants to change or precise his request.
- The car manufacturer should have the possibility to configure if the selection of a symptom in the guided fault modus is obligatory (for example with an option like writing the request in the protocol if no symptom was returned with that request), but for more convenience a button that allow to skip this step should still be available. This button allows the technician to keep a certain degree of control of what he does.

The ergonomic considerations of this module can contribute to the success of the usage of the tool and allows to save time for each diagnostic session. A quantification of the saved time was difficult to perform, because it depends if a perceived symptom has relevant suspicion links for a considered case. Currently, german manufacturers do not use the symptom database, making an evaluation difficult, but with another database (with over 2000 symptoms) associated with relevant suspicion links, the time saved can reach around 0.28 tests/diagnosis session. If this average performance improvement is scaled to the size of a European after-sale network of a manufacturer, the module allows to save approximately 240 working hours a day (and around 56 thousand hours a year). Finally, as seen, the part of the feedback engine dedicated to that module allows to favorise symptoms that are selected for certain keyword compared to other. Despite an increasing popularity of the selected symptom, the influence on the similarity metric is limited by the coefficient α. This limitation is caused by the bilinearity of the Euclidian scalar product and the positive scalability of the Euclidian norm (as demonstrated in chapter III section 3.1.3). Thus the principle of equi-repartition of the perceived symptoms is still preserved for the search engine. The feedback coefficient α of the search engine should remain inferior or equal to the value 0.5 and increase gradually with the number of feedbacks. If no feedback has been processed the similarity issue from the request weight vector and the popularity matrix is always equal to zero, which cause the normal similarity to be lower (due to the multiplication with the feedback factor). This feature of the feedback engine could not be quantified in terms of performance gain because no typical request from technician could be collected to constitute

a test database, but the engine fits particulary well for request with only one or two keywords. For example, a request entitled “headlights” would return symptoms like “headlights do not work”, “headlights do always work”, “Indicator of headlights position does not work” etc… And if the first of these symptoms have been selected, then it will be returned for the next search with this keyword and the other will be filtered. This consequence fits in an environment were worker are under time pressure and where technicians will probably use only one or two keyword to specify the request. One strong limitation of the selected approach for this module is the poor vocabulary which is taken into account, composed only by the base of terms based on the symptoms, but the next section discusses possible improvement to overcome this drawback (cf. chapter IV section 4.2). The last point to discuss on this approach is the capapability to reason in different language. The same algorithm was used for a French, a German and an English symptom database (with a segmentation procedure in German).

During the guided fault finding sessions, the hybrid approach which combined the heuristic approach of SIDIS Enterprise and the model based diagnostic via the causal links shows the best result. The factor related to the evaluation of the suspicion rule alone allows to improve the diagnosis. This factor relies on a logarithmic function to assign a suspicion moment to the diagnostic object, which compared to the linear function of SIDIS Enterprise provides a better distinction between the candidates. The higher the number of active suspicion links, the higher its partial suspicion moment. The results in terms of performance for this factor can be shown by the simulation with the hybrid approach as well as the meta-heuristic approach. Both have the same behavior for the evaluation of the partial suspicion moment induced by the suspicion rules. The parameter β is of no importance when the models do not contain causal links, because the candidate generation still remains the same with different values β, except that the partial suspicion moment is scaled. The parameter α influences only the precision factor, when the models do not contain causal links. For the reason that most of the suspicion links are pointing toward diagnostic objects with the same level, the influence of the precision factor is therefore negligible. Once the causal links are added to the model, the parameter β plays a central role because it balances the certainty between suspicion and causal links. If the balance exceeds the value 0.4 the causal links are nearly ignored and the orbits of diagnostic sessions with the hybrid approach are very similar to the ones obtained by SIDIS Enterprise’s algorithm. For lower values, the causal suspicious moment comes in the foreground for the candidate generation, with the consequence that the hybrid approach emulates a directed graph exploration. The reason is that the causal relation coresponds to frequencies of correlated suspects, the graph exploration induce source elements (with the maximum of outgoing causal links) to be ranked first in the test agendas and these are not always the cause of the faulty behavior of the vehicle. This is the main drawback of the whole feedback engine, all adapted weigths and new links corespond to frequently encountered cases and not to background knowledge. This case is illustrated with the simulation on vehicle C (see chapter IV section 3.4.3), where the feedback engine takes other cases into account before the weight parameters are updated and new links created. The variation of the other cases creates a dispersion effect which does not allow a valuable performance gain for the diagnostic sessions. In summary, the limitation occurs with the idea that correlation about faulty components is taken as causality. To avoid this effect, the next section (see chapter IV section 4.2) proposes some possible solutions that may be implementable in SIDIS Enterprise.

4.2 Outlook

For future improvement on the perceived symptom search module, the orientation would be a semi-automatic completion of a thesaurus. A specific automotive thesaurus like the one used in the prototype shows the best performance improvement, when request contains terms that are not specified in the base of terms. The reason is that the language used for the perceived symptoms is strictly normalized for each car manufacturer in order to facilitate the translation, but the after-sale network is not aware of the specificity of that normalized language, therefore they would use synonyms in their request. The unknown terms of the request should been protocoled and related to the selected symptoms by the technician. A simple Bayesian network with the frequency analysis could then show, which unkown terms may be associated with known terms. These association beliefs

between the terms should be suggested to a linguist, which task is dedicated to maintain the thesaurus of the car manufacturer and validate or reject the association-relations, which are based on the frequency analysis. This method has the great advantage to enhance the base of terms without background knowledge, which is the ground why the final validation has to be performed by a human operator [Hagiwara et al., 2005], [Grefenstette, 1994].

Despite the performance improvement operated by the hybrid approach of the diagnostic engine it is possible to achieve better results with the evaluation of the causal links. In the current approach, they are evaluated as a function depending on their weight and if they are active or not. With the help of a graph exploration algorithm, the candidates can be generated based on the maximum number of active suspect. The next test agenda could then be generated by elimination of the causal chain in the case of an OK test answer with the use of forward propagation. Thus, the diagnosis task can be assimilated to the traveling salesperson problem except that the topology of the graph represents the possible causalities and the nodes the possible suspects. The problem is that the causality graph can still contains cycles and the exploration may lead to a combinatory explosion. But recent research as in [Asahiro et al., 2007] proposes a weighted nearest neighbor algorithm for the graph exploration, which is more competitive. Another orientation could be used for the test agenda: instead of actualizing dynamicly a list of tests, the test agenda could be presented as a fuzzy cognitive map. This means that all suspected diagnostic objects as well as their child should be presented to the user with the possiblity to expand branch of the diagnostic object tree that have a lower suspicion moment. The links between the relevant objects should also been represented. This will lead to a map were suspected nodes are clearly identified (a color gradient for each node should be applied as a function of the suspicion moment). The technician would then have the choice of exploring precisely a special area or subtree of a group of component. This method, for the guided diagnostic procedure, would need more involvement of the technician and his cognitive skills. But with the benefit that his skills, if he leads effiently to the fault localization, would later been learned by the feedback algorithm. The fuziness of the map results from the dispertion of the suspicious moment of all objects, and from the cognitive aspects of the perception based information processing from the technician [Massoud et al., 2005].
As discussed previsouly, a strong drawback of the feedback engine is to converge to the most frequently encountered cases and to adapt the weight parameters toward that direction. Despite some proposed threeshold values, it is the philosophy of the naive version of the Bayesian network, which places correlation and causality on the same level and leads to the creation of causal links which do not corespond to a physical interaction between components. In order to auto-complete the models with causal links which represent relevant physical interactions, the components in the diagnostic object tree should be labeled into different categories (e.g. electrical, mechanical component) and only links between the same types of components should be validated. This would be a weak variant for the auto-completion, but it already needs a key-value dictionnary to store the labels of all diagnostic objects, which leads to higher costs for the feedback engine. Moreover, for some components which interact between different physical domains as an air flow meter sensor, it would be difficult to assign a label to the component, because some components belong to different labels. In the air flow sensors the thermistor is labeled as electronical but the pipe bringing the air is mechanical. Thus, the best solutions which will also benefit to the diagnosis will be to distinguish the failure modes of the components and adapt the causal relationships depending on the particular encoutered failure mode. If the suspicion links points directly towards precise failure modes, then the hybrid approach would search the causes within the filtered causal graph which is related to the detected failure modes. This method could strongly increase the fault localization process, but it still augments the cost of the development of models. A generic set of failure modes could be developped and put at the disposition of the authors to add immediately some classical failure modes to a component. For example an ECU could have the generic failure modes like: Incorrect programm variant, short to ground etc. With the precision of the failure modes in the models for the diagnosis it would be possible to achieve higher performances for the guided diagnostic sessions, but more effort has to be spent in the developments of models.

Conclusion

1 Summary

This dissertation had explored the field of diagnosis applied to the automotive industry with the help of the commercial product SIDIS Enterprise. At the beginning we analyzed the importance of the after-sale world in the car manufacturer's value chain, as well as the requirements of manufacturers for computer assisted diagnostic tools. The bottom line of the diagnotsic tools designed to equip the whole after sale network of a manufacturer was to limit as much as possible the costs of the development of models. Furthermore, the analysis of the parts of electronic breakdown and electronic equipment shows that both tendencies augment with the effect that the failure localization becomes more and more complex. The need for computer assisted tools for the car manufacturer is therefore essential to save time for the fault localization, for the client's satisfaction, and for the image of the brand.
This thesis leads to a global modular framework for automotive diagnosis composed by:

- **A perceived symptom search engine**: for the prolongation of symptoms and manifestations in natural language in the computer assisted diagnostic tool.
- **An automatic ODX exchange system**: for the automatic sorting of ECUs file description.
- **A hybrid heuristic and model based diagnostic strategy**: for efficient fault localization in the workshops.
- **A feedback engine**: in order to benefit from the return of experience

The design of these components in forms of modules was selected to fit inside SIDIS Enterprise's architecture. After the simulations executed on the tree vehicles in chapter IV, it is possible to quantify the benefits of this framework in terms of performance.

- **Statement 1**: The time economy per guided fault finding procedure reaches 3.28 tests per diagnostic session.

This economy is possible because of three factors: the first one depends on the perceived symptoms. In the case that an abnormal behavior of the vehicle can be found in the perceived symptom database and if these relevant symptoms are linked to diagnostic objects through suspicion rules, it will accelerate the diagnostic (because the candidate leading to the faulty components will be ranked first in the test agenda). The perceived symptom search engine allows to save time for the navigation through the corresponding tree, but the real gain depends on the number of suspicion links, which are related to the relevant symptoms. Because these ones are effective in the computation of the first test agenda and they allow converging toward the optimal diagnostic path for a given fault in a vehicle. The second factor which is emphasized within this time economy is related to the hybrid approach. Especially the discrimination induced by the logarithmic behavior proved its benefits. Thus, the economy per diagnotic session can reach an average of two tests. Finally, the last factor comes from the causal links issue, from the feedback evaluation engine which can allow an economy of 1 test per session. But this result must be taken carefully, as previously mentioned in the discussion, the feedback engine auto-complete the models with links and weight coefficients issue from a frequency analysis. Thus, it is not the causality which is modeled but the correlation between variables based on a statistic evaluation.

- **Statement 2**: The quality of off-board diagnosis was improved, with an average gain of 2 tests per session.

This statement results specially from the implemented reasoning in the diagnostic engine. The core of the diagnosis relies on: the evaluation of the suspicion links, the causal networks and the lifetime, which corresponds to a more tangible physical reality. The suspicion rules are still at the top of the

candidate generation phase, but the logarithmic suspicion irradiation provides a higher discriminative power. Besides, the dependencies are modeled by causal links like the lifetime is by the Heaviside function. Both allow the models to reflect partially the complexity of vehicle's electronic for the diagnostic task. Finally, this hybrid method bridges the model based world with traditional expert systems (which are currently wide spread in car manufacturers after-sale networks).

- **Statement 3**: The cost of model development and model optimization was reduced.

Thanks to the ODX exchange module, the ECUs description files can be imported automatically in the authoring system of SIDIS Enterprise. This feature allows an important cost economy for car manufacturer, because ECUs are very often changed. Nevertheless, it is difficult to quantify exactly the gain in term of performance because many elements are missing to simulate a real situation like: knowledge and experience degree of the author responsible of this sub-part of the models, available documents, etc. The percentage of 37.2% time economy for the import of ECU description files of a vehicle equipped with 80 ECUs is obtained by an extrapolation on one ECU file imported manually and then automatically with the module. At least, the models are completed with new suspicion and causal links as well as adapted weight coefficients thanks to the feedback evaluation engine. Thus, the problem of authors who do not commit themselves to specify a weight coefficient is avoided. But as mentioned, the results issue from the feedback engine must be taken carefully because it converges toward the most frequent cases. The threshold values for the automatic completion allow to partially resolve this problem, but the definitive validation of the creation of new objects should be made by an expert.

The work presented in this thesis has allowed to provide a practical oriented framework for the automotive diagnosis, with the combination of different knowledge sources and their access for an efficient fault localization like: expert knowledge (suspicion rules), construction knowledge (causal links), return of experience (feedback engine), qualitative failure description (perceived symptoms). The next section reports about possible future work to enhance this modular framework.

2 Outlook

2.1 Future developments and extensions

Concerning the research of perceived symptoms, the results reported in this thesis show that one crucial factor can improve the search module. It is the thesaurus, which allows achieving a better precision and recall for the module by the association of new terms with known terms (from the BoT). A practical approach, inspired from the work of [Hagiwara et al., 2005] can allow to develop a specific module in SIDIS Enterprise for the automatic completion of the thesaurus. The principle relies in the association between the stems in the request and the selected perceived symptoms by the technician in the workshops. A Bayesian network could compute the beliefs association between the stems in the symptoms and in the request. Thus, if the belief probability reaches a certain threshold value, the terms used in the request could be associated with the terms used in the selected symptoms and the frequency of this association divided by the total number of feedbacks containing the considered world gives a confidence index. If a new term occurs in a request, it would be replace by the preferred term (from the BoT); the term frequency would no longer be an integer but would be replaced by the confidence index which will slightly decrease the similarity between request and symptom (compared to the similarity obtained with the preferred term in the request). This method has the strong advantage of processing automatically the data and the thesaurus will not grow endlessly due to the fact that the specific automotive language is limited.

For the improvement of the ODX exchange system, three ideas can be used. The first one concerns reinforced learning techniques. One challenge that arises in reinforced learning is the trade-off between exploration and exploitation. To obtain the best rewards for the link creation between ODX rough data and ECU functions, a reinforced learning agent must prefer actions that it has tried in the

past and found to be effective in producing reward. There are two kinds of actions in this situation: the creation of a link between ECU's rough data and a function or the creation of a new function and its link to the communication data. The reward in this situation can be quantified by a human operator, who confirms or decline the performed actions. The agent has to exploit what it already knows in order to obtain reward, but also has to explore the new created links in order to make better actions of selection in the future. In reinforced learning the agent must try a variety of actions and progressively favor those that appear to be best. This computational approach would lead to an automated goal-directed ECU file importation and classification. It is distinguished from other computational approaches by its emphasis on learning by the agent from direct interaction with its environment. Nevertheless a formal definition of the agent's environment in terms of states, actions and reward must be provided. These definitions intend to represent essential features of the artificial intelligence problem. Another idea for the improvement of the multi-agent system is to include an uncertainty variable for the string comparator agent and develop a pro-activeness of the other agents. The string comparator agent uses so called bags of words for the comparison, but for each result an uncertainty. This uncertainty could then be used by the global supervisor agent for the decision making process (for the schedule of a deeper examination of the parameters and values of a fixed function). By this simple additional feature the mutli-agent system would have higher communication skills and tend to be more like a joint cognitive system. The third idea for this module is to give the agent alternative cognitive capabilities. Instead of a string comparison, a new agent should be present. This one should reason on the ECUs characteristics and on a formalized ontology of ECUs function in order to focus on relevant data during the search process. This method requires an abstract modeling of ECUs functionalities, which augments the costs of the model. A human operator will precise in which part of the vehicle the new ECU is intended to operate (e.g. engine management) and if the ontology is suited to the problem, the new high level agent will focus request a comparison between functions related to that vehicle sub-system (e.g. Read rpm). Such a reasoning can help to deliver better results for the actions due to the fact that the agent uses concepts and distances between concepts for the decision making process instead of a weighted comparison of function's characteristics.

The diagnostic algorithm could also be ameliorated with the help of graph exploration and refutation mechanism. Each time a test is executed, the state of the current component associated to that test should be actualized by forward propagation through the causal network. The test ranking should then be organized by a metric which combines the smallest number of refutations and the associated test cost. Whereby, the number of refutation should integrate fuzzy logic and specially the certainty of a causal relationship. This construction of hyptotheses for each element goes toward a consistency based approach, but due to the high number of suspected elements in the first test agendas it may need more computational resources. Moreover, the evaluation of each hypothesis could also take some time. A special attention must be brought to the development of the fuzzy evaluation of the number of refutations, where the certainty of the causal relationships plays a central role. One interesting method would be to present the test agenda as a fuzzy cognitive map instead of a static list of 7 tests. This presentation would implicate more the technician in the workshop on the diagnosis task and allow to develop a reinforced learning from feedback protocols. The developed feedback evaluation engine in the scope of this thesis relies on the construction of beliefs networks, which 'measure' the dependency between the variables with a frequency analysis. The coupling of fuzzy cognitive maps with reinforced learning would be a more interesting solution for the diagnosis, because the diagnostic path of each session would be analyzed and the reinforced learning techniques would adapt the coefficients and causal links to approximate the optimal path. The main drawback of this technique is the computational power required for the protocol evaluation, because each diagnostic session needs to be reproduced on the feedback server and the hypothesis space needs to be constructed and evaluated with all possible cases for a given vehicle. Despite this complexity, an intermediate step could evaluate a heuristic score of a given hypothesis (e.g. a higher weight coefficient for a fault code or a new causal link) for some given cases. If the intermediate score proved that the hypothesis is correct, then this orientation should be further investigated. This strategy allows to avoid the combinatory explosion of the hypothesis exploration and so the systems learns to optimize the models for the diagnosis.

For the implementation as well as for the extension and new development of the modular framework, the following changes, trends and challenges in the automotive industry and global economy have to be considered.

2.2 Future trends in automotive diagnosis

There are some trends that may cause some evolution in the automotive diagnosis in the near future. In the context of socio-economic trends in the future, an increasing individualization will drive the need for customized cars with a multitude of special features. The aging of populations means that a bigger share of drivers will have special needs with respect for example to comfort and support of eyesight. It is obvious that oil resources are a special concern for the automotive industry and number of oil consumers is increasing due to more drivers in the BRIC (Brazil, Russia, India and China). According to estimations of the Organisation for Economic Co-operation and Development (OECD), the total number of vehicles in OECD countries is expected to grow by 32% from 1997 to 2020 and, on a global scale, by 74% in the same period. The anticipation of future scarcity of conventional fuels is a major driver for several technological developments. Environmental legislation is one of the major factors of development in the automotive sector. This holds especially true for the emission management and engine management electronic controls.
Considering automotive innovations in the future, it is forecast that 90% will be based on electronics development and 80% of that amount will be based on software development according to the conclusion of the automotive roundtables of McKinsey. Today, telematic and multimedia applications are on the forefront and mechanically actuated components are being replaces with electrically actuated ones. Accordingly, the transformation has occurred over the last two decades in the automotive industry from mechanically oriented companies to electronically, mechatronically, or software oriented companies which will be further accelerated in the future. The challenge lies in the drastically increasing complexity of the vehicle's electronic systems, which is a challenge for engineering, design and diagnosis. Due to more electronic functions such as assistance and comfort systems, an optimized energy management is required in a vehicle. This will require larger generators and batteries which will lead step by step to the hybrid drivetrain generation. One trend is for example the usage of the Intelligent Battery Sensor which increases the accuracy of the battery parameters and the availability of power.
Another trend is the X-by-Wire technology in the automotive industry. The X-by-Wire concept can be realized by replacing some of the conventional electrical mechanical or hydraulic connections. Advantages are lower weight, less space is required, easier service and in the long run lower cost. This concept leads to a closer networking and interfaces of functions. This step to a fully integrated electronics architecture puts high demands on safety, availability and defect tolerance. With this increasing system complexity the testability and the diagnostic ability have to be increased in order to balance the high cost pressure.

A last trend is the legislation of the after sale market which may change. For equitable competitions between services shops, the European commission can propose a regulation that obliges car manufacturers to save all the diagnostic data of a vehicle in a standard format on a storage device. Actually, the competition studies about independent and authorized repairers do not outlines the practical difficulties for independent repairers to access the diagnostic data, but this trend may change in the future.

At some point, automotive manufacturers will reach the limits for installing electronic components into a vehicle due to limited space in vehicles. Another limit is related to the testing process.

3 Conclusion

The automotive industry will also be in the future a driver for innovations and new technologies. The increasing individual demands and further changes of basic conditions will be a challenge for automotive manufacturers. In particular, in the field of management, the complexity of design, system

integration and the experience of the after sale should be used for an efficient and sutainable life cycle management.
Therefore, the new introduced framework is making a significant contribution by an attempt to integrate design and after-sale into a same workflow. Due to the long cycle of replacement of diagnostic tools for manufacturers, it is essential that the new diagnostic tools provide the availability to work with old methodologies like expert system in order to be compatible with the old process and documents of the authors. The bridge provided by the new modular framework allows the transition from expert system based diagnosis to model based diagnosis which handles better the strong amount of dependencies between components.

Researchers in artificial intelligence accord a high interest to the notion of causality, because a formalized definition of this notion will make it possible to develop more accurate self-learning algorithms. This thesis has provided a contribution in the analysis of causalities for the diagnosis by having a practical view of it. The business cases show that this new methodologies are feasible and can be applied to the automotive industry. Automotive manufacturer which are reacting flexibly and consistently to the described challenges will also provide highly qualitative and innovative products in the future.

Bibliography

[AAMA, 1999] AAMA,
New passenger car sales: 1970-97, American Automobile Manufacturers Association, Motor Vehicle Facts & Figures, 1998, Detroit, MI, Wards's Communications, personal communication, Apr. 7, 1999; Used passenger car sales: ADT Automotive, 1999 Used Car Market Report Nashville, TN; Leased passenger cars: CNW Marketing/Research, Bandon, OR, personal communication, Jan. 25, 1999.

[ACEA, 2005] ACEA,
European Automobile Industry Report. URL: http://www.acea.be/ASB20/axidownloads20s.nsf/Category0ACEA/0ED967FB0DA32BECC125704B0035F273/$File/IndustryReport05.pdf (Last consulting: 07/02/2007)
2005

[ADAC, 2006] ADAC,
Pannenstatistik 2005,
http://www.adac.de/images/Pannenstatistik%202005_tcm8-145338.pdf
(Last consulting: 12/11/2007)
2006

[Alcides, Schmidt, 2004] Alcides C., Schmidt G.,
Semantic search engines
In Proceedings Third International School and Symposium on Advanced Distributed Systems
Guadalajara, Mexico, January 2004, pp. 145-157

[Alonso et al., 2007] Alonso C.J., Prieto O.J., Rodriguez J.J., Bregon A.,
Stacking Dynamic Time Warping for the diagnosis of dynamic systems
In proceeding 12th Conference of the Spanish Association for artificial intelligence Salmanca, Spain, November 2007
ISBN: 3540752706

[Alpar, Niedereichholz, 2000] Alpar P., Niedereichholz J.,
Data mining im praktischen Einsatz
Vieweg Verlag, 2000
ISBN 3528057483

[Alpaydin, 2004] Alpaydin E.,
Introduction to Machine Learning
MIT Press, 2004
ISBN : 9780262012119

[Ao, 2008] Ao S.-L.,
Data Mining and Applications in Genomics
Lecture Notes in Electrical Engineering , Vol. 25, 2008,
ISBN: 9781402089749

[Asahiro et al., 2007] Asahiro Y., Miyano E., Miyazaki S., Yoshimuta T.,
Weighted Nearest Neighbor Algorithms for the graph Exploration Problem on Cycles
In proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science,
Harrachov, Czech Republic, January 2007, pp.164-174
ISBN: 9783540695066

[Azarian et al. 2008] Azarian A., Siadat A., Bauchat J.L.,
Domain specific Information retrieval System with a correspondence graph
Proceeding of the 2nd International Conference on Digital Information Management
vol. 1, Lyon, France, 28-31 October, 2007, pp.1-6
ISBN: 9781424414758

[Azarian, Siadat, 2007] Azarian A., Siadat A.,
A proposal of a practical approach for quantified quality software evaluation during the development cycle
Proceedings of the 7th WSEAS International conference on applied informatics & communications
Vouliagmeni, Greece, 24-26 August 2007, vol. 7, p. 331-356
ISBN: 9789608457966

[Azarian, Siadat, 2009] Azarian A., Siadat A.,
Domain specific Information retrieval system with a correspondence graph applied to car diagnosis
International Journal of Computational Intelligence Research
vol. 5 (Issue 1), 2009, pp. 41-51
ISSN: 09731873

[Backmeister et al., 2006] Backmeister A., Dogan B., Hecker H., Köhler H., Kolbe K., Öhlenschläger M., Schleicher R., Watzal W., Zweigler W.
ASAM MCD-2D (ODX) Version 2.1.0 - Data Model Specification" in ASAM specification, 2006 (http://www.asam.net).

[Bauer, 2003] Bauer H.,
Kraftfahrtechnisches Taschenbuch
Vieweg, Wiesbaden, 2003

[Baumeister, Seipel, 2002] Baumeister J., Seipel D.,
Diagnostic Reasoning with Multilevel Set–Covering Models
In Proceedings of the 13th International Workshop on Principles of Diagnosis (DX-02), Semmering, Austria, 2002

[BMW, 2008] BMW Group,
Pressemeldung: BMW Group steigert 2007 Absatz um 9,2 Prozent.
2008

[Borgelt, Kruse, 2005] Borgelt C., Kruse R.
Probabilistic Graphical Models for the diagnosis of analog electrical circuits.
In proceedings of the 8th European conference on Symbolic and quantitative approaches to reasoning with uncertainty
Barcelona, Spain, July 2005, pp. 100-102.

[Bortolazzi, 1996] Bortolazzi J., Hirth T., Raith T.
Specification and Design of electronic control units.
In Proceedings of the EURO-DAC 96
Geneva, Switzerland, 1996

[Bozon, 2005] Bozon L.,
Dealer groups in Europe: Their growth and strategies, ICDP Research Report 02/2005, Solihull, UK 2005.

[Braess, Steiffert, 2003] Braess H., Steiffert U.,
Kraftfahrzeugtechnik.
Vieweg, Wiesbaden, 2003

[Brandl, 2008] Brandl M.,
Article 10.01.2008 14:31 Kurier,
http://www.kurier.at/sportundmotor/motor/119596.php, (last consulting: 16/06/2008)
2008

[Breiman et al., 1984] Breiman L., Friedman J. H., Olshen R.A., Stone C.J.,
Classification and Regression Trees.
Wadsworth, Belmont, California, USA, 1984

[Brisson et al., 2006] Brisson L., Collard M., Pasquier N.,
Ontologies et base de connaissances pour le pré-traitement et le post-traitement en fouille de données in Fouille de données complexes workshop in EGC
conference
Lille, France, 2006

[Brisson, Collard, 2007] Brisson L., Collard M.,
Une expérience d'intégration des connaissances dans un processus de fouille de données in Rapport de recherche
2007

[Broy, 2003] Broy M.,
Automotive Software Engineering
In Proceedings of the 25th international conference on software engineering
Portland, Oregon, USA, 2003, pp. 719-720

[Buchaman, Shortliffe, 1985] Buchaman B.G., Shortliffe E.M.,
Rule Based Expert systems: The MYCIN Experiments of Standford Heuristic programming project.
Addison Wesley, May 1985

[Buntime, 1992] Buntime W.,
Learning Classification Trees
Statistics and Computation, Vol 2, 1992, pp. 63-73

[CEA, 2006] Press Kit,
Commissaire à l'énergie Atomique, 2006

[Chieux, Guillaneuf, 2005]. Chieux T., Guillaneuf C.,
The European aftermarket – trends in the market and pros-pects for authorised and independent repairers,
ICDP research report 05/05, Solihull, United Kingdom, 2005

[Console, Torasso, 1991] Console L., Torasso P.,
A Spectrum of Logical Definitions of Model-based Diagnosis.
Computational Intelligence, 1991, pp.133-141

[Cornelius, 2004] Cornelius T. L.,
Intelligent Knowledge-Based Systems,
vol. 3, 2004

[Cremmel et al. 2008] Cremmel A., Azarian A., Siadat A.,
Proposal of an ODX Data Exchange System Applied to Automotive Diagnosis
In Proceedings of the 3rd International Conferences on Cybernetics & Intelligent Systems (CIS)
and Robotics, Automation & Mechatronics (RAM)
Chengdu, China, 22-24 Sept. 2008, pp. 1275-1280
ISBN: 9781424416738

[Croft, 1983] Croft W.,
Experiments with representation in a document retrieval system
Research and development, 1983, pp. 1-21

[DaimlerChrysler, 2004] DaimlerChrysler,
New safety Report.
Corporate Headline, Stuttgart, Germany, 2004

[DAT, 2007] DAT,
DAT report 2007,Kfz. Betrieb, pp. 27

[Davis, 1989] Davis R.,
Expert Systems: How far can they go? Part one
AI Magazine, Vol. 10, No 2, 1989, pp. 61-67

[Everett, 2003] Everett R.,
Diffusion of innovation,
Free Press, 2003
ISBN: 0743222091

[Flämig, 1991] Flämig W.,
Grammatik des Deutschen,
Springer Verlag, Berlin, 1991

[Francois, Leray, 2004] François O., Leray P.,
Etude comparative d'algorithmes d'apprentissage de structure dans les réseaux bayésiens.
Journal électronique d'intelligence artificielle, 2004, pp. 1-19

[Frost, Sullivan, 2008] Frost and Sullivan corp.,
Electronics ca Publications. Automotive Semiconducters,
URL: www.global-electronics.net (last consultation: 13/05/2008)
2008

[Gessel, 1995] Gessel M.,
Counting Acyclic Diagraphs by sources and sinks
Brandeis University, January 1995

[Gessel, Sagan, 1996] Gessel M., Sagan B.,
The Tutte polynomial of a graph, depth-first search, and simplicial complex partitions. Electr. J. Comb., 1996

[Getoor, 2005] Getoor L., Diehl C.P.,
Link Mining: a survey in SIGKKD
Explorations 7.2., 2005, pp. 3-12

[Gigon et al. 2009] Gigon F., Azarian A., Siadat A., Seemann W.,
Combination of Heuristic and Model Based Diagnostic Methods Applied to Car Diagnosis
Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing, 3-5 June, Moscow, Russia, 2009

[Gorny, Ligeza, 2001] Gorny B., Ligeza A.,
Model-Based Diagnosis of dynamic systems: Systematic conflict generation
In Proceedings of the International Conference of Model-Based Reasoning: Scientific Discovery, Technological Innovation, Values (MBR'01),
Collegio Ghislieri, University of Pavia, Pavia, Italy, May 2001

[Grasso et al., 2007] Grasso M., Lavagna M., Sangiovanni G.,
Contextualized Possibilistic networks with temporal framework for knwoledge base reliability improvement
In Proceedings of the 7th International Workshop on Fuzzy Logic and Applications
Camogli, Italy, July 2007
ISBN: 35407339910

[Grefenstette, 2005] Grefenstette G.,
Explorations in Automatic Thesaurus Discovery
Kluwer Academic Publishers, 2005, pp. 33-39
ISBN: 9780792394686

[Hagiwara et al., 2005] Hagiwara M., Yasuhiro O., Toyama K.,
PLSI Utilization for automatic thesaurus contruction
In Proceedings Second International Joint Conference
Jeju Island, Korea, October 2005, pp. 334-336
ISBN: 978-3-540-29172-5

[Harman, 1993] Harman D.,
The First Text Retrieval Conference (TREC-1). National Institute of Standards and Technology, (NIST Spec. Publ. 500-207), Gaithersburg, 1993

[Hodge, 2000] Hodge G.,
NKOS Taxonomy of Knowledge Organization Sources/Systems,
URL: http://nkos.slis.kent.edu/KOS_taxonomy.htm (Last consulting: 21/08/2005)
2000

[IP/06/302, 2006] Rapid Press Release of European commission,
Competition: Commission welcomes changes to BMW's distribution and servicing agreements
IP/06/302, Published the 13/03/06
2006

[Jarmulak et al., 1997] Jarmulak J., Kerkhoffs E.J.H., Van't Veen P.P.,
CBR in ultrasonic Rail Inspection System. In: D.B. Leake and E. Plaza (eds).
In Proceedings of the 2nd International Conference on CBR.
Rhode Island, USA, July 1997, pp. 43-52

[Karanikas et al., 2000] Karanikas H., Tjotjis C., Theodoulidis B.,
An approach of text mining using information extraction
In Proceedings of principles and practice of knowledge discovery in databases
Lyon, France, 2000

[Kiencke, Nielsen, 2005] Kiencke U., Nielsen L.,
Automotive Control Systems: For Engine, Driveline, and Vehicle
2nd Edition, Springer, 2005, p. 150
ISBN: 3540231390

[Kleer, 1986] De Kleer J.,
An assumption based truth maintenance system.
Artificial Intelligence, Vol. 28, 1986, pp. 127-162

[Kleer, Reiter, 1992] de Kleer J., Reiter R.,
Characterizing diagnoses systems
Arificial Intelligence, 1992, pp. 197-222

[Kricke, 2005] Kricke K.,
"ODX, der neue Standard für Diagnose-Daten im Kraftfahrzeug" in Diagnose von E/E-Systemen im Automobil, München (Germany), 2005, pp. 1-27

[Krovetz, 1993] Krovetz R.,
Viewing Morphology as an Inference Process
In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval
Pittsburgh, PA, USA, 1993, pp. 191-203

[Krovetz, 1997] Krovetz R.,
omonymy and polysemy in information retrieval,
In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics
Madrid, Spain, 1997, p. 72

[Leen, Heffernan, 2002] Leen G., Hefferman D.,
Expanding Automotive Electronic Systems.
Computer, Vol.35, N. 1, 2002, pp.88-93

[Lefevre et al., 2004] Lefevre E., Manata J., Jolly D.,
Classification par la théorie de l'évidence pour la gestion de tournées de véhicules
In Proceedings of R.F.I.A.
Toulouse, France, 2004, pp. 1-9

[Lefevre, 2000] Lefevre P.,
La recherche d'informations,
Hermès, 2000, pp. 20-40

[Liao et al., 1999] Liao M., Abecker A., Bernardi A., Hinkelmann K., Sintek M.,
Ontologies for knowledge retrieval in organisational memories
In Proceedings of the Learning Software Organizations workshop
Kaiserslautern, Germany, 1999, pp. 19-26

[Liberation, 2004] Liberation,
Press article, URL: http://www.liberation.fr/page.php?Article=243530 (Last consultation: 03/02/2007)
2004

[Liebemann et al., 2004] Liebemann E., Meder K., Schuh J., Nenninger G.,
Safety and Performance Enhancement. Convergence Transportation Electronics Association
Detroit, USA, 2004

[London Economics, 2006] London Economics,
Developments in car retailing and after-sales markets under Regulation N° 1400/2002
- Volume I - Final report to EC DG Competition By London Economics
June 2006

[Loupy, 2000] Loupy (de) C.,
Evaluation de l'apport de connaissances linguistiques en désambiguïsation Sémantique en Recherche documentaire
Thèse de l'Université d'Avignon et des Pays de Vaucluse, Laboratoire informatique d'Avignon
URL : http://www.langnat.com/~loupy/papers/these.pdf (Last consulting : 18/10/2005)
2000, pp. 2:20-22,2:25-27, 4:47-48, 4:50-63, 9:136

[Manning, Raghavan, 2005] Manning C., Raghavan P.,
Text Retrieval and Mining
URL: http://www.stanford.edu/class/cs276b/handouts/lecture11.ppt (Last consultation: 18/10/2005)
2005 Cours CS276B, Standford

[Maron, Kuhn, 1960] Maron M.E., Kuhn J.L.,
On relevance, probabilistic indexing and information retrieval, Journal of the Association for Computational Machinery, Vol. 7 No. 3, 1960, pp. 216-244

[Martin, Plaza, 2004] Martin F.J., Plaza E.,
Ceaseless Case-Based Reasoning
In Proceedings of the 7th European Conference on Case-Based Reasoning, Madrid, Spain, August, 2004, pp.288
ISBN: 3540228829

[Masoud et al., 2005] Masoud N., Lotfi A. Z., Janusz K.,
Soft Computing for Information Processing and Analysis
Springer 2005, pp.393-397
ISBN: 978-3-540-22930-8

[Milde et al., 1999] Milde H., Hotz L., Kahl J., Neumann B., Wessel S.,
MAD: A Real World Application of Qualitative Model-Based Decision Tree Generation for Diagnosis
In Proceedings of the 12th International Conference on Industrial and Engineering Applications of Artifical Intelligence and Expert Systems, IEA/AEI-99
Cairo, Egypt, May 1999, pp.246, 253-254.

[Milde, Hotz, 2001] Milde H., Hotz L.,
Generating fault trees from mixed quantitative and qualitative electrical device models Laboratory for artificial intelligence - University of Hambourg
2001

[Mizzaro, 1997] Mizzaro S.,
Relevance: the whole history
Journal of the American Society for Information Science, Vol. 48, No. 9, 1997, pp. 810--832

[Moore, 2008] Moore A.W.,
Lecture on Decision trees
School of Computer Science
Carnegie Mellon University
URL: http://www.autonlab.org/tutorials/dtree.html (Last consultation: 20/06/2008)
Date unknown

[Nie, 2004] Nie J.-Y.,
Analyse et Indexation des documents et des requêtes
Cours IFT6255 (hivers) - University of Montréal
URL: http://www.iro.umontreal.ca/~nie/IFT6255/Indexation.html (Last consulting: 09/08/2005)
2004

[Olive et al., 2004] Olive X., Trave-Massuye S. L., Poulard H.,
Variantes de l'AO* pour générer automatiquement des arbres de diagnostic presque optimaux
In proceedings of the 14th congrès francophone de Reconnaissance des formes et intelligence artificielle
Toulouse, France, January 2004

[Parka, Sugumaran, 2004] Parka S., Sugumaran V.,
Designing multi-agent systems a framework and application
Expert systems with application, vol. 10, 2004, pp. 1-13

[Pavlicek et al. 2007] Pavlicek D., Pechoncek M., Marik V., Flek O.,
Multi-Agent based diagnostics of electronic subsystems
In Proceedings of the 3rd international conference on Industrial application of Holonic and Multi-Agent Systems
Regensburg, Germany, Sept. 2007, p. 372

[Peng, Reggia, 1990] Peng Y., Reggia J.A.,
Abductive inference models for diagnostic problem-solving
Springer, 1990

[Pierra et al., 2004] Pierra G., Dehainsala H., Ameur Y.A., Bellatreche L., Chochon J., Mimoune M. E.-H.,
Base de données à base ontologique : le modèle OntoDB
In Proceedings of Bases de données avancées
Montpellier, France, 2004

[Piwowarski, 2003] Piwowarski B.,
Techniques d'apprentissage pour le traitement d'informations structurées : application à la recherche d'information.
Thesis of the University of Paris VI, Spécialité Informatique, 2003, pp. 23-47

[Plant, Murell, 2007] Plant R., Murrell S.,
The Executive's Guide to Information Technology: Principles, Business Models, and Terminology
Cambridge University Press, 2007
ISBN: 0521853362

[Porter, 1980] Porter M. F.,
An algorithm for suffix stripping
Program, 1980, pp. 130-137

[Porter, 2005] Porter M. F.,
Porter stemming algorithm for different languages.
URL : http://www.snowball.tartarus.org (last consultation: 1/12/2005)
2005

[Price, 1999] Price J.C.,
Computer based diagnostic systems, Springer, 1999, pp.65-69

[Rényi, 1961] Rényi A.,
On measures of entropy and information
In Proceedings of the 4th Berkeley Symposium on Mathematical statistics and probability,
Berkeley univeristy, California, USA, 1960, pp. 547-561

[Ripplinger, 2006] Ripplinger T.,
System description,
Siemens AG - internal document (specification) - 2006

[Risjsbergen, 1979] Rijsbergen (van) C.J.,
Information Retrieval; Second Edition, Butherworths ; London, chapter 2-3, 1979

[Robertson et al., 1994] Robertson S., Walker S., Hancock-Beaulieu M., Gatford M.,
Okapi at TREC-3 ; Actes de Third Text Retrieval Conference (TREC-3),
NIST special publication, Gaithersburg, Maryland, USA, 1994, pp. 109-126

[Robin, 2003] Robin S.,
Lecture: "Diagnostic à base de modèle - Master 3"
DEA 2003-2004, pp. 24-25

[Ruml, 2002] Ruml W.,
Adaptive Tree Search.
Ph.D. thesis, Harvard University, May, 2002

[Salton, 1989] Salton G.,
Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer
Addison-Wesley, 1989

[Salton, Lesk, 1968] Salton G., Lesk M.,
Computer evaluation of indexing and text processing ; Journal of the Association for Computing Machinery, Vol. 15, No. 1, 1968, pp. 8-36

[Sandkar, Simon, 2004] Sankar K. P., Simon C.K. ,
Foundations of Soft Case-Based Reasoning
John Wiley & Sons Inc; Wiley Series on Intelligent Systems
2004
ISBN: 0471086355

[Schroeder, 1998] Schroeder M.,
Autonomous model based diagnostic agents.
Springer, 1998, pp. 12

[Sculley et al., 2006] Sculley D., Wachman G. M., Brodley C. E.,
Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers.
In Proceedings of the 15th Text Retrieval Conference
Gaithersburg, USA, 2006

[Shannon, 1948] Shannon C.,
A Mathematical Theory of Communication
The Bell System Technical Journal, Vol. 27, 1948, pp. 379-423,623-656

[Shukla, Tiwari, 2007] Shukla S.K., Tiwari M.K.,
Soft decision trees: A genetically optimized cluster oriented approach
Expert Systems with Applications, 2007

[Silver, 2000] Silver D. L.,
Selective Transfer of Neural Network Task Knowledge
Ph.D. thesis, University of Western Ontario, 2000

[Smith, Kandel, 1993]. Smith S., Kandel A.,
Editeur : CRC Press Inc (16 août 1993)
Langue : Anglais
ISBN: 08493890210

[Sparck-Jones, 1975] Sparck-Jones K., Rijsbergen (van) C.,
Report on the Need for and Provision of an "Ideal" Information Retrieval Test Collection
British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, 1975.

[Struss, 1997] Struss P.,
Model-based and qualitative reasoning: An introduction, Annals of Mathematics and Artificial Intelligence, Vol 19, 1997, pp. 355-381

[Struss, 2003] Struss P.,
Lecture: Wissensbasierte Systeme
Techniche Universität München, 2003

[Taylor, 2005] Taylor B.J.,
Methods And Procedures for the Verification And Validation of Artificial Neural Networks , Springer
ISBN: 0387282882

[Torasso, Console, 1989] Torasso P., Console L.,
Diagnostic problem solving
North oxford academic, 1989, p. 3

[Trevett, 2002] Trevett N. R.,
X-by-Wire, New Technologies for 42V Bus Automobile of the Future,
South Carolina Honors College, April 2002

[Tyler, 2007] Tyler A.R.,
Expert Systems Research trends,
Nova publishers, 2007, pp. 217
ISBN: 1600216889

[Ueno et al., 1992] Ueno H., Yamamoto Y., Fuduka H.,
Knowledge modeling based on object model – an approach to multi-use engineering knowledge base in Information Modelling and Knowledge Bases: Foundations, Theory and Applications: No. 3 (Frontiers in Artificial Intelligence and Applications, 13), 1992, pp. 75,76
ISBN: 9051990731

[VDI, 2003] VDI report 1794,
Innovative Occupant and Partner Crash Protection
VDI reports, Düsseldorf, Germany, 2003

[Weber, Weisbrod, 2002] Weber A., Weisbrod J.,
Requirements Engineering in Automotive Development - Experiences and Challenges. In Proceedings of the IEEE Joint International Conference on Requirements Engineering
University of Essen, Essen, Germany, 2002

[Wen et al., 2003] Wen Z., Crossman J., Cardillo J., Murphey Y.L.,
Case base reasoning in vehicle fault diagnosis
In IEEE proceedings of the International Joint Conference on Neural Networks (4), Portland, USA, 2003, pp. 2679-2684

[William, 2003] William R.,
Understanding automotive electronics,
Newnes, 6th Revised edition, 2003, pp. 10-20
ISBN: 0750675993

[Woolridge, Jernings, 1995] Woolridge M., Jennings N.R.,
Intelligent agents: theory and practice
Knowledge engineering review, 1995, pp. 115-152

[Zhao, Ouyang, 2007] Zhao X., Ouyang D.,
Improved Algorithms for deriving all minimal conflict sets in Model-Based Diagnosis
In Proceedings of the Third International Conference on Intelligent Computing Qingdao, China, August 2007

[Zimmermann, Schmidgall, 2006] Zimmermann W., Schmidgall R.,
Bussysteme in der Fahrzeugtechnik,
Friedr. Vieweg & Sohn Verlag, 2006.
ISBN : 9783834801666

Appendix A

This appendix contains additional material, which complements the issues described in the thesis. This document is divided into 3 different sections, which correspond to the 3 first chapters of the thesis: the first one reports about additional feature of SIDIS Enterprise, especially the incremental publication process. The second part tackles the workflow of the natural search engine procedures. The last section is related to chapter III of the thesis and thus concerns the model used for the simulation of the diagnosis session, as well as the case base and the detailed results for different values of the parameters. For the reason that some datasets are extremely large as the language resources, some tables only present samples.

1 Specific features of SIDIS Enterprise

1.1 Functional modules

Functional modules provide the user with core system functions. The functional modules in SIDIS Enterprise will be described in more detail in the current section. Examples are:

- Vehicle identification.
- Perceived symptom compilation.
- Vehicle system test.
- Basic diagnostics.
- Information search.
- Information display.

Further manufacturer-specific functional modules can be developed if required and included in the overall structure of the workshop system and/or in the applications manager.
A functional module generally has several **Views**. These correspond to a special kind of display for the functional module. For example, the vehicle identification module has 3 views:

- Identification via automatic read out of the VIN.
- Identification via manual input of the VIN.
- Identification via selection of vehicle characteristics.

Each of these views has its own user interface within the functional module with assigned command buttons. The operator can use these command buttons to control the desired partial function. The following table contains an extract from standard functional modules in SIDIS Enterprise. The arrangement and design of individual views as well as the assigned command buttons depend on the kind of user interface. Thus, the arrangement of views and commands may vary with the manufacturer.

Module name	View	Commands
Vehicle identification	by reading the VIN	Start
	by entering the VIN	Delete
		OK
	by selecting characteristics	Delete
		Reset
Perceived symptom compilation	by list selection	Add
		Delete
	by keyword search	Add
		Delete
Task management		Update list
		Open
		Delete

Module name	View	Commands
Task information	Task data	Update
		Note
	Vehicle data	Update
		Re-identify
	Service plan	Execute / Go to
		Delete
		move (forward)
		move (back)
	Diagnostics report	Displaying
		Print
		Delete
Test schedule		Update
		Run test
		Documentation
Vehicle system test	ECU list	Start
		Cancel
		ECU functions
		Documentation
	Fault list	Update
		Delete Fault memory
		Documentation
	Network view (manufacturer only, no standard view)	Start
		Cancel
		ECU functions
		Documentation
Basic diagnostics	ECU functions	Identification
		Diagnostic query
		Clear fault memory
		Single test
		Final control element control
		Documentation
	Program	Start
		Documentation
	Code	Start
		Documentation
	EOBD functions	Mode selection
		Start
Diagnosis end	Summary	Comment
		Print
	User feedback	Submit assessment
		Suggested improvement
Information search	by structure selection	Reset selection
		Displaying
		Include in service plan
	by search text	Start
		Displaying
		Include in service plan
Testing	Multimeter	Test channels
		Display options
		Stimuli
		Calibration

Module name	View	Commands
		Start/stop test
		Navigation in test reading memory
	Oscilloscope	Test channels
		Trigger
		Display options
		Set stimuli
		Calibration
		Start/stop test
		Measure test curve
		Navigation in test reading memory
		Test channel
		Trigger
		Display options
		Stimuli
		Calibration
		Start/stop test
		Set device parameters
		Allocate devices
		Start/stop stimulation
		Select V measurement
		Identify engine
		RPM signal
		Display options
		Calibration
		Start/stop test
		Measure
		Test value memory navigation
		Manage theoretical curves

Table A-1: Functional modules of SIDIS Enterprise

1.2 Linkage concept

A linkage concept supports the reuse of structures and content. This also allows service documents to be associated with vehicle data. Figure A-1 shows an example of the links between the vehicle configuration root and the vehicle characteristics (used for the vehicle identification).

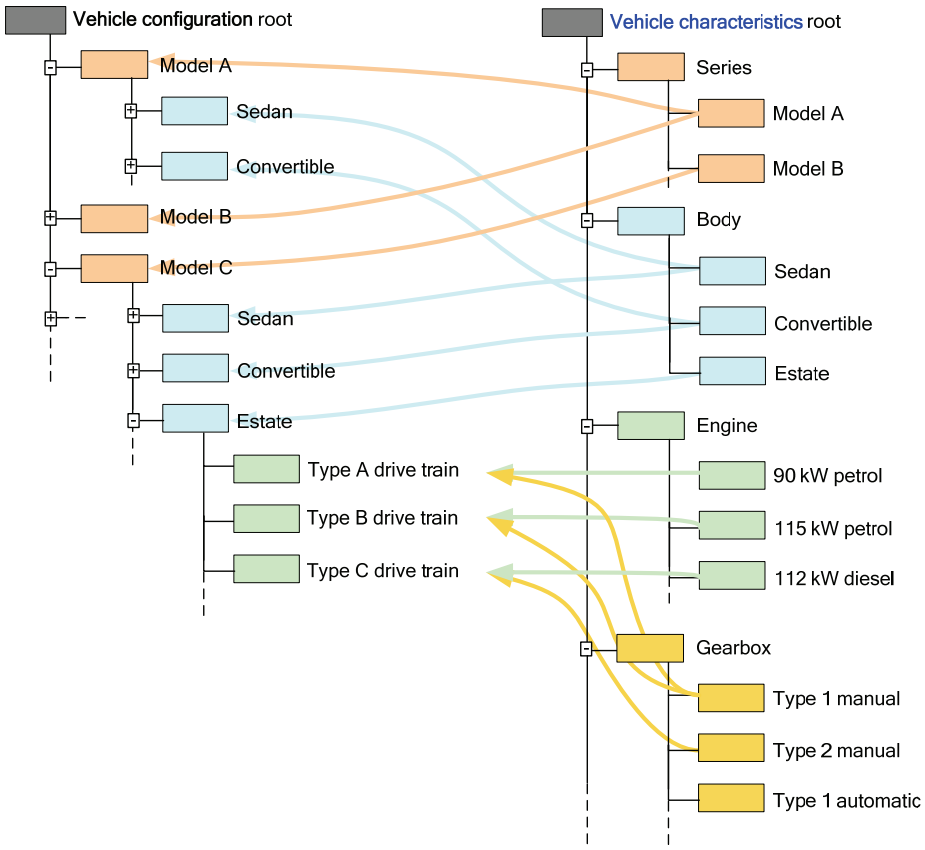

Figure A-1: Configuration tree

The links between the structures can be used to connect relevant objects like a component of the diagnostic object tree with a document in the document tree.

1.3 Publication or update

Publication is the selection of a diagnostics data directory compiled for a particular target group or for a specific purpose.
Generally, there is a differentiation between the types of publications depending on the purpose:

- Verification of diagnostics data generated for a particular workshop system client;
- Production final checks;
- Productive use in contract workshops;
- Productive use in independent workshops;
- Productive use in a particular country, etc.

Publication for verification on a workshop system test client may be incomplete because SIDIS Enterprise allows iterative testing. It is sufficient for the published version to contain a related data file, which is to be tested. For example, this might be the data (vehicle description, program and expert rules) for a particular vehicle function.
The publication trace file allows identification and correction of faults associated with the publication. Example: Erroneous repair document, missing service program, suspected link which does not refer to a diagnostic object, etc.
The publication basically consists of the definition and the execution steps described in the subsequent sections.

1.3.1 Define publication

Publication definition consists of the following component parts:

- Publication job

- Publication rules
- Publication filter
- Check routine
- Publication languages with replacement languages

All the information required for a publication are stored and/or linked in the publication job. Publication jobs are generally independent of each other. There is no administration of publication packets. The publication is modelled in a tree structure. The information in a publication job is described below:

- **Name** of the publication job : This name is used to identify the publication.
- Type **of publication:** The following publication types exist:
  - Update publication without info model
  - Update publication with info model
  - Publication with info model and previous deletion of CMS data

The updated publication builds on data already in the workshop system. These are either supplemented by new data or have their characteristics changed. In the case of a publication with previous deletion, all the data in the CMS database are deleted before the data are added to the workshop system. Thus, this publication always has an info model.

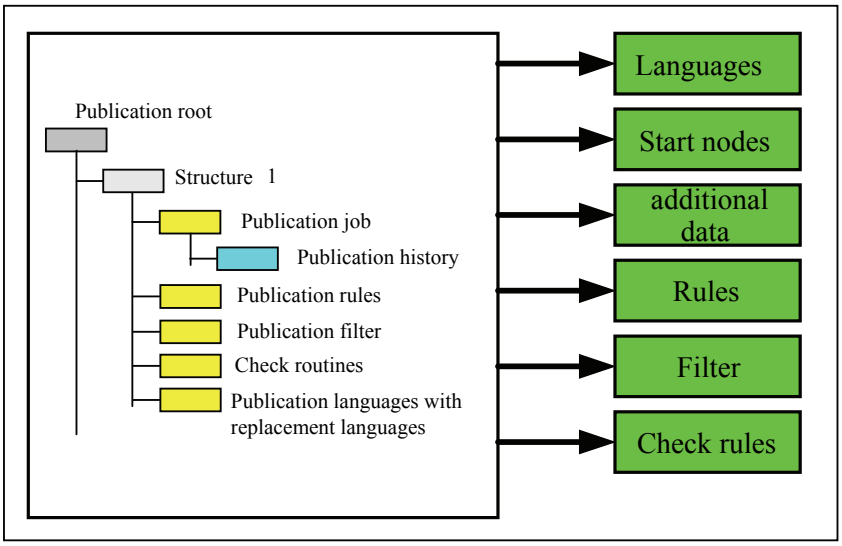

Figure A-2: Publication tree

A publication has the following characteristics:

- Type **of publication; the following types are possible:**
  - Test publication (does not require released tree structures)
  - Productive publication (released tree structures a prerequisite)

In the case of productive publications only the data released for the publication are published. The publication type is taken into account in the publication rules.

- **Change start date**: Only those data are published whose change date is after this date.
- **Languages**: The languages, which are to be published in this publication are defined. The replacement languages are also defined there. A publication job should only be linked to one language definition.
- **Start nodes:** The linked nodes form the starting point for evaluation of publication rules. Multiple start nodes can be linked to a publication job.
- **Additional files:** The contents of linked nodes are published as files in the publication packet, i.e. they are stored on the workshop system and do not go to the CMS. Multiple additional files can be linked to a publication job. Links are created by drag and drop.
- **Rules:** The methods for determining the publication extent are defined by the link with the publication rules. Multiple publication rules can be linked to a publication job.

- **Filters:** The nodes found using the publication rules are filtered again by the link with the publication filters. Multiple publication filters can be linked to a publication job.
- **Check rules:** The tests of the consistency of the data are determined by the link with the check rules. Multiple check rules can be linked to a publication job.
- **Test status** of the publication: This is an info field for the diagnostics author.

1.3.2 Publishing

The diagnostics author can start the currently selected publication job in the publication tree. When starting a publication job, he can also enter as a filter the date from which changes should be searched (change start date). The publication starts after the date has been entered. It runs asynchronously to the client, i.e. the diagnostics author can continue working on the client. Messages can be displayed so long as the clients are logged in. The diagnostics author can always determine from the publication history how far the publication has been executed. This history also indicates the status of publications, which the author has not started himself.
At the end of the publication, the published data are located in a specified directory. A report provides information on the settings used and any errors which have occurred.

1.3.3 Complete verification of diagnostic data

All diagnostic data, especially service programs, must undergo validation on a workshop client set up specially for testing before they are supplied to the workshop users. Workshop verification is a systematic and final test of the diagnostic data with the support of the data's editor, aimed at resolving any problems. This workshop verification step can be carried out in parallel, depending on the progress of editing the diagnostic data and the publication. The SIDIS Enterprise system allows problem-free return to previous steps in order to ensure incremental production of diagnostic data.
Integration tests in accordance with the testing and integration schedule are carried out for all diagnostic data during workshop verification. This overall testing is carried out in a test environment which, like the productive system, is based on a central workshop server and one or more workshop clients. The focus of this testing (unlike verification during data generation) is on testing the overall behaviour of the diagnostic data.

1.3.4 Data distribution

Data distribution or transmission means the provision of published diagnostic data for the end devices (workshop clients) in workshops. Transmission is selective and according to publication criteria, e.g. workshop user rights, country, brand, etc.

1.4 Details about the diagnosis algorithm

The diagnostic algorithm of SIDIS Enterprise takes into account the attributes in Table A-2 associated to nodes and links:

Attribute	Notation	Default Value
Number of suspicion rules from a boolean or fuzzy rule	*NSL*	
Weight of a boolean rule	*WR*	100%
Weight of a fuzzy rule	*WZ*	100%
Number of premisse of a specified	$NP(R_i, x)$	

operator *x* (here of type AND, OR or fuzzt Z)		
Rank of a suspicion rule	*RSL*	
Weight of symptoms	*WS*	100%
CBR case weight	CBR_i	50%
Number of CBR answers	*NV*(*CBR*)	

Table A-2: Attribute list and default values

There is a distinction between two types of inference rules: the expert or suspicion rules that point directly from a symptom (default's code or perceived symptom) to a diagnostic object and a boolean or a fuzzy rule. This last type of rule is composed of premises issued from symptoms that are pointing to an operator (boolean or fuzzy), the operator is linked with the diagnostic object. Operators can be and, or, not for boolean's type and for fuzzy types it can be e.g. "Comprised between 10.000 and 20.000 km". The suspicion link from the operator is only initiated if the operator returns the value true. Both types of rules are depicted in Figure A-3.

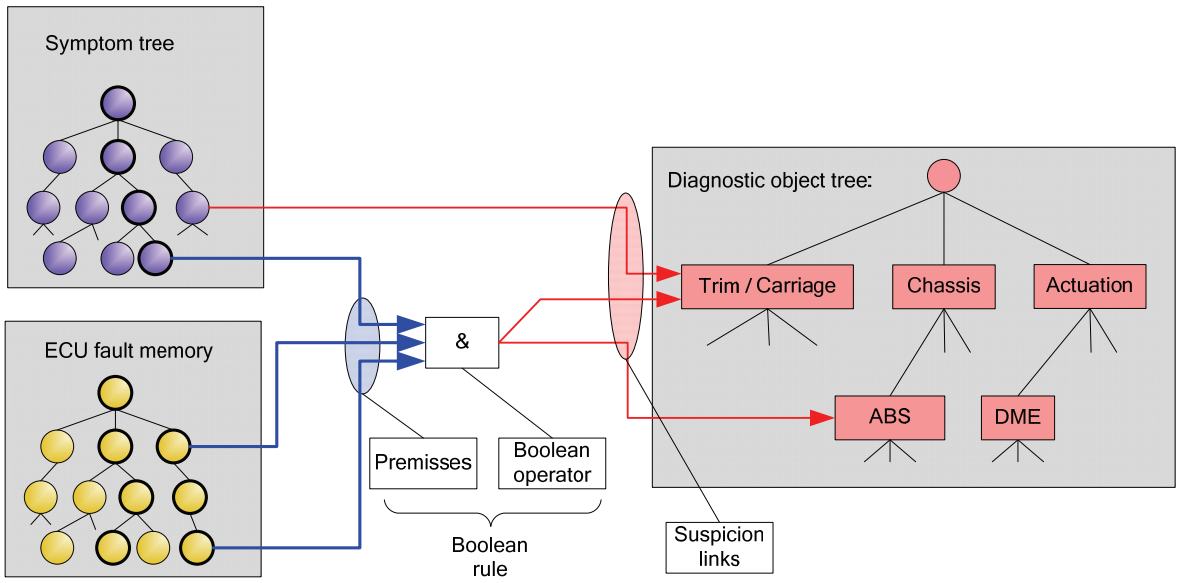

Figure A-3: Types of inference rules

Activation of the appropriate premise PR_i and rule evaluation:

- Boolean (Eq. A-1) or Fuzzy (Eq. A-2) -with *Z* an assignment function- rule are initiated if

$$Ri = PR1 \,\&\, PR2 \,\&\, \ldots. \,\&\, PRn$$

(Eq. A-1)

$$Min\_Limit < \sum_i Z(PR_i, PR_i = true)) < Max\_Limit$$

(Eq. A-2)

- If rules are initiated. The evaluation of the partial moment $PM(R_i,DO_x)$ for the rule R_i depends of the type of the rule (boolean or fuzzy), the number of premises $NP(R_i)$ of type AND or OR, the total number of suspicion links $NSL(R_i)$ which points to diagnostic objects after the operator, the weight of the rule *WR* and the rank of the outgoing suspicion rule (after the boolean or fuzzy operator) *RSL*:
  - Boolean rule: digressive in (Eq. A-3) and equi-partition in (Eq. A-4):

$$PM\left(Ri,DOx\right)=WR.\frac{(2^{NPR(Ri,AND)-1}+NPR(Ri,OR))*RSL_x}{\sum_{j=1}^{j=NSL(Ri)}RSL_j}$$

(Eq. A-3)

$$PM\left(Ri,DOx\right)=WR.\frac{2^{NPR(Ri,AND)-1}+NPR_i(OR)}{NSL(Ri)}$$

(Eq. A-4)

- Fuzzy rule: digressive in (Eq. A-5) and equipartition in (Eq. A-6):

$$PM\left(Ri,DOx\right)=WZ(R_i).\frac{RSL_x}{\sum_{k=1}^{k=NSL(Ri)}RSL_k}$$

(Eq. A-5)

$$PM\left(Ri,DOx\right)=\frac{WZ}{NSL(Ri)}$$

(Eq. A-6)

- If an external CBR system is implemented in SIDIS Enterprise, the diagnostic algorithm computes a suspicion degree for each returned case CBR_i. This evaluation depends of the number of suspected cases $NV(CBR_i)$ and the number of acute symptom suspicions $WS(DO_x)$ for each suspected diagnostic object DO_x in (Eq. A-7):

$$PM(CBRi,DOx)=WS-\frac{CBR}{NV(CBRi)}$$

(Eq. A-7)

- The CBR weight default's value is equal to 50%, which is lower than the usual default weight because of their fuzziness regarding to knowledge based on suspicion generation.

The diagnosis algorithm of SIDIS Enterprise assigns a suspicion degree of each component, the more incoming links point to a diagnostic object, the more chance that his suspicion moment is higher. Once the expert rules are evaluated, the suspicion is transmitted to the child nodes by the inheritance mechanism. In the system there are 4 main weight parameters to configure (symptom, diagnostic object, hierarchical links and suspicion link weights). In order to affine the fault localization, the deviation over these parameters needs to be maximized to empower the discrimination. For one vehicle with 40 electronic equipments that represents around 8000 diagnostic object nodes, 6000 default codes, 1000 perceived symptoms. Regarding the algorithm there is around 18000 weight coefficients to configure in the model. This task is nearly impossible for a human operator. Moreover, for a total coverage of the diagnostic algorithm the number of rules equals the given value in (Eq. A-8)

$$2^{m+n}-2^k$$

(Eq. A-8)

Independent expert rules need to be configured with n number of components of the system, m number of symptoms and k number of hierarchy links.

2 Symptom search engine

2.1 Linguistic resources

2.1.1 Stemming resources

The Table A-3 and Table A-4 list some samples of prefixes and suffixes, which are cut from the stemming algorithm which was implemented in the prototype.

Prefixes					
French	**German**	**English**	**French**	**German**	**English**
anti	an	a			il
con	au	ab			im
de	auf	ac			in
par	be	acr			inrta
pre	bei	aden			inter
re	ent	aero			ir
	ge	agro			meso
	im	amt			meta
	in	an			mid
	um	ana			non
	un	anar			out
	ver	anem			over
	ver	anglo			post
	vor	ante			pre
	zer	anthrop			pro
	zu	aut			re
		bar			rect
		bathy			sub
		dis			super
		dys			sym
		eco			trans
		eo			ultra
		ex			un
		fore			uni
		hyper			ur
		hypo			

Table A-3: Sample of the prefixes for stemming

French	German	English
a	dlich	abli
able	dlig	ablies
ables	e	ably
ai	em	alism
aient	en	aliti

ais	end	alities
ait	ern	ality
ance	ern	anci
ances	es	ationnal
ant	est	ationnal
ante	heit	ator
antes	ig	ators
ants	ig	biliti
as	ik	bilities
asse	isch	bility
assent	keit	bli
asses	lich	blies
assiez	lich	bly
assions	lig	ed
assions	llich	edly
ateur	llig	eed
ateurs	s	eedly
ation	st	enci
ations	tif	entli
atrice	ung	entlies

Table A-4: Sample of the suffixes for stemming

2.1.2 Stop lists

The Table A-5 contains a sample of the list of stop-words, which were filtered by the perceived symptom search engine.

French	German	English
a	ab	a
celles	darauf	as
dessus	die	corp
jusque	eurem	his
ne	Ihrer	more
quelle	meiner	of
tant	seit	says
afin	vom	about
celui	zur	at
dès	aber	could
la	daraus	if
ni	dies	most
quelqu'un	euren	on
te	ihrer	she
ailleurs	meines	after
cependant	seitdem	be
donc	von	for
laquelle	zwar	in
non	allein	mr
quelqu'une	darin	one
telle	diese	some

ainsi	eurer	all
certain	ihres	because
donné	mich	from
là	selbst	inc
nos	vor	mrs
quelque	zwischen	or
telles	als	also
alors	darüber	been
certaine	dieselbe	had
dont	eures	into
le	Ihres	ms
notamment	mir	other
quelques-unes	sich	an
tes	während	but
après	zwischens	has
certaines	also	is
du	darum	mz
	ihren	
	meinem	
	seiner	
	unsrer	
	zu	
	daran	
	dich	
	eure	
	Ihren	
	meinen	
	seines	
	unsres	
	zum	
	so	

Table A-5: Sample of stop-words

2.1.3 Irregular verb resources

The Table A-6 contains an extract of the list of irregular verbs, which are replaced by their infinitive form if they are encountered by the perceived symptom search engine. Due to the size of the table, only a sample was presented as an example.

German		
brechen	brach	gebrochen
geben	gab	gegeben
laufen	lief	gelaufen
nehmen	nahm	genommen
schleifen	schliff	geschliffen
schließen	schloß	geschlossen

stoßen	stieß	gestoßen
bringen	brachte	gebracht
erkennen	erkannte	erkannt
English		
become	became	become
break	broke	broken
come	came	come
French		
être	suis	es
avoir	ai	as

Table A-6: Extract of irregular verbs

2.1.4 Dictionaries

For the specific German language processing step, which consists in cutting long words and trying to identify 2 words within a dictionary, the following resource file had been added as resource into the prototype project. This file is a German dictionary composed by:

- A German dictionary from the Netspell project version 2.1.7 (licensed under GPL 2nd version).
- The German automotive dictionary made by AUDI.

Sample dictionnary	
29	Abbaufront
30	Abbaugerät
31	Abbaugeräusche
32	Abbauleistung
33	Abbauleitzentrale
34	Abbaumaschine
35	Abbaumaterial
36	Abbaumenge
37	Abbaumöglichkeit
71	abbremsen
72	Abbremsens
73	Abbremsgeschwindigkeit
74	Abbremsung
75	Abbremsverhalten
76	abbrennen
32551	Heizanlagen
32552	heizen
32553	Heizens
32554	Heizer
32555	Heizfläche
32556	Heizkessel
32557	Heizkissen
32558	Heizkörper
32559	Heizlüfter
32560	Heizmaterial

Table A-7: Sample of the German dictionnary entries

Table A-7 represents a sample of the 80.000 entries of the dictionary.

2.1.5 Thesaurus

Table A-8 and Table A-9 contain samples of the entries in the German and the French thesaurus, which were implemented in the prototype.

Sample of the German thesaurus				
Scheinwerfer	Beleuchtung			
Verbrennung	Kraftstoff	Kohlendioxid	Stickstoffgase	Wasserdampf
Abgasreinigung	Abgasrückführung	Abgasverhalten		
kühlen	erkalten	kalt	kälter	kühl
kuhlung				
Anti-Blockier-System	Blockieren der Räder	Bremsens lenkbar	Spur behalten	
Set up	aerodynamischen Funktionen			
Gesamtkraft				
Vorderradaushängung	Achsschenkelbolzen	Vorderrades	Aufhängung	Drehbarkeit
Aktive Geschwindigkeitsregel	Serienreife	Assistenzsystem	Schnellstraßen	Geschwindigkeits-regelung
vibration	vibriert	erholen	dämpfung	zögern
Getriebesteuerung	Gangwahl			
Fahrer	Beifahrer	Seitenairbags	schützen	Unfall
Luft	Heizung	Klimaanlage	kalt	warm
Lärm	Motor	Motorlärm	anlasser	Motorstart

Table A-8: Sample of the thesaurus in German

Sample of the French thesaurus					
essuies-vitres	Essuyage	essuis-vitres	ballais d'essuis-glace	essui-glace	
commandes	boutons	bouton	commande	ordre	pilotage
arrêt	stop	arrêts	stops	arrêter	immobilisation
vitesse	vitesses	rapidité	intensité		
intermittence	intermittent	irrégulier	discontinu	sporadique	variable
essuyage	lessivage	nettoyage			
givre	dégivrer	gel	froid	gelée	
projecteurs	phare	réflecteur	Eclairage	spot	feux
rétroviseurs	rétro	miroir			
allumage	démarrrage	départ	allumement	briller	
automatique	auto	machinal	mécanique	électronique	
position	situation	état	condition	attitude	
permanence	continuité	stabilité	constance	durabilité	immutabilité
brouillard	brume	brumaille			
clignotant	Indicateurs de directions				
communication	transmission	échange	relation	information	

Table A-9: Sample of the thesaurus in French

2.2 Search engine algorithm

2.2.1 Symptom collection

2.2.1.1 List of symptoms

Table A-10 contains an extract of the list of perceived symptoms with a unique identifier, which is used to precise the parent and the child of each symptom. The list used for the experiences contains more than 2000 entries, here only a few entries are given as example.

Sample of perceived symptoms in English			
ID	**EN Title**	**Level**	**Parent**
0	Symptoms	0	0
1	Lightings and Street signs	1	0
2	Interior lighting	2	1
3	ceiling lights by hand drive operation	3	2
4	ceiling lights by hand drive operation	4	3
5	ceiling lights by hand drive operation : do not function at all	5	4
6	ceiling lights by hand drive operation : do not function correctly	5	4
7	front ceiling lights by hand drive operation	4	3
8	front ceiling lights by hand drive operation : do not function at all	5	7
9	front ceiling lights by hand drive operation : do not function correctly	5	7
10	rear ceiling lights by hand drive operation	4	3
11	rear ceiling lights by hand drive operation : do not function at all	5	10
12	rear ceiling lights by hand drive operation : do not function correctly	5	10
13	central ceiling lights by hand drive operation	4	3
14	central ceiling lights by hand drive operation : do not function at all	5	13
15	central ceiling lights by hand drive operation : do not function correctly	5	13
16	ceiling lights temporized by opening carries	3	2
17	ceiling lights temporized by opening carries: do not function at all	4	16
18	ceiling lights temporized by opening carries: do not function correctly	4	16
19	interior lightings of comfort	3	2
20	interior lightings of comfort	4	19
21	interior lightings of comfort : do not function correctly	5	20
22	lighting cases	4	19
23	lighting cases: do not ignite at all	5	22

Table A-10: List of symptoms

2.2.1.2 List of symptoms for the correspondence graph

Table A-11 represents the list of symptoms, which were used for the correspondence graph search engine. Table A-12 and Table A-13 represent the concept list and the weight matrix of the graph in German language. In Table A-13, the empty cells stand for a weight coefficient equal to zero.

n°	Symptoms
1	Klimatisierung knöpfe funktionieren nicht
2	Klimatisierungsbildschirm funkelt
3	Kontrolle der Klimatisierung funktioniert nicht
4	Die Luftausgabe der Klimaanlage funktioniert nicht
5	Luftstrom der Klimaanlage ist nicht regulierbar

6	Der Luftstrom ist maximal
7	Der Luftstrom der Heizung / Klimaanlage ist schwach
8	Klimatisierung funktioniert zeitweilig
10	der Luftstrom funktioniert nicht an der Scheibe
11	Keinen Luftstrom der Klimatisierung / Heizung an den Füßen
12	Luftstrom der Heizung Klimatisierung nur an den Füßen
13	Scheibenwischer reinigen automatisch
14	Scheibenwischer funktionieren nicht auf höchster Stufe (maximale Einstellung)
15	Vorderscheibenwischer funktionieren nicht
16	Die Scheibenwischer halten auf der Mitte der Scheibe (kein ganzer Zyklus)
17	Das Reinigen der Hinterscheibe funktioniert nicht
18	die Heizung / Klimatisierung pfeift
19	Bei niedriger Einstellung pfeift die Klimatisierung / Heizung
20	Temperatur der Heizung / Klimatisierung ist zu warm / heiß
21	Temperatur der Heizung / Klimatisierung wird nicht kälter
22	Reinigungsmittel kommt im Auto hinein bei Reinigung der Vorderscheibe
23	Die Vorderscheibenwischer halten nicht
24	Die Geschwindigkeit der Scheibenwischer vermindert sich bei hoher Geschwindigkeit
25	Seit Wartungsphase funktionieren die Scheibenwischer nicht

Table A-11: Symptoms for the graph

n°	Concept
1	Klimatisierung Klimaanlage Heizung
2	Knöpfe Kontrolle Einstellungen Einstellung einstellen eingestellt
3	Bildschirm Screen Bordcomputer Anzeige Klimatisierungsbildschirm Control panel
4	Luftausgtabe Luftstrom blasen bläst Ventilation Strom
5	funkelt Blitzt
6	Regelung regulierbar Einstellbar Stufe
7	maximal max groß hoch maximaler größer großer höchster hoch
8	schwach niedrig niedriger minimal klein min Minimum
9	funktioniert funktionieren gehen geht
10	Zeitweilig Weile Zeit Moment
11	Scheibe oben
12	warm heiß wärmer warmer heißer
13	kalt kühl kühler kälter kalt
14	Füßen fuß unten
15	nur
16	kein keiner
17	Scheibenwischer wischen Wischer
18	reinigen rein Reinigung sauber Reinigungsmittel
19	Geräusch krach Lärm pfeifen pfeift hören laut
20	automatisch automatisiert
21	Geschwindigkeit

22	Vorderscheibenwischer vorne
23	halten halt stoppen stoppt
24	Wartung Wartungsphase Austausch
25	vermindern vermindert kleiner geringer
26	größer vergrößert vergrößern höher

Table A-12: Concept List for the graph

	Symptoms																							
Concepts	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	1	1	1	1	1	1	1	1	1	1						1	1	1	1				
2	1		1																					
3		1																						
4				1	1	1	1	0.3	1	1	1						1							
5		1																						
6					1								0.5					0.5						
7						1							1						0.5				1	
8							1			1								1					1	
9	1		1	1					1	1			1			1								1
10								1																
11									1															
12																			1	0.5				
13																			0.5	1				
14										1	1													
15											1													
16										1														
17												1	1	1	1	1						1	1	1
18												1				1					1			
19																	1	1						
20												1										0.5		
21													1										1	
22												0.6	0.6	1		0.2					0.6	1		1
23								1							1							1		1
24																								1
25																							1	
26																								

Table A-13: Weight matrix for the graph

2.2.1.3 Test request

Table A-14 and Table A-15 contain the request in natural language used for the index search engine and for the correspondence graph approach.

Test n°	Request	Expected answers
1	Problème de la commande manuelle des plafonniers	8
2	Eclairage du coffre intérieur ne s'allume pas	3
3	problème d'éclairage de la boîte à gant	3
4	problème avec le lecteur de carte avant	3
5	éclairage miroir ne fonctionne pas du tout	3
6	spot de lecture reste allumée sans arrêt	3
7	commandes intérieures des éclairages ne fonctionnent pas	3
8	commande d'éclairage des rétroviseurs ne fonctionnent pas	3
9	feux de positions ne s'allument pas	4
10	feux de croisement ne s'allume pas sur commande	2
11	problèmes feux de routes ne s'allume pas	3
12	problème avec les feux antibrouillard avant	2
13	témoins des clignotants ne fonctionne pas	1
14	alarme oublis des feux est toujours activée	1
15	appels des phares ne fonctionne pas	2
16	problème témoin feux de route	4
17	Antidémarrage ne fonctionne pas	2
18	mal fonctionnement alarme d'oubli de clé de contact	3
19	sirène de détection intrusion ne s'arrête pas	4
20	problème ouverture et fermeture par clé	2
21	problème avec le verrouillage intérieur	2
22	rabattement auto des rétroviseurs ne fonctionne pas	2
23	affichage état de verrouillage du véhicule ne fonctionne pas	5
24	détection des obstacles de l'aide au stationnement ne fonctionne pas	3
25	Témoin marche arrière	4
26	détection par télé caméra ne fonctionne pas	2
27	problème avec le détecteur d'angle mort	2
28	problèmes liés à l'affichage des données de trafic	5
29	Localisation sonore des données de trafic ne fonctionne pas correctement	7
30	alerte pile roue usée fonctionne sans arrêt	4
31	alerte crevaison clignote sans arrêt	4
32	message de sous gonflage illisible	1
33	affichage multifonction de la vitesse moyenne ne fonctionne pas	5
34	température extérieure erronée	5
35	affichage température d'huile irréaliste	5
36	affichage constant de défaut de suspension	1
37	alarme oubli ceinture arrière ne fonctionne pas	1
38	programme neige BVA	6
39	alerte de niveau liquide de frein	4
40	alerte pression d'huile toujours affichée	1
41	essuie vitre arrière ne fonctionne pas en mode automatique	4
42	réglage manuel des rétroviseurs ne fonctionnent pas	3
43	déploiement automatique des rétroviseurs ne fonctionne pas du tout	2
44	dégivrage arrière ne s'arrête pas	4
45	impossible de neutralise la survitesse	2

46	manque d'efficacité ABS	1
47	freinage véhicule trop long	1
48	problème avec la direction assistée	7
49	alerte franchissement involontaire de ligne	2
50	problème avec le chauffage des sièges	4

Table A-14: Request in natural language for the index search engine

Test n°	Request	Expected answers
1	Die Luft der Heizung kommt nicht raus	4
2	Es gibt ein Problem mit der Geschwindigkeit der Scheibenwischer	2
3	Die Klimatisierung pfeift	2
4	Ein Problem der Anzeige des Control panel der Klimatisierung ist aufgetaucht	2
5	Die Temperatur der Klimatisierung ist nicht reguliert	2
6	Die Ventilation hört systematisch auf	1
7	Die Betätigung der Controls der Klimaanlage sind ohne Wirkung.	1
8	Die Luftströmung durch der Heizung scheint gestört zu sein	2
9	Die Luft der Klimaanlage ist immer zu kalt	1
10	Die Scheibenwischer beenden nie den ganzen Zyklus	1

Table A-15: Request in natural language for the graph search engine

2.2.2 Algorithmic Complexity

Table A-16 details the notation used to determine the complexity of the index search engine with the different weight formulae.

M	Number of symptoms
N	Dimension of the base of terms
E	Number of entries in the thesaurus
e(i)	Number of synonyms in the thesaurus for the i^{th} entry.
q	Number of relevant word in the request
f(i)	Number of relevant words in symptom n°*i*
er	% of non null values of a matrix
g(i)	Number of synonyms in the thesaurus for the i^{th} word of the request

Table A-16: Parameter of the algorithms

In order to simplify the calculation of the complexity, the parameters listed in Table A-17 are replaced by their average value based on the symptom lists, which contain more than 2000 entries.

E	66
$E\sum_{E} e(i)$	336
Q	5
g(i)	0,734
$\overline{f(i)}$	17
er	1.60%

Table A-17: Average values of certain parameters

2.2.2.1 The index search engines

The overall procedure of the index search engine is depicted in Figure A-4.

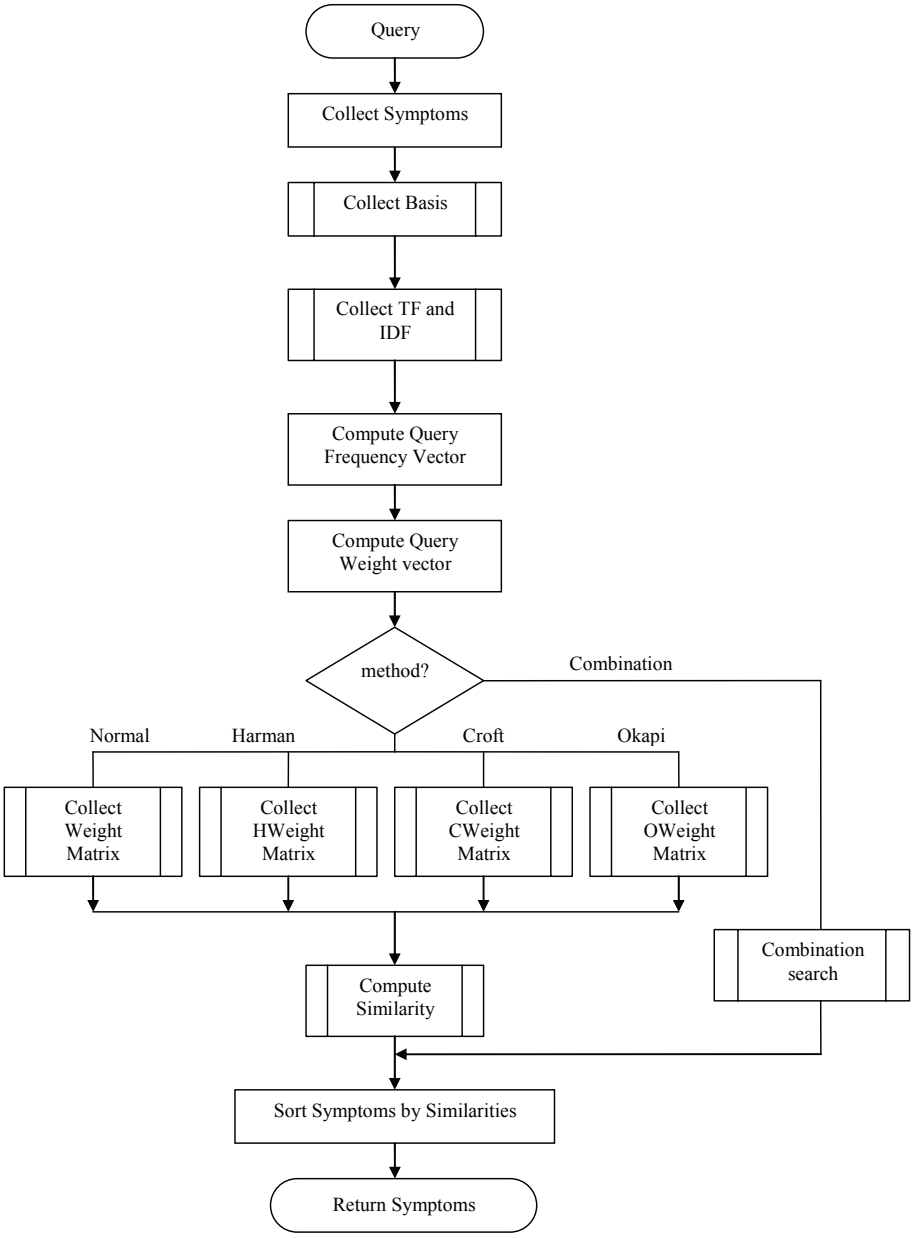

Figure A-4: Procedure of index search engine algorithms

The Table A-18 gives the complexity of each step of the procedure.

		Complexity and time				
		TF.IDF	**Harman**	**Croft**	**Okapi**	**Average time in ms**
Collection of the symptoms	Number of operations	*2M*				**20**
Collection of the base of	Number of operations	*N*				90

terms	Approximation	N				
TF	Number of operations	$erMN$	220			
	Approximation	$\approx 0,016\ MN$				
IDF	Number of operations	N	100			
	Approximation	N				
thesaurus	Number of operations	$q(E\sum_E e(i) + \sum_K g(i))$	520			
	Approximation	1683,67				
queryfreq	Number of operations	qN	50			
	Approximation	$5N$				
queryweight	Number of operations	N	50			
weight matrix	Number of operations	$erMN$	643	653	833	613
	Approximation	$0,016MN$				
compute similarity	Number of operations	NM	100			
Total	Number of operations	αMN	1793	1803	1983	1763
	Approximation	$1,032MN+2M+7N+1683,67$				

Table A-18: Algorithmic complexity

2.2.2.2 The search with the correspondence graph

The fundamental parameters of this method is the number of cells in the first layer C and the number of symptoms S. Let q be the number of relevant words of the request. The complexity of the algorithm is clearly $o(CS)$, which is the most important factor. The main steps of the algorithm are represented in Figure A-5.

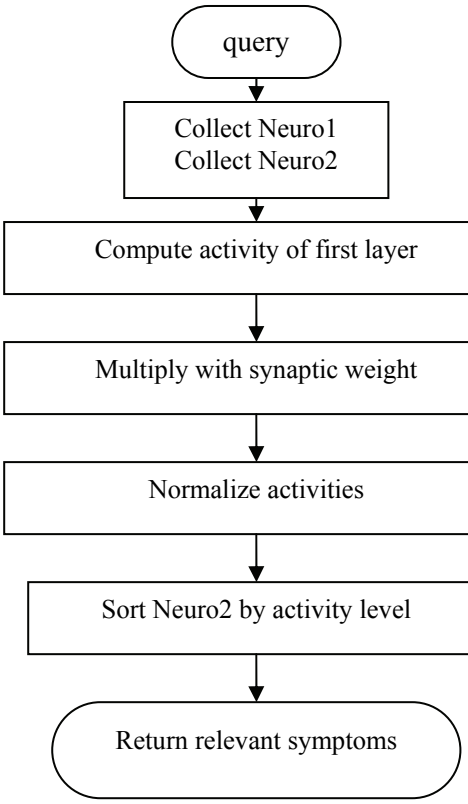

Figure A-5: Main steps of the search with the correspondence graph

3 Diagnosis simulation

3.1 Description of the model

The model used in chapter III from section 1.1 to 2.2 is described in this paragraph. The diagnostic object tree is given in Table A-19:

Node ID	EN Title	Incoming suspicion rules	Life time	Number of Childs	Is Leaf	Parent node ID	Child node ID
0	Diagnostic Objects	0	12000	3	FALSE	0	1;2;3
1	node 2	0	130000	3	FALSE	0	4;5;6
2	node 3	1	140000	3	FALSE	0	14;15;16
3	node 4	1	150000	2	FALSE	0	23;24
4	Node 5	1	100000	2	FALSE	1	7;8
5	node 6	4	80000	2	FALSE	1	9;10
6	node 7	4	70000	3	FALSE	1	11;12;13
7	node 13	0	60000	0	TRUE	4	
8	node 14	0	20000	0	TRUE	4	
9	node 15	0	30000	0	TRUE	5	
10	node 16	0	40000	0	TRUE	5	
11	node 17	1	50000	0	TRUE	6	
12	node 18	1	60000	0	TRUE	6	
13	node 19	0	45000	0	TRUE	6	
14	node 8	4	27000	2	FALSE	2	17;18
15	node 9	5	280000	2	FALSE	2	19;20

16	node 10	3	32000	2	FALSE	2	21;22
17	node 20	0	34000	0	TRUE	14	
18	node 21	0	25000	0	TRUE	14	
19	node 22	0	26000	0	TRUE	15	
20	node 23	0	18000	0	TRUE	15	
21	node 24	0	16000	0	TRUE	16	
22	node 25	0	120000	0	TRUE	16	
23	node 11	1	9000	2	FALSE	3	25;26
24	node 12	2	8542	3	FALSE	3	27;28;29
25	node 26	0	26000	0	FALSE	23	27
26	node 27	0	36000	0	TRUE	23	
27	node 28	0	38000	0	TRUE	24	
28	node 29	0	59000	0	TRUE	24	
29	node 30	0	25000	0	TRUE	24	

Table A-19: Diagnostic object tree

The related tests details are given in Table A-20.

Node ID	Test Cost OK	Test Cost NotOK	Test Uncertainty	Test Coverage
0	5	5	100	100
1	3	3	100	100
2	4	4	100	100
3	5	5	100	100
4	6	6	100	100
5	7	7	100	100
6	6	6	100	100
7	10	10	100	100
8	7	7	100	100
9	5	5	100	100
10	4	4	100	100
11	5	5	100	100
12	2	2	100	100
13	6	6	100	100
14	4	4	100	100
15	3	3	100	100
16	2	2	100	100
17	7	7	100	100
18	5	5	100	100
19	4	4	100	100
20	2	2	100	100
21	7	7	100	100
22	8	8	100	100
23	1	1	100	100
24	2	2	100	100
25	9	9	100	100
26	10	10	100	100
27	4	4	100	100
28	7	7	100	100
29	5	5	100	100

Table A-20: Test specificities

The suspicion links are given in Table A-21.

Link ID	Link is from symptom	Link is from fault code	ID of origin	ID of targer	Rank
1	TRUE	FALSE	2	4	1
2	TRUE	FALSE	2	5	1
3	TRUE	FALSE	2	6	1
4	TRUE	FALSE	3	5	1
5	TRUE	FALSE	3	14	1
6	TRUE	FALSE	3	15	1
7	TRUE	FALSE	4	15	1
8	TRUE	FALSE	4	16	1
9	TRUE	FALSE	4	23	1
10	FALSE	TRUE	4	2	1
11	FALSE	TRUE	4	11	1
12	FALSE	TRUE	4	12	1
13	FALSE	TRUE	5	5	1
14	FALSE	TRUE	5	6	1
15	FALSE	TRUE	5	24	1
16	FALSE	TRUE	6	3	1
17	FALSE	TRUE	6	15	1
18	FALSE	TRUE	7	24	1
19	FALSE	TRUE	8	5	1
20	FALSE	TRUE	8	6	1
21	FALSE	TRUE	8	14	1
22	TRUE	FALSE	6	14	1
23	TRUE	FALSE	6	15	1
24	TRUE	FALSE	6	16	1
25	TRUE	FALSE	7	15	1
26	TRUE	FALSE	7	16	1
27	TRUE	FALSE	8	6	1
28	TRUE	FALSE	8	14	1

Table A-21: Suspicion link

The fault code data are given in Table A-22.

Fault Code ID	Text	Expert Rule Nbr	Weight	Level	Childs	Parent	Number of Childs
0	Fault Codes	0	100	0	1;2;3	0	3
1	ECU1	0	100	1	4;5	0	2
2	ECU2	0	100	1	6;7	0	2
3	ECU3	0	100	1	8	0	1
4	FC S4	3	100	2		1	0
5	FC S5	3	100	2		1	0
6	FC S6	2	100	2		2	0
7	FC S7	1	100	2		2	0
8	FC S8	3	100	2		3	0

Table A-22: Fault codes

The perceived symptoms are listed in Table A-23.

Symptom ID	Text	Expert Rule Nbr	Weight	Level	Childs	Parent	Number of Childs
0	Symptoms	0	100	0	;1;5	0	2
1	Group 1	0	100	1	;2;3;4	0	3
2	S1	3	100	2		1	0
3	S2	3	100	2		1	0
4	S3	3	100	2	;5	1	1
5	Group 2	0	100	1	;6;7;8	0	3
6	S9	3	100	2		5	0
7	S10	2	100	2		5	0
8	S11	2	100	2		5	0

Table A-23: Perceived symptom tree

The description of the cases (symptoms and broken component) is given in Table A-24.

Scenario ID	Active Symptoms	Active fault code	Broken component	Optimal orbit
0	S1, S2, S3		Node 16	2
1	S2, S3		Node 23	2
2		FC S5, FC S6, FC S7, FC S8	Node 29	4
3	S3, S9	FC S8	Node 24	2
4	S9, S10		Node 23	2
5		FC S4, FC S5	Node 18	1
6	S9, S10	FC S8	Node 25	2
7	S3	FC S4	Node 22	2
8		FC S6, FC S7	Node 29	4
9		FC S6, FC S7	Node 28	2
10	S2, S3	FC S4, FC S6	Node 23	2
11	S9	FC S8	Node 24	2
12	S3	FC S4, FC S5	Node 20	2
13	S10, S11		Node 22	2
14	S10		Node 24	2

Table A-24: Cases for the simulation of diagnosis session

3.2 Experiences with SIDIS Enterprise's diagnosis engine

The orbit of the guided fault finding procedures with the diagnosis engine of SIDIS Enterprise is given in Table A-25:

SIDIS Enterprise Guided fault finding procedures					
cases	Optimal time	Optimal card	Time	Cardinal	Wrong tests
1	11	2	38	9	7
2	5	2	23	7	5
3	14	4	45	11	7
4	9	2	19	4	2

5	5	2	15	5	3
6	2	1	9	2	1
7	10	2	37	7	5
8	7	2	14	4	2
9	7	4	33	9	5
10	9	2	28	8	4
11	5	2	43	11	7
12	9	2	13	3	1
13	10	2	27	6	4
14	7	2	15	4	2
15	9	2	9	2	0
Average:	7.93	2,2	24.53	6.13	3.66

Table A-25: SIDIS Performance

The same orbit performed with different symptom weights (cf. chapter III section 1.2.1) is given in Table A-26.

Cases	Optimal cardinal	Diagnosis with default symptom weight	Diagnosis with symptom weight proportional to hitting sets
1	2	9	3
2	2	7	3
3	4	11	5
4	2	7	5
5	2	5	4
6	1	2	2
7	2	7	4
8	2	7	4
9	4	7	6
10	2	6	5
11	2	13	3
12	2	6	5
13	2	7	6
14	2	5	5
15	2	3	2
Average	2.2	6.8	4.1

Table A-26: SIDIS Performance with different symptom weights

The Table A-27 detail the orbit for a test agenda ranking with balance of suspicion moment and test cost (detailed in chapter III section 1.2.2).

α =0.6	Ranking of test agendas with balanced suspicion and ratio maximum test costs by current test cost					
	cases	Optimal time	Optimal card	time	Cardinal	Wrong tests
	1	11	2	38	9	7
	2	5	2	23	7	5
	3	14	4	45	11	7
	4	9	2	22	5	3

5	5	2	15	5	3
6	2	1	31	7	5
7	10	2	37	7	5
8	7	2	14	4	2
9	7	4	33	9	5
10	9	2	28	8	4
11	5	2	36	9	6
12	9	2	26	5	3
13	10	2	28	6	4
14	7	2	19	5	3
15	9	2	9	2	0
Average	7.93	2.2	26.93	6.6	4.13

Table A-27: Performance with balanced suspicion and test costs ratio

Table A-28 details the orbit for a test agenda ranking with balance of suspicion moment and normalized test cost (detailed in chapter III section 1.2.3).

	Ranking of tests agendas with balanced suspicion and normalized ratio maximum test costs by current test cost					
	cases	Optimal time	Optimal card	time	Cardinal	Wrong tests
	1	11	2	38	9	7
	2	5	2	23	7	5
	3	14	4	45	11	7
	4	9	2	22	5	3
	5	5	2	15	5	3
	6	2	1	31	7	5
α =0.9999	7	10	2	37	7	5
	8	7	2	14	4	2
	9	7	4	33	9	5
	10	9	2	28	8	4
	11	5	2	36	9	6
	12	9	2	26	5	3
	13	10	2	28	6	4
	14	7	2	19	5	3
	15	9	2	9	2	0
	Average:	7.93	2.2	26.93	6.6	4.13

Table A-28: Performance with balanced suspicion and normalized test costs ratio

The Table A-29 details the result obtained with a soft constraint between test costs and suspicion with the parameters $\alpha = 0.5$ and $\beta = 0.1$.

Linearization heuristic with suspicion and ratio maximum test cost by current test costs					
cases	Optimal time	Optimal card	time	Cardinal	Wrong tests
1	11	2	38	13	11
2	5	2	117	12	10
3	14	4	13	15	11
4	9	2	119	12	10

5	5	2	11	13	11
6	2	1	124	11	9
7	10	2	125	12	10
8	7	2	30	9	7
9	7	4	238	12	8
10	9	2	238	16	12
11	5	2	42	17	14
12	9	2	124	15	13
13	10	2	119	11	9
14	7	2	13	6	4
15	9	2	119	6	4
Average	7.93	2.2	98	12	9.53

Table A-29: Performance with exponential linearization of suspicion and test costs ratio

Table A-30 details the results of a soft constraint between suspicion and test costs with first depth inspection with $\alpha = 0.5$.

	Linearization heuristic with suspicion and ratio maximum test cost by current test costs and average child's test costs					
	cases	Optimal time	Optimal card	Time	Cardinal	Wrong tests
$\alpha = 0.5$	1	11	2	38	9	7
	2	5	2	23	7	5
	3	14	4	45	11	7
	4	9	2	20	5	3
	5	5	2	15	5	3
	6	2	1	2	1	0
	7	10	2	37	7	5
	8	7	2	12	5	3
	9	7	4	33	9	5
	10	9	2	28	8	4
	11	5	2	38	10	7
	12	9	2	13	3	1
	13	10	2	10	2	0
	14	7	2	9	3	1
	15	9	2	12	3	1
	Average:	7.93	2.2	22.33	5.86	3.46

Table A-30: Performance with suspicion and tests costs first depth inspection

Table A-31 details the results when the symptom question is implemented within the normal diagnosis engine of SIDIS Enterprise.

SIDIS Enterprise with symptom questions					
cases	Optimal time	Optimal card	Time	Cardinal	Wrong tests
1	11	2	38	9	7
2	5	2	23	7	5
3	14	4	45	11	7
4	9	2	19	4	2

5	5	2	15	5	3
6	2	1	7	1	0
7	10	2	37	7	5
8	7	2	14	4	2
9	7	4	33	9	5
10	9	2	28	8	4
11	5	2	43	11	7
12	9	2	13	3	1
13	10	2	27	6	4
14	7	2	15	4	2
15	9	2	9	2	0
Average:	7.93	2.2	24.4	6.06	3.6

Table A-31: Performance with symptom question

3.3 Experience with the meta-heuristic

Table A-32 details the results with the entropy based approach and with SIDIS Enterprise's usual engine (cf. chapter III section 2.2.1.1).

SIDIS Enterprise guided fault finding procedures		
cases	Meta-heuristic based suspicion	SIDIS Enterprise
1	7	9
2	7	7
3	10	11
4	4	4
5	5	5
6	2	2
7	7	7
8	4	4
9	8	9
10	8	8
11	10	11
12	3	3
13	6	6
14	4	4
15	2	2
Average:	5.8	6.13

Table A-32: Guided fault finding with both approaches

Table A-33 details the results obtained for the approach with the meta-heuristic with an increasing weight of the precision factor. (cf. chapter III section 2.2.1.2).

SIDIS Enterprise cardinal and meta-heuristic entropy based suspicion and increasing precision								
cases	Cardinal SIDIS	10%	20%	30%	40%	50%	60%	70%
1	9	8	9	9	8	8	8	8
2	7	6	6	6	4	4	4	4
3	11	11	11	11	12	12	12	12

4	4	4	4	4	7	7	7	7
5	5	5	5	5	3	3	3	3
6	2	2	2	2	1	1	1	1
7	7	7	7	7	8	8	8	8
8	4	4	4	4	8	8	8	8
9	9	9	9	9	7	7	7	7
10	8	4	4	4	5	5	5	6
11	11	11	11	11	6	6	6	6
12	3	3	3	3	12	12	12	12
13	6	4	4	4	4	4	4	4
14	4	3	3	3	3	3	3	5
15	2	2	2	2	2	2	2	3
Average:	6.13	5.53	5.6	5.6	6	6	6	6.26

Table A-33: Guided fault finding with meta-heuristic

The Table A-34 details the results obtained with the meta-heuristic based approach with an increasing value of the factor related to the test costs.

SIDIS Enterprise cardinal and metaheuristic entropy based suspicion and incresaing test costs											
cases	**Cardinal**	**Carnial - 10% cost**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
1	9	7	7	7	7	7	7	7	18	18	18
2	7	4	4	4	4	4	4	4	4	4	4
3	11	11	5	4	4	4	4	4	4	4	4
4	4	6	3	3	3	3	3	3	3	3	3
5	5	3	4	4	4	4	4	5	5	5	5
6	2	1	1	1	1	1	1	1	1	1	1
7	7	7	7	7	7	7	7	8	8	9	9
8	4	8	5	5	5	5	5	5	5	5	5
9	9	4	4	2	2	2	2	2	2	2	2
10	8	6	6	6	5	5	5	5	5	5	5
11	11	2	2	2	2	2	2	2	2	2	2
12	3	13	13	13	13	11	11	11	11	8	9
13	6	6	6	6	6	6	6	25	25	25	21
14	4	2	2	2	2	2	2	2	2	2	2
15	2	2	2	2	2	2	2	2	2	2	2
Average:	**6.13**	**5.46**	**4.73**	**4.53**	**4.46**	**4.33**	**4.33**	**5.73**	**6.46**	**6.33**	**6.13**

Table A-34: Guided fault finding's orbit with the meta-heuristic and increasing test costs factor

Table A-35 summarizes the average results obtained with the meta-heuristic approach and an increasing weight of the life time factor.

Relative weight of life time factor	0%	10%	20%	30%	40%	50%	60%
Average cardinal of SIDIS Enterprise guided fault finding orbit	6.13	6.13	6.13	6.13	6.13	6.13	6.13
Average cardinal of the guided fault finding with a combined metric of life time and suspicion links	5.6	5.4	5.13	6.26	6.53	8.9	9

Table A-35: Guided fault finding with the meta-heuristic and increasing life time factor

Table A-36 details the causal links included in the model used for the simulation.

ID	Node ID of Origin	Node ID of Target	Weight of causal link
1	4	5	100
2	4	6	100
3	5	4	100
4	5	6	100
5	6	5	100
6	15	16	100
7	15	23	100
8	23	5	100
9	24	5	100

Table A-36: Causal links in the model

Table A-37 details the results obtained by the meta-heuristic with an increasing weight of the causal suspicion factor.

SIDIS Enterprise cardinal and meta-heuristic entropy suspicion and causal suspicion											
cases	Cardinal	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
1	9	7	7	7	7	7	8	7	7	7	7
2	7	4	4	4	4	4	4	4	4	4	4
3	11	12	12	12	12	12	12	12	12	12	12
4	4	8	7	7	7	7	7	7	7	7	8
5	5	3	3	3	3	3	3	3	3	3	3
6	2	1	1	1	1	1	1	1	1	1	1
7	7	7	7	7	7	7	7	7	7	7	7
8	4	8	8	8	8	8	8	8	8	8	8
9	9	4	4	4	4	4	4	4	4	4	4
10	8	6	6	6	6	6	6	6	6	6	6
11	11	4	3	3	3	3	3	3	3	3	4
12	3	17	13	13	12	12	12	12	12	15	17
13	6	5	5	5	5	5	5	5	5	5	5
14	4	2	2	2	2	2	2	2	2	2	2
15	2	2	2	2	2	3	3	3	3	3	3
Average:	6.13	6	5.6	5.6	5.53	5.6	5.66	5.6	5.6	5.8	6.06

Table A-37: Impact of the causal factor in the meta-heuristic

Table A-38 summarizes the results obtained by the meta-heuristic with an increasing weight of the causal suspicion factor.

Relative weight of causal moment	Average cardinal of SIDIS Enterprise guided fault finding orbit	Average cardinal of the guided fault finding with a combined metric of suspicion and causal links
0	6.13	6.13
0.1	6.13	6
0.2	6.13	5.6
0.3	6.13	5.6
0.4	6.13	5.53
0.5	6.13	5.6
0.6	6.13	5.66

0.7	6.13	5.6
0.8	6.13	5.6
0.9	6.13	5.8
1	6.13	6.06

Table A-38: Impact of the weight of the causal factor on the guided fault finding orbit

Table A-39 gives the relevant results of the research of the function points of the meta-heuristic approach. The parameters that change are given in each corresponding row and their values in the header column.

		Variation								
		0%	10%	20%	30%	40%	50%	60%	70%	80%
Average cardinal of guided fault finding orbit	SIDIS Enterprise	6.13	6.13	6.13	6.13	6.13	6.13	6.13	6.13	6.13
	a0=10; *a2*=10; *a3*=100; *a4*=20; *a1* increasing	6.14	7.64	5.35	5	5	4.92	5.57		
	a0=10; *a1*=-10; *a3*=100; *a4*=20; *a2* increasing	5.85	6.78	5.5	6.42					
	a0=10; *a1*=-10; *a2*=10; *a3*=100; *a4* increasing	5.64	5.57	5.57	5.57	5.57	5.42	5.5	5.5	5.5

Table A-39: Investigation of the function points of the meta-heuristic

3.4 Experience with the combined approach

3.4.1 Data of the first use case

Table A-40 details the diagnostic object tree used in the use case of Chapter III section 2.3.2.2. The model does not contain any fault code.

ID	EN Title	IsLeaf	Test Cost OK	Test Cost NotOK	HParent	Hchilds
0	Objets Diag	FALSE	100	100	0	1;3;4
1	Engine Electronics	FALSE	100	100	0	2
2	6075_Beanstadungen	TRUE	100	100	1	
3	Engine Control	TRUE	100	100	0	
4	Engine Start	FALSE	100	100	0	5;6;7;8;9; 10;11;12
5	9563_Engine start	TRUE	100	100	4	
6	6747_Car Acess System supply	FALSE	100	100	4	
7	5931_Brake light switch	TRUE	100	100	4	
8	2251_Ignition start switch	TRUE	100	100	4	
9	8011_EWS Data Link	TRUE	100	100	4	
10	0827_Wheel speed	TRUE	100	100	4	
11	6459_Parking	TRUE	100	100	4	
12	8619_Clamp50	TRUE	100	100	4	
13	9083_Error1:Car access system	TRUE	100	100	6	
14	2155_Error1:Current Service task	TRUE	100	100	6	

15	4315_Error8:CAS Car access system	TRUE	100	100	6	
16	5099_Error8:DME Motor Electronics	TRUE	100	100	6	
17	8043_Error10:DME motor electronics	TRUE	100	100	6	

Table A-40: Diagnostic object tree

Table A-41 details the symptom tree of the same model.

ID	Title	Expert Rule Nbr	Weight	Level	Childs	Parent	Number of Childs
0	Symptômes	1	100	0	;1	0	1
1	S1	14	100	1		0	0

Table A-41: Symptom description

Table A-42 and Table A-43 detail the suspicion rules and causal links of the same model.

ID	From Symptom	From FaultCode	Origin	Ending	Rank
1	TRUE	FALSE	0	0	1
2	TRUE	FALSE	1	2	1
3	TRUE	FALSE	1	5	1
4	TRUE	FALSE	1	6	1
5	TRUE	FALSE	1	13	1
6	TRUE	FALSE	1	14	1
7	TRUE	FALSE	1	15	1
8	TRUE	FALSE	1	16	1
9	TRUE	FALSE	1	17	1
10	TRUE	FALSE	1	7	1
11	TRUE	FALSE	1	8	1
13	TRUE	FALSE	1	9	1
14	TRUE	FALSE	1	10	1
15	TRUE	FALSE	1	11	1
16	TRUE	FALSE	1	12	1

Table A-42: Description of the expert rules

ID	Origin	Target	Prior Certainty
1	6	7	62
2	6	17	54
3	6	8	80
4	7	15	90
5	7	14	20
6	17	10	83
7	17	15	12
8	8	14	75
9	8	10	10
10	11	13	62
11	11	9	54
12	11	16	80
13	13	5	5
14	13	12	16
15	9	2	21
16	9	5	15

17	16	12	4
18	16	2	16
19	3	7	100
20	3	17	100
21	3	8	100
22	3	15	100
23	3	14	100
24	3	10	100
25	3	15	100
26	3	14	100
27	3	10	100
28	3	13	100
29	3	9	100
30	3	16	100
31	3	5	100
32	3	12	100
33	3	2	100
34	3	5	100
35	3	12	100
36	3	2	100

Table A-43: Description of the causal relationships

Table A-44 and Table A-45 detail the impact of the causal moment on the first test agenda for different values of α and β. The "incremental delta" row is the square distance between two successive objects of the test agenda (value of the causal moment). This indicates the discrimination power between the object in the test agenda. The higher the "incremental delta", the higher are the differences between the suspected candidates.

	Value of α								
Nodes	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
5099	0.17	0.382	0.655	1.02	1.53	2.29	3.57	6.12	13.7
9083	0.163	0.366	0.628	0.978	1.46	2.2	3.42	5.87	13.2
4315	0.157	0.354	0.608	0.946	1.402	2.12	3.31	5.67	12.7
2155	0.155	0.3506	0.601	0.935	1.402	2.1	3.27	5.61	12.6
811	0.13	0.292	0.501	0.78	1.17	1.75	2.73	4.68	10.5
827	0.126	0.283	0.486	0.756	1.134	1.7	2.64	4.53	10.21
843	0.111	0.25	0.428	0.666	1	1.5	2.33	4	9
Delta	0.386	0.869	1.491	2.321	3.472	5.214	8.120	13.926	31.264
Incremental delta	0.030	0.070	0.121	0.188	0.285	0.424	0.657	1.128	2.543

Table A-44: Investigation of the impact of α in the first test agenda

	Value of β						
Nodes	0.1	0.2	0.3	0.4	0.5	0.6	1
Node 5099	13.77	30.56	39.8	48	56.8	65.9	100
Node 9083	13.2	30.4	39.7	47.9	56.6	65.8	100
Node 8043	12.9	30.32	39.2	47.6	56.45	65.7	100
Node 4315	12.7	30.22	38	47.2	56.38	65.3	100
Node 2251	12.6	29.12	37.6	46.3	55.76	65.2	100
Node 5931	10.8	28.67	37.2	46.2	55.42	64	100
Delta	31.09587	73.2159	94.54295	115.6293	137.7522	160.0002	244.949

Incremental delta	1.924812	1.206027	1.421267	1.03923	0.75326	1.276715	0

Table A-45: Investigation of the impact of β in the first test agenda

3.4.2 Results of the combined approach with the previous model

This section contains the details about the combined approach of a heuristic and a model based diagnosis. The simulations are run through the same case base and model as described in section 4.1. Table A-46 to Table A-56 list the results obtained with the combined approach with changing values of α, β and γ.

	No life time (C_{LT}=0), β = 1 and γ = 1									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	9	9	9	9	9	9	9	9	9	9
case 2	4	4	4	4	4	4	4	4	4	4
case 3	12	12	12	12	12	12	12	12	12	12
case 4	11	11	11	11	11	11	11	11	11	11
case 5	3	3	3	3	3	3	3	3	3	3
case 6	4	4	4	4	4	4	4	4	4	4
case 7	6	6	6	6	6	6	6	6	6	6
case 8	8	8	8	8	8	8	8	8	8	8
case 9	5	5	5	5	5	5	5	5	5	5
case 10	5	5	5	5	5	5	5	5	5	5
case 11	10	10	10	10	10	10	10	10	10	10
case 12	17	17	17	17	17	17	17	17	17	17
case 13	5	5	5	5	5	5	5	5	5	5
case 14	5	5	5	5	5	5	5	5	5	5
average	7.43	7.43	7.43	7.43	7.43	7.43	7.43	7.43	7.43	7.43

Table A-46: Combined approach with different values for α, β = 1 and γ = 1

	No life time (C_{LT}=0), β = 0.9 and γ = 1									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	7	7	7	7	7	7	7	7	7	7
case 2	4	4	4	4	4	4	4	4	4	4
case 3	12	12	12	12	12	12	12	12	12	12
case 4	11	11	11	11	11	11	11	11	11	11
case 5	3	3	3	3	3	3	3	3	3	3
case 6	4	4	4	4	4	4	4	4	4	4
case 7	6	6	6	6	6	6	6	6	6	6
case 8	8	8	8	8	8	8	8	8	8	8
case 9	5	5	5	5	5	5	5	5	5	5
case 10	5	5	5	5	5	5	5	5	5	5
case 11	10	10	10	10	10	10	10	10	10	10
case 12	17	17	17	17	17	17	17	17	17	17
case 13	4	4	4	4	4	4	4	4	4	4
case 14	5	5	5	5	5	5	5	5	5	5
average	7.21	7.21	7.21	7.21	7.21	7.21	7.21	7.21	7.21	7.21

Table A-47: Combined approach with different values for α, β = 0.9 and γ = 1

	No life time (C_{LT}=0), β = 0.8 and γ = 1									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	7	7	7	7	7	7	7	7	7	7
case 2	4	4	4	4	4	4	4	4	4	4
case 3	12	12	12	12	12	12	12	12	12	12
case 4	11	11	11	11	11	11	11	11	11	11
case 5	3	3	3	3	3	3	3	3	3	3
case 6	4	4	4	4	4	4	4	4	4	4
case 7	6	6	6	6	6	6	6	6	6	6
case 8	8	8	8	8	8	8	8	8	8	8
case 9	5	5	5	5	5	5	5	5	5	5
case 10	5	5	5	5	5	5	5	5	5	5
case 11	10	10	10	10	10	10	10	10	10	10
case 12	17	17	17	17	17	17	17	17	17	17
case 13	4	4	4	4	4	4	4	4	4	4
case 14	5	5	5	5	5	5	5	5	5	5
average	7.21	7.21	7.21	7.21	7.21	7.21	7.21	7.21	7.21	7.21

Table A-48: Combined approach with different values for α, β = 0.8 and γ = 1

	No life time (C_{LT}=0), β = 0.7 and γ = 1									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	7	7	7	7	7	7	7	7	7	7
case 2	4	4	4	4	4	4	4	4	4	4
case 3	12	12	12	12	12	12	12	12	12	12
case 4	11	11	11	11	11	11	11	11	11	11
case 5	3	3	3	3	3	3	3	3	3	3
case 6	4	4	4	4	4	4	4	4	4	4
case 7	6	6	6	6	6	6	6	6	6	6
case 8	8	8	8	8	8	8	8	8	8	8
case 9	2	2	2	2	2	2	2	2	2	2
case 10	5	5	5	5	5	5	5	5	5	5
case 11	10	10	10	10	10	10	10	10	10	10
case 12	15	15	15	15	15	15	15	15	15	15
case 13	4	4	4	4	4	4	4	4	4	4
case 14	5	5	5	5	5	5	5	5	5	5
average	6.86	6.86	6.86	6.86	6.86	6.86	6.86	6.86	6.86	6.86

Table A-49: Combined approach with different values for α, β = 0.7 and γ = 1

	No life time (C_{LT}=0), β = 0.6 and γ = 1									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	7	7	7	7	7	7	7	7	7	7
case 2	4	4	4	4	4	4	4	4	4	4
case 3	12	12	12	12	12	12	12	12	12	12
case 4	8	8	8	8	8	8	8	8	8	8
case 5	3	3	3	3	3	3	3	3	3	3
case 6	4	4	4	4	4	4	4	4	4	4
case 7	6	6	6	6	6	6	6	6	6	6
case 8	8	8	8	8	8	8	8	8	8	8
case 9	2	2	2	2	2	2	2	2	2	2
case 10	5	5	5	5	5	5	5	5	5	5

case 11	7	7	7	7	7	7	7	7	7	7
case 12	15	15	15	15	15	15	15	15	15	15
case 13	4	4	4	4	4	4	4	4	4	4
case 14	5	5	5	5	5	5	5	5	5	5
average	6.43	6.43	6.43	6.43	6.43	6.43	6.43	6.43	6.43	6.43

Table A-50: Combined approach with different values for α, $\beta = 0.6$ and $\gamma = 1$

	No life time (C_{LT}=0), $\beta = 0.5$ and $\gamma = 1$									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	9	9	9	9	9	9	9	9	9	9
case 2	4	4	4	4	4	4	4	4	4	4
case 3	11	11	11	11	11	11	11	11	11	11
case 4	7	7	7	7	7	7	7	7	7	7
case 5	3	3	3	3	3	3	3	3	3	3
case 6	4	4	4	4	4	4	4	4	4	4
case 7	6	6	6	6	6	6	6	6	6	6
case 8	6	6	6	6	6	6	6	6	6	6
case 9	2	2	2	2	2	2	2	2	2	2
case 10	8	8	8	8	8	8	8	8	8	8
case 11	6	6	6	6	6	6	6	6	6	6
case 12	16	16	16	16	16	16	16	16	16	16
case 13	4	4	4	4	4	4	4	4	4	4
case 14	5	5	5	5	5	5	5	5	5	5
average	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5	6.5

Table A-51: Combined approach with different values for α, $\beta = 0.5$ and $\gamma = 1$

	No life time (C_{LT}=0), $\beta = 0.4$ and $\gamma = 1$									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	9	9	9	9	9	9	9	9	9	9
case 2	4	4	4	4	4	4	4	4	4	4
case 3	11	11	11	11	11	11	11	11	11	11
case 4	3	3	3	3	3	3	3	3	3	3
case 5	4	4	4	4	4	4	4	4	4	4
case 6	5	5	5	5	5	5	5	5	5	5
case 7	7	7	7	7	7	7	7	7	7	7
case 8	6	6	6	6	6	6	6	6	6	6
case 9	2	2	2	2	2	2	2	2	2	2
case 10	8	8	8	8	8	8	8	8	8	8
case 11	2	2	2	2	2	2	2	2	2	2
case 12	16	16	16	16	16	16	16	16	16	16
case 13	5	5	5	5	5	5	5	5	5	5
case 14	2	2	2	2	2	2	2	2	2	2
average	6	6	6	6	6	6	6	6	6	6

Table A-52: Combined approach with different values for α, $\beta = 0.4$ and $\gamma = 1$

	No life time (C_{LT}=0), $\beta = 0.3$ and $\gamma = 1$									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	9	9	9	9	9	9	9	9	9	9
case 2	5	5	5	5	5	5	5	5	5	5
case 3	8	8	8	8	8	8	8	8	8	8

case 4	3	3	3	3	3	3	3	3	3	3
case 5	6	6	6	6	6	6	6	6	6	6
case 6	7	7	7	7	7	7	7	7	7	7
case 7	7	7	7	7	7	7	7	7	7	7
case 8	3	3	3	3	3	3	3	3	3	3
case 9	2	2	2	2	2	2	2	2	2	2
case 10	9	9	9	9	9	9	9	9	9	9
case 11	2	2	2	2	2	2	2	2	2	2
case 12	17	17	17	17	17	17	17	17	17	17
case 13	5	5	5	5	5	5	5	5	5	5
case 14	2	2	2	2	2	2	2	2	2	2
average	6.07	6.07	6.07	6.07	6.07	6.07	6.07	6.07	6.07	6.07

Table A-53: Combined approach with different values for α, β = 0.3 and γ = 1

	No life time (C_{LT}=0), β = 0.2 and γ = 1									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	8	9	9	9	9	9	9	9	9	9
case 2	8	8	8	8	8	8	8	8	8	8
case 3	8	8	8	8	8	8	8	8	8	8
case 4	3	3	3	4	4	4	4	7	7	7
case 5	6	6	6	6	6	6	6	6	6	6
case 6	7	7	7	7	7	7	7	7	7	7
case 7	8	8	8	8	8	8	8	8	8	8
case 8	3	3	3	3	3	3	3	3	3	3
case 9	2	2	2	2	2	2	2	2	2	2
case 10	9	9	9	9	9	9	9	9	9	9
case 11	2	2	2	2	2	2	2	2	2	2
case 12	19	19	19	19	19	19	19	19	19	19
case 13	6	6	6	6	6	6	6	6	6	6
case 14	2	2	2	2	2	2	2	2	2	2
average	6.5	6.57	6.57	6.64	6.64	6.64	6.64	6.86	6.86	6.86

Table A-54: Combined approach with different values for α, β = 0.2 and γ = 1

	No life time (C_{LT}=0), β = 0.1 and γ = 1									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	8	8	8	9	9	9	9	9	9	9
case 2	8	8	8	8	8	8	8	8	8	8
case 3	7	7	7	7	7	7	7	7	7	7
case 4	3	3	3	7	7	7	7	7	7	7
case 5	6	6	6	6	6	6	6	6	6	6
case 6	9	9	9	9	9	9	9	9	9	9
case 7	8	8	8	8	8	8	8	8	8	8
case 8	3	3	3	3	3	3	3	3	3	3
case 9	2	2	2	2	2	2	2	2	2	2
case 10	12	9	9	9	9	9	9	9	9	9
case 11	2	2	2	2	2	2	2	2	2	2
case 12	19	19	19	19	19	19	19	19	19	19
case 13	6	6	6	6	6	6	6	6	6	6
case 14	2	2	2	2	2	2	2	2	2	2
average	6.79	6.57	6.57	6.93	6.93	6.93	6.93	6.93	6.93	6.93

Table A-55: Combined approach with different values for α, $\beta = 0.1$ and $\gamma = 1$

	No life time (C_{LT}=0), $\beta = 0$ and $\gamma = 1$									
α	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
case 1	7	7	7	7	7	7	7	7	7	13
case 2	18	18	18	18	18	18	18	18	18	2
case 3	22	22	22	22	22	22	22	22	22	22
case 4	3	3	3	3	3	3	3	3	3	15
case 5	11	11	11	11	11	11	11	11	11	2
case 6	1	1	1	1	1	1	1	1	1	1
case 7	16	16	16	16	16	16	16	16	16	13
case 8	5	5	5	5	5	5	5	5	5	5
case 9	4	4	4	4	4	4	4	4	4	4
case 10	18	18	18	18	18	18	18	18	18	2
case 11	2	2	2	2	2	2	2	2	2	2
case 12	17	17	17	17	17	17	17	17	17	8
case 13	18	18	18	18	18	18	18	18	18	14
case 14	2	2	2	2	2	2	2	2	2	15
average	10.3	10.3	10.3	10.3	10.3	10.3	10.3	10.3	10.3	8.43

Table A-56: Combined approach with different values for α, $\beta = 0$ and $\gamma = 1$

Table A-57, Table A-58 and Table A-59 summarize the results in average values for the research of the function points of the combined approach.

β	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Average orbit of the combined approach (C_{LT}=0)	10.1	6.84	6.67	6.07	6	6.5	6.42	6.857	7.21	7.21	7.42
Average orbit of SIDIS Enterprise's algorithm	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42

Table A-57: Summary of the combined approach for different values of β

	No life time (C_{LT}=0), $\alpha = 0.9$, $\beta = 0.4$										
γ	1	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2
Average orbit	6	6.14	6.5	6.07	6.14	6.21	6.57	6.57	6.57	6.5	6.5
SIDIS Enterprise's average orbit	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42

Table A-58: Summary of the combined approach for different values of γ, $\alpha = 0.9$ and $\beta = 0.4$

	Average orbit with life time for $\alpha = 0.9$, $\beta = 0.4$ and $\gamma = 1$										
C_{LT}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Average orbit	6	6	5.85	6	6.21	6.21	6.14	6.14	6.57	6.42	8.07
Average SIDIS Orbit	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42	7.42

Table A-59: Summary of the combined approach for different values of C_{LT}, $\alpha = 0.9$, $\beta = 0.4$ and $\gamma = 1$

Appendix B

This appendix contains the details about the developed prototype and about the models used in the experiences made in chapter IV sections 3.2 to 3.4. Each model is detailed in a separate section. The presentation of each model is ordered as follow: the diagnostic object tree, the perceived symptom tree, the fault code tree, the suspicion rules, the cases descriptions and finally the updated parameters by the feedback engine. In order to make the tables as small as possible, only the relevant nodes of the tree have been selected and the most of the IDs have been re-assigned (except for the fault codes for vehicle C in section 4.1).

1 Developed prototype

1.1 Details about the data tables

1.1.1 Main fields of the project class

Table B-1 details the major fields and components of the project class.

Modifiers	Field types	Field Names	Description and comments
Private	treeview	*treeview1*	The control for the perceived symptom tree
Private	treeview	*treeview2*	The control for the fault memory text tree
Private	treeview	*treeview3*	The control for the diagnostic object tree
Private	datagrid	*datagrid1*	The control for the suspicion rules
Private	datagrid	*datagrid2*	The control for the causal links
Private	datagrid	*datagrid3*	The control for the display of the attributes of the selected nodes
Public	dataset	*dataset1*	Contains the table collections with the data representing the model of a vehicle
Private	contextMenuStrip	*contextMenuStrip1*	Contains specific commands related to the edition of the trees as: add node, add child node, delete node, etc…
Private	TableLayoutPanel	*tableLayoutPanel1*	A container for all other controls
Private	Enum TreeTypes	*Selected_tree*	This variable can take the values: Symptoms, FaultCodes or DiagObject depending of the current selected tree by the user
Private	TreeNode	*Selected_Node*	This variable is fixed to the current selected tree node in one of the tree view control
Private	String	*current_directory*	This field contains the path of the executable file of the program
Private	String	*pathAndFileName*	This field contains the path and name of the file that is currently opened, or saved
Public	List<Case>	*LoadedCases*	Contains the list of current loaded business cases
Public	String	*Language*	This variable is fixed to the current language of the project: "fr-FR" for French, "de-DE" for German and "en-EN" for English

Table B-1: List of major fields in the project class

1.1.2 Symptom and fault code data tables

Table B-2 details the columns of the symptom and fault codes data tables.

Data table 1 and 2 for perceived symptoms and fault codes			
Index	**Column title**	**Variable type**	**Description and comments**
0	*ID*	int	The unique identifier of the element
1	*FR_Title*	string	The title in French
2	*DE_Title*	string	The title in German
3	*EN_Title*	string	The title in English
4	*ExpertRuleNbr*	int	The number of outgoing suspicion links
5	*Active*	bool	Value indicating if this element is active or not
6	*Weight*	double	The weight of the element
7	*ROEWeight*	double	The weight obtained by the feedback engine
8	*Level*	int	The level of the element in the tree
9	*Childs*	string	The identifiers of the child's elements in the tree
10	*Parent*	int	The identifier of the parent node in the tree
11	*NumberChilds*	int	The number of child element
12	*IndexTreeview*	int	The index in the tree view
13	*DepictedInTree*	bool	Value indicating if the node is shown in the tree view
14	*OPWeight*	double	The operational weight of the element (which depending on the used diagnosis strategy can be equal to the weight, the feedback weight, or calculated through the discrimination power of the outgoing links)

Table B-2: Definition of the first and second data tables

1.1.3 Diagnostic objects data table

Table B-3 details the columns of the diagnostic object data table.

Data table 3: diagnostic objects			
Index	**Column title**	**Variable type**	**Description and comments**
0	*ID*	int	The unique identifier of the element
1	*FR_Title*	string	The title in French
2	*DE_Title*	string	The title in German
3	*EN_Title*	string	The title in English
4	*IncomingRulesNumber*	int	The number of incoming suspicion links
5	*IncomingCausalLinks*	int	The number of incoming causal links

6	*OutgoingCausalLinks*	int	The number of outgoing causal links
7	*PartialMoment*	double	The partial suspicion moment of the object
8	*PartialCausalMoment*	double	The partial causal suspicion moment of the object
9	*TotalSM*	double	The global suspicion moment of the object
10	*Weight_Hlink*	double	The weight of the hierarchy link
11	*ROEWeightHLink*	double	The weight of the hierarchy link obtained by the feedback engine
12	*Lifetime*	double	The lifetime of the component defined in kilometres
13	*NumberChilds*	int	The number of child of that element
14	*IsLeaf*	bool	Boolean value indicating if this element is a leaf in the tree or not
15	*TestCostOK*	double	The test cost of the associated test to that component (for an OK answer)
16	*TestCostNotOk*	double	The test cost of the associated test to that component (for an notOK answer)
17	*TestResult*	double	The test result during the diagnosis session
18	*PriorTestResult*	bool	The prior test result
19	*TestUncertainty*	double	the test uncertainty
20	*TestCoverage*	double	The test coverage
21	*TestRequireUserAction*	int	The number of user actions required by the tests
22	*Hparent*	string	The identifiers of the parent component
23	*Hchilds*	string	The identifiers of the child component
24	*Cparent*	string	The identifiers of the causal ancestors of the component
25	*Cchilds*	string	The identifiers of the causal successors of the node
26	*DepictedInTree*	bool	Boolean value indicating if the component was depicted into the tree or not

Table B-3: Definition of the diagnostic objects

1.1.4 Link data tables

Table B-4 describes the suspicion and causal links data tables.

Data table 4: suspicion links			
Index	**Column title**	**Variable type**	**Description and comments**
0	*ID*	int	The unique identifier of the element
1	*fromSymptom*	bool	Boolean value equal to true if the rule starts from a perceived symptom
2	*fromFaultCode*	bool	Boolean value equal to true if the rule starts from a fault code

3	*Origin*	int	Identifier of the origin of the link
4	*Ending*	int	Identifier of the target of the link
5	*Rank*	int	Rank of the suspicion rule
6	*ROEWeight*	double	Weight from the feedback engine
Data table 5: causal links			
Index	**Column title**	**Variable type**	**Description and comments**
0	*ID*	int	The unique identifier of the element
1	*Origin*	int	Identifier of the origin of the link
2	*Target*	int	Identifier of the target of the link
3	*PriorCertainty*	double	Weight of the suspicion rule
4	*ROECertainty*	double	Weight from the feedback engine

Table B-4: Data tables of suspicion and causal links

1.1.5 Data tables related to the language fields

Table B-5 details the columns of the language data tables. All attributes have their equivalent for a stemmed base of terms with the prefix "STE_".

Data Table 6, 7 and 8: Language resources for perceived symptom retrieval engine (only columns without stemming)			
Index	**Column title**	**Variable type**	**Description and comments**
21	*BOT*	string	The base of term
22	*DocFreq*	string	Non-null values of the document frequency matrix
23	*invDocFreq*	string	Non-null values of the inverse document frequency matrix
24	*TermFreqX*	string	Position and value of non null values in the term frequency matrix
25	*TermFreqY*	string	
26	*TermFreqVal*	string	
27	*IDFX*	string	Same as previous for IDF matrix
28	*IDFY*	string	
29	*IDFVal*	string	
30	*HarmanX*	string	Same as previous for Harman matrix
31	*HarmanY*	string	
32	*HarmanVal*	string	
33	*CroftTmax*	string	Same as previous for Croft matrix
34	*CroftX*	string	
35	*CroftY*	string	
36	*CroftVal*	string	
37	*OkapiX*	string	Same as previous for Okapi matrix
38	*OkapiY*	string	
39	*OkapiVal*	string	
40	*Symlength*	string	Symptom length in number of non empty words
41	*avSymlength*	string	Average symptom length of the entire set of perceived symptoms
45	*NeuroX*	string	Position and value of non null values in the synaptic matrix of the correspondence graph
46	*NeuroY*	string	
47	*NeuroVal*	string	

48	*FeedX*	string	Same as previous for the feedback or popularity matrix
49	*FeedY*	string	
50	*FeedVal*	string	

Table B-5: Extract of data tables of the language resources without stemming

1.2 Engines of the prototype

1.2.1 Perceived symptom search engine

Table B-6 describes the fields and methods used by the symptom search engine.

Main Fields				
Index	**Modifier**	**Fields type**	**Fields name**	**Description and comments**
1	protected internal	limit (enum)	*Limit*	Enumeration of the types of acceptance limit, can be equal to: UpperValue or Heuristic
2	protected internal	computingmethod (enum)	*ComputingMethod*	Enumeration of the types of weight computing method. Can be equal to: normal (*tf.idf* formula), *Croft*, *Harman*, *Okapi*, *Neuro* (for the correspondence graph)
3	protected internal	SimilarityMethod (enum)	*SimilarityMethod*	Enumeration of the possible calculation of the similarity. Can be equal to: *Jaccard*, *Cosinus*, *Euclide*, *Dice* or *Neuro*
4	protected internal	double	*Upperlim*	The acceptability limit
5	protected internal	bool	*UseStem*	Boolean indicating if the stemming algorithm is used
6	protected internal	bool	*UseThesaurus*	Boolean indicating if the thesaurus is used for the research of synonyms
7	protected internal	bool	*Usesplittingwords*	Boolean indicating if long words are splitted (only in German)
8	protected internal	bool	*Usereplaceverbs*	Boolean indicating if the irregular verbs are replaced by their infinitive form
9	protected internal	bool	*Use_feedback*	Boolean indicating if the feedback is used
10	protected internal	double	*Feedback_coeff*	Value of the feedback coefficient
11	protected internal	double	*Croft_Constante*	Value of the Croft constant
12	protected internal	list<string>	*BaseOfTerms*	List of the base of terms without stemming
13	protected internal	int [ ]	*Docfrequency*	Array of the document frequencies
14	protected internal	double [ ]	*inversedocfrequency*	Array of the inverse document frequencies
15	protected internal	int [,]	*TermFrequency*	Term frequency matrix
16	protected	double [,]	*IDF*	Inverse document frequency

	internal			matrix
17	protected internal	double [,]	*Harman*	Harman matrix
18	protected internal	int [ ]	*Croft_Tmax*	Max term frequency
19	protected internal	double [,]	*Croft*	Croft matrix
20	protected internal	double [,]	*Okapi*	Okapi matrix
21	protected internal	int [ ]	*Symptomlength*	Array of symptom length
22	protected internal	double [,]	*NeuroMatrix*	Adjacent matrix of the correspondence graph
23	protected internal	double	*av_sym_length*	Average symptom length
24	public	int [,]	*FeedbackMatrix*	Popularity or feedback matrix
25	public	int [ ]	*Request_Frenquency_Vector*	Array of request frequency vector
26	private	double [ ]	*Request_weight_vector*	Array of request weight vector
27	private	double [,]	*SymMatrix*	Symptom weight matrix
28	public	double [ ]	*Resultingarray*	Array of the similarities of the request with all symptoms
29	private	List<double>	*Resultinglist*	List of the similarity scores of relevant symptoms
Main methods				
1	public	void	*setSearchVariable*	Public methods that initialized the limit, *computingmethod*, *SimilarityMethod*, *Upperlim*, *UseStem* etc.
2	public	List<int>	*SearchSymptoms*	Method that initializes the option of a search and computes the similarity between the request/symptoms and return the lists of IDs of the relevant symptom
3	protected internal	void	*UpdateFeedbackmatrix*	Method that updates the feedback matrix depending of the returned symptoms and the selected symptoms

Table B-6: Main fields and methods of the perceived symptom retrieval engine

1.2.2 Diagnostic engine

Table B-7 contains the major fields and methods of the diagnostic engine of the prototype.

Main Fields				
Index	**Modifier**	**Fields type**	**Fileds name**	**Description and comments**
1	private	int	*selected_CaseIndex*	Index of the case in the list of cases, that will be simulated
2	public	List<int>	*First_Test_Agenda*	List of IDs of suspected components after the calculation of the first test agenda

3	public	List<int>	*Test_Agendas*	List of IDs of suspected components after the calculation of *i*-th test agenda
4	public	List<int>	*AlreadyTested*	List of IDs of the already tested objects
5	public	List<int>	*OKTested*	List of Ids of the tested components with the answer OK
6	private	List<int>	*NotOKTested*	List of Ids of the tested components with the answers notOK
7	public	List<int>	*BadTestCertainty*	List of Ids of test with a wrong confidence
8	public	List<int>	*BadTestCoverage*	List of Ids of tests with a wrong coverage
9	public	int	*orbit*	Orbit of the guided fault finding procedure
10	public	double	*cost_of_session*	Costs of the guided fault finding procedure in minutes
11	public	int	*uselesstest*	Number of useless tests performed during the guided fault finding procedure
12	public	double	*alpha_causal*	Order of the entropy for the calculation of the causal suspicious moment
13	public	double	*beta_causal*	Linearization coefficient for the combination of the heuristic moment and the causal moment
14	public	double	*gama_causal*	Exponent of the linearization coefficient *beta_causal*
15	public	bool	*Use_causal_ feedback*	Boolean defining if the weight of the causal links obtain by thhe feedback evaluation engine are used
16	private	bool	*Use_Symptom_ Question*	Boolean defining if the perceived symptom questioning is used
17	public	double	*Life_time_factor*	Weight of the lifetime factor
18	private	Enum	*CurrentAlgo*	Enumeration of the possible diagnosis algorithm (e.g. SIDIS Enterprise, meta-heuristic, combined approach)
19	private	Enum	*TestRankingMethod*	Enumeration of the possible tests ranking metrics (e.g. suspicion divided by test costs)
20	private	Enum	*SymWeights*	Enumeration of the possible symptoms and fault codes weightening methods (e.g. normal weights, weight depending of hitting sets, etc.)
Main methods				
1	public	void	*InitializeCase*	Method that initializes the case and fixes the value of all the parameters
2	public	void	*ComputeFirst TestAgenda*	Method that computes the first test agenda
3	public	void	*ComputeNext TestAgenda*	Method that computes the i^{th} test agenda (for $i>1$)

Table B-7: Main fields and methods of the diagnostic engine

1.2.3 Feedback engine

Table B-8 details the main attributes and methods of the feedback engine of the prototype.

Main Fields				
Index	**Modifier**	**Fields type**	**Fileds name**	**Description and comments**
1	public	List<int>	*First_Test_Agenda*	List of IDs of suspected components after the calculation of the first test agenda
2	public	List<int>	*OKTested*	List of Ids of the tested components with the answer OK
3	private	List<int>	*NotOKTested*	List of Ids of the tested components with the answers notOK
4	public	List<int>	*BadTestCertainty*	List of Ids of test with a wrong confidence
5	public	List<int>	*BadTestCoverage*	List of Ids of tests with a wrong coverage
6	public	List<int>	*AllSuspiciousObjects*	List of all suspicious components issue from the calculation of the first test agenda
7	public	bool	*AutoUpdate FeedbackVal*	Boolean for the automatic updating or not of the parameters issue from the feedback evaluation
8	public	int	*NumberOfIterations*	Number of performed automatic simulation before the evaluation of the feedbacks proceed
9	public	int [,]	*Request_Frenquency_Vector*	The term frequency vector of the request
8	public	List<int>	*SelectedSymptoms*	The list of the IDs of the user-validated retrieved symptoms for the diagnostic session.
Main methods				
1	public	void	*AutoUpdate Feedbackvalues*	Method that performs the evaluation of the suspicion rules, causal links, tests costs, etc… and saves the value in the data tables

Table B-8: Main fields and methods of the feedback engine

2 Vehicle A

2.1 Basic knowledge structure

The Table B-9, Table B-10, Table B-11 and Table B-12 lists the relevant diagnostic objects, perceived symptoms, fault codes (and the associated fault code text) and suspicion rules.

ID	Text	Test confidence	Test coverage	Parent node ID
0	Root diagnostic objects	100	100	0
1	Actuation/Engine	100	100	0
2	Selfdiagnosis capable systems	100	100	1
3	Engine fuel supply	100	100	1
4	Fuel preparation / Injection	100	100	1
5	Cooling system	100	100	1

6	Connections to ESP	100	100	2
7	Engine management ECU	100	100	2
8	Connections to ABS	100	100	2
9	Crank actuation	100	100	2
10	Exhaust-gas system	100	100	2
11	Starter electric power supply	100	100	2
12	Preheat system	100	100	2
13	Engine function	100	100	2
14	F_45 Stop light switch ESP	100	100	6
15	G201_45 Brake pressure	100	100	6
16	G419_45 ESP Sensor signal	100	100	6
17	G85_45 Sensor fr steering angle	100	100	6
18	F36_27 clutch pedal switch	100	100	7
19	G108_24 Lambda sensor 1	100	100	7
20	G130_24 Lambda sensor after catalytic convertor	100	100	7
21	G42_24 Sensor for intake air temperature	100	100	7
22	SIG45_24 switch for stop lights and brake pedal switch	100	100	8
23	Pressure sensor for brake force amplification	100	100	8
24	Sensor for steering angle	100	100	8
25	V93_45 Recirculating pump	100	100	8
26	SYS13_ Engine overtorqued	100	100	9
27	SYS13_ Abnormal operating noise	100	100	9
28	KD1366 Plate concealed / not fixed	100	100	9
29	SYS13_ Crank axel rotation idle	100	100	9
30	G212_26 Potentiometer for exhaust gas recirculation	100	100	10
31	G235_26 Sensor for exhaust gas temperature	100	100	10
32	K83 Warning indicator for exhaust gas	100	100	10
33	N18_26 valve for exhaust gas recirculation	100	100	10
34	SYS26 exhaust system before cat	100	100	10
35	N345_26 switching valve for cooler Exhaust gas recirculation	100	100	10
36	Verusst SYS26 exhaust pipes	100	100	10

37	Check SYS26 exhaust gas recirculation	100	100	10
38	A_27 battery health check	100	100	11
39	A_27 battery charge	100	100	11
40	A_27 replace battery when the encode	100	100	11
41	SYS27_Genereller test over under voltage	100	100	11
42	B__27 Starter	100	100	11
43	J179_28_SG_Fuer Glüheiztautomatik	100	100	12
44	J359_28 relays for small heat output	100	100	12
45	J360_28 relays for large heating capacity	100	100	12
46	F265_19 thermostat for identification field-driven Engine cooling	100	100	5
47	G83_19 coolant temperature sensor Cooler out	100	100	5
48	J293_19 control unit for fan for Kühlmitttel	100	100	5
49	Electronic regulated SYS_19 cooling system check	100	100	5
50	SYS24 idle checking	100	100	13
51	Check catalyst SYS26	100	100	13
52	Encode ECU J220_24	100	100	13
53	Create Readinesscode	100	100	13

Table B-9: Relevant node of the diagnostic objects of the vehicle A

ID	Text	Weight	Level	Parent node ID
0	Symptoms	100	0	0
1	Engine	100	1	0
11	Engine temperature too high	100	2	1
12	Indicator Engine temperature does not work	100	2	1
13	Water temperature in engine cooling system is too high	100	2	1
14	Oil temperature in engine too high	100	2	1
2	Emission	100	1	0
21	Black emission when engine starts	100	2	2
22	Black emission when engine runs at full speed	100	2	2
23	Black emission when engines starts with cold exterior	100	2	2

	temperature			
24	Black emission at full acceleration	100	2	2
3	Brake lights	100	1	0
31	brake lights does not work	100	2	3

Table B-10: Relevant node of the perceived symptom tree of the vehicle A

ID	Text	Weight	Level	Parent
0	Fault code root	100	0	0
1	mot1281	100	1	0
2	mot6000	100	0	0
3	bre2000	100	0	0
4	1798 Locked engine control unit	100	1	1
5	17698 Sensor for cooling fluid temperature G83 signal too high	100	1	1
6	17699 G83 cooling fluid temperature sensor G83 signal too small	100	1	1
7	17947 Implausible signal F36	100	1	1
8	16518 Sensor1 no activity	100	1	1
9	16158 Sensor 1 signal too slow	100	1	1
10	000_ No Display	100	0	1
19	16610_P0266 sensor for G79_185 implausible signal	100	0	1
20	006 no signal/ no communication	100	0	1
21	16955_F_ Implausible signal	100	1	1
22	18320_Pressure_F_G294_SC_Plus	100	0	1
23	18321_ pressure sensor _ f _ Brake force amplification_G294_SC_ground	100	0	1
24	18322 pressure sensor_f_G294_ Implausible signal	100	1	1
29	16473_ Fuel pressure _control circuit _implausible Signal	100	0	1
30	17686 Valve_Dosing_ fuel_N290	100	1	2
31	16804_Lambda correction_after_Cat_convertor_Rule	100	0	2
32	16520 Sonde1 of electrical failure in the circuit	100	0	2
33	16705 Sensor G28 implausibles signal	100	0	2
41	17748 camshaft position sensor signal too small	100	0	3
44	18039 G79 sensor signal too high	100	0	3
45	18039 G79 sensor signal too small	100	0	3

Table B-11: Relevant node of the fault code tree of the vehicle A

ID	From Symptom	From FaultCode	Origin	Ending	Rank
1	TRUE	FALSE	11	5	1
2	TRUE	FALSE	21	8	1
3	TRUE	FALSE	31	6	1
4	FALSE	TRUE	4	7	1
5	FALSE	TRUE	5	47	1
6	FALSE	TRUE	5	48	1
7	FALSE	TRUE	6	47	1
8	FALSE	TRUE	6	48	1
9	FALSE	TRUE	7	18	1
10	FALSE	TRUE	7	49	1
11	FALSE	TRUE	7	19	1
12	FALSE	TRUE	8	19	1
13	FALSE	TRUE	8	49	1
14	FALSE	TRUE	9	48	1
15	FALSE	TRUE	9	20	1
16	FALSE	TRUE	10	21	1
17	FALSE	TRUE	19	23	1
18	FALSE	TRUE	19	24	1
19	FALSE	TRUE	20	23	1
20	FALSE	TRUE	20	24	1
21	FALSE	TRUE	21	8	1
22	FALSE	TRUE	22	8	1
23	FALSE	TRUE	23	18	1
24	FALSE	TRUE	23	6	1
25	FALSE	TRUE	23	16	1
26	FALSE	TRUE	24	6	1
27	FALSE	TRUE	24	14	1
28	FALSE	TRUE	29	18	1
29	FALSE	TRUE	30	19	1
30	FALSE	TRUE	31	20	1
31	FALSE	TRUE	33	21	1
32	FALSE	TRUE	41	15	1
33	FALSE	TRUE	44	6	1
34	FALSE	TRUE	44	15	1

35	FALSE	TRUE	45	6	1
36	FALSE	TRUE	45	16	1

Table B-12: Relevant suspicion rules of the vehicle A

2.2 Cases description

The Table B-13, describe the cases used for the simulation with the km stand of the vehicle, the IDs of the active perceived symptoms, the IDs of the active fault codes, the IDs of the tests that answers notOK and the faulty component.

Case	Vehicle km reading	Active Symptoms ID	Active fault codes ID	Test NotOK	Broken Node_ID
Case 1	55000	11	4;5;6;7;8;9;10	47;5;1	47
Case 2	29000	21	19;20;21;22;23;24	1;22	1;22
Case 3	37000	31	19;20;21;22;29;30; 31;33;41;44;45	22;1;8; 15;6;2	22;15
Sub-case 3	37000		44;45;20;21	15;6;2	15

Table B-13: Description of the three cases used for vehicle A

2.3 Results of feedback evaluation

The Table B-14 and Table B-15 list the created causal links (with their weight) as well as the updated rank of the suspicion rules. All objects are referenced by their correspondant ID.

ID	Origin	Target	Weight
1	0	0	100
2	47	7	33
3	47	18	33
4	47	48	33
5	47	19	33
6	47	20	33
7	47	21	33
8	22	8	33
9	22	18	33
10	22	6	33
11	22	16	66
12	22	14	33
13	22	15	66
14	22	6	33

Table B-14: Causal links created by feedback engine for vehicle A

ID	Ending	Rank
6	48	2
8	48	2
20	24	2
33	6	2
36	16	2

Table B-15: Updated rank of the suspicion rules in vehicle A

3 Vehicle B

3.1 Basic knowledge structure

The Table B-16, Table B-17, Table B-18 and Table B-19 lists the relevant diagnostic objects, perceived symptoms, fault codes (and the associated fault code text) and suspicion rules.

ID	Text	Test confidence	Test coverage	Parent node ID
0	Engine	100	100	0
1	Self diagnosis capable systems	100	100	0
2	Fuel preparation injection	100	100	0
3	Exhaust system	100	100	0
4	Starter power	100	100	0
5	Connection to the air conditioning	100	100	0
6	Motor	100	100	1
7	Cooling system	100	100	1
8	Features engine control unit	100	100	1
9	Nouveau subsystems constraints	100	100	1
10	G28_23 sensor for engine rotation speed	100	100	2
11	G31_23 sensor for load pressure	100	100	2
12	G62_23 coolant temperature sensor	100	100	2
13	G70_23 air mass measure	100	100	2
14	KD2639 exhaust gas recirculation	100	100	3
15	N345_26 switching valve for cooler Exhaust gas recirculation	100	100	3
16	SYS26_exhaust pipe soothed	100	100	3
17	SYS26 exhaust gas recirculation test	100	100	3
18	C_27 generator error	100	100	4
19	C_27 generator examination	100	100	4
20	J53_27 Relay Starter	100	100	4
21	J655_27 relay for battery cutoff	100	100	4
22	SIG87_24 signal and from the air conditioning	100	100	5
23	SIG87_24 signal PM climate	100	100	5
24	SIG24_87 heating temperature option	100	100	5

25	Nouveau SIG24_87 engine speed	100	100	5
26	SYS27_oelueberwachung	100	100	6
27	G28_23 engine speed	100	100	6
28	J317_23 relays for power supply terminal 30	100	100	6
29	J17_20 fuel pump relay	100	100	6
30	G38_19 coolant temperature sensor Cooler out	100	100	7
31	F265_19 thermostat for identification field-driven Engine cooling	100	100	7
32	J293_19 control unit for fan coolant	100	100	7
33	KD1962 tube for cooling fluid	100	100	7
34	Engine control unit	100	100	8
35	SYS24 idle checking	100	100	8
36	SYS24 Sensor retainer before cat	100	100	8
37	Create Readinesscode	100	100	8
38	Encode	100	100	34
39	Create readiness code	100	100	34
40	G295_24 sensor for NOx	100	100	6
41	G267_80 potentiometer knob temperature option	100	100	6
42	G336_24 Lambda sensor and lambda scheme	100	100	6
43	J293_19 control unit for fan cooling fluid	100	100	6
44	J220_24 control unit for Motortronic	100	100	6
45	N30_33_24 Injectors	100	100	6
46	N70_N127_N291_N292 ignition coils	100	100	6

Table B-16: Relevant nodes of the diagnostic object tree of vehicle B

ID	Text	Weight	Level	Parent node ID
0	Symptoms	100	0	0
1	Display of oil temperature	100	1	0
11	Wrong display of oil temperature	100	2	1
12	Display of oil temperature always on minimum	100	2	1
13	Display of oil temperature always on maximum	100	2	1
14	Display of oil temperature oscillates	100	2	1
2	ABS Display	100	1	0
21	ABS Display always blinking	100	2	2
22	ABS display always on	100	2	2

Table B-17: Relevant nodes of the perceived symptom tree of vehicle B

ID	Text	Weight	Parent node ID
0	Fault codes root	100	0

1	mot6000	100	0
2	bre2000	100	0
3	16705 G28 implausibles signal	100	1
5	16500 G62 implausibles signal	100	1
6	1744 Motor torque monitoring rule limit exceeded	100	1
7	17973 Control J338 error in Default settings	100	1
8	17972 Control under voltage in Default settings	100	1
9	17501 Fuel measure malfunctioning	100	1
10	16944 Power supply implausibles signal	100	1
16	17748 G28 position camshaft limit exceeded	100	2
17	16621 P0237 sensor pressure G31 signal to small	100	2
18	18080_P1672 cooling air control short-circuit After mass	100	2
19	00257 P0101 G70 air mass measure implausibles Signal	100	2
20	17441 Signal G295 outside tolerance	100	2
22	16514 Sonde1 no activity	100	1
23	17972 DK control unit J33 error	100	1
31	16890 Sensor 1 internal resistance implausible	100	1
32	16890 Idle control speed signal except Tolerance	100	1

Table B-18: Relevant fault codes for vehicle B

ID	From Symptom	From FaultCode	Origin	Ending	Rank
1	TRUE	FALSE	0	0	1
2	TRUE	FALSE	11	3	1
3	TRUE	FALSE	11	6	1
4	TRUE	FALSE	21	7	1
5	FALSE	TRUE	3	6	1
6	FALSE	TRUE	3	7	1
7	FALSE	TRUE	5	38	1
8	FALSE	TRUE	6	39	1
9	FALSE	TRUE	7	34	1

10	FALSE	TRUE	8	46	1
11	FALSE	TRUE	8	34	1
12	FALSE	TRUE	9	45	1
13	FALSE	TRUE	9	38	1
14	FALSE	TRUE	10	34	1
15	FALSE	TRUE	22	35	1
16	FALSE	TRUE	22	42	1
17	FALSE	TRUE	23	8	1
18	FALSE	TRUE	23	40	1
19	FALSE	TRUE	16	10	1
20	FALSE	TRUE	17	11	1
21	FALSE	TRUE	18	12	1
22	FALSE	TRUE	18	46	1
23	FALSE	TRUE	18	38	1
24	FALSE	TRUE	19	13	1
25	FALSE	TRUE	19	45	1
26	FALSE	TRUE	20	16	1
27	FALSE	TRUE	20	40	1
28	FALSE	TRUE	31	42	1
29	FALSE	TRUE	31	35	1
30	FALSE	TRUE	32	8	1
31	FALSE	TRUE	32	40	1

Table B-19: Relevant suspicion rules for vehicle B

3.2 Cases description

The Table B-20 describe the cases used for the simulation with the km stand of the vehicle, the IDs of the active perceived symptoms, the IDs of the active fault codes, the IDs of the tests that answers notOK and the faulty component.

Case	Vehicle km reading	Active Symptoms ID	Active fault codes ID	Test NotOK	Broken Node_ID
Case 1	37000	11	3;5;6;7;8;9;10;22;23	1;6;26	26
Case 2	63000	21	16;17;18;19;20;31;32	1;7;30	30

Table B-20: Case description for vehicle B

3.3 Results of feedback evaluation engine

The Table B-21, Table B-22 and Table B-23 list the created causal links (with their weight) as well as the updated fault code weights and the updated rank of the suspicion rules. All objects are referenced by their correspondant ID.

ID	Origin	Target	Weight
1	0	0	100
2	26	38	50
3	26	39	50
4	26	34	75
5	26	46	50
6	26	45	50
7	26	38	50
8	26	35	50
9	26	42	50
10	30	10	50
11	30	11	50
12	30	12	50
13	30	13	50
14	30	45	50
15	30	46	50
16	30	16	50
17	30	40	50
18	30	42	50
19	30	35	50
20	30	8	50
21	30	40	50

Table B-21: Causal links created through feedback engine for vehicle B

ID	Weight
0	100
1	100
2	100
3	100
4	100
5	0
6	0
7	0
8	0

9	0
10	0
16	0
17	0
18	0
19	0
20	0
22	0
23	0
31	0
32	0

Table B-22: Updated fault code weight for vehicle B

ID	From Symptom	From FaultCode	Origin	Ending	Rank
2	TRUE	FALSE	11	3	2

Table B-23: Updated suspicion rule weight for vehicle B

4 Vehicle C

4.1 Basic knowledge structure

The Table B-24, Table B-25, Table B-26 and Table B-27 lists the relevant diagnostic objects, perceived symptoms, fault codes (and the associated fault code text) and suspicion rules.

ID	Text	Test Confidence	Test Coverage	Parent node ID
0	Diagnostic objects	100	100	0
1	Propulsion engine subsystems	100	100	0
2	Exhaust system	100	100	0
3	Starter	100	100	0
4	Ignition	100	100	0
5	Heating	100	100	0
6	Motor	100	100	1
7	Cooling system	100	100	1
8	Fuel injection	100	100	1
9	Crank power	100	100	1
10	G212_26 potentiometer for exhaust gas recirculation	100	100	2
11	G235_26 timer 1 for exhaust gas temperature	100	100	2

12	K83_26 exhaust warning light	100	100	2
13	N18_26 valve for exhaust gas recirculation	100	100	2
14	E45_27 switch for GRA	100	100	3
15	G365 power for GRA	100	100	3
16	J53_27 relays for start-up	100	100	3
17	J655_27 relays for battery shutdown	100	100	3
18	G163_28 reverb donor 2	100	100	4
19	G61_28 Knock sensor1	100	100	4
20	G66_28 Knock sensor2	100	100	4
21	G40_28 reverb signal	100	100	4
22	Potentiometer G267_80 temperature option	100	100	5
23	N279_82 check valve for coolant	100	100	5
24	S24_82 overheating backup heating	100	100	5
25	Z66_82 heater for fuel heating	100	100	5
26	Sensor for engine rotation speed	100	100	6
27	Sensor for brake pedal pressure	100	100	6
28	E45_27 switch for GRA	100	100	6
29	F__94 Stop lights button	100	100	6
30	V7_19 fan for cooling fluid	100	100	7
31	V177_19 Fan 2 for cooling fluid	100	100	7
32	J293_19 control unit for fan for cooling fluid	100	100	7
33	J336 error JK366 in control unit for fans	100	100	7
34	J248_23 SG for direct injection	100	100	8
35	G40_23 hall sensor signal	100	100	8
36	G42_23 signal for air temperature	100	100	8
37	G476_23 coupling line donor	100	100	8
38	J248_23_SG defaults	100	100	34
39	J248_23_SG power supply	100	100	34
40	F194_27 Interlock switch	100	100	6
41	G28_23 signals for engine speed	100	100	6
42	G42_23 signals for air temperature	100	100	6

43	G62_23 signals cooling fluid temperature	100	100	6
44	G70_23 air mass measure	100	100	6
45	G185_5 signals 2 for acceleration pedal position	100	100	6
46	G476_23 coupling line donor	100	100	6
47	J17_20 fuel pump relay	100	100	6
48	J53_27 Relay 1 Starter	100	100	6
49	J293_19 SG for fan for coolant	100	100	6
50	J317_23 relays for power supply Kl30	100	100	6
51	J329_97 relays for power supply Kl15	100	100	6
52	J359_28 relays for small heat output	100	100	6
53	J360_28 relays for large heating capacity	100	100	6
54	J655_27 relays for Battery cut-off	100	100	6
55	F265_19 thermostat for identification field-driven Engine cooling	100	100	7
56	G83_19 donors for coolant temperature Cooler out	100	100	7
57	N293_23 switching valve for air intake tube damper	100	100	8
58	N257_23 valve for bypass door air filter	100	100	8
59	N240_N243_23 injection valves Zyl1 to Zyl4	100	100	8
60	Replace pump nozzle	100	100	8

Table B-24: Relevant nodes of the diagnostic object tree for vehicle C

ID	Text	Weight	Level	Parent node ID
0	Symptoms	100	0	0
1	Engine	100	1	0
11	Engine does not start	100	2	1
12	Engine vibrates abnormaly	100	2	1
2	ABS Display	100	1	0
21	ABS Display always blinking	100	2	2
22	ABS display always on	100	2	2

Table B-25: Relevant nodes of the perceived symptom tree of vehicle C

ID	Text	Weight	Parent node ID

0	Fault code	100	0
1	mot1281	100	0
2	bre2003	100	0
3	get002	100	0
4	16705 P0321 G28 no signal	100	1
5	16496 P0112 G42 signal too small	100	1
6	18042_P1634_000 no display	100	1
7	17000 P0615 control relays break	100	1
8	16501 P0017 coolant temperature signal G62 too small	100	1
9	05441_P1541 of fuel pump relay J17 Break	100	1
10	18009_P1601 relays for voltage supply Kl30_J317	100	1
11	16485_P0101 air mass sensor _G70 signal to small	100	1
12	00257_P0101 air mass measure G70 implausible Signal	100	1
13	16497 Signal G42 air temperature too large	100	2
14	19462 P3006 relays for large heating capacity J360_U_SC to ground	100	2
15	19459 P3003 no display	100	2
16	19460 P3004 relays for small heat output	100	2
17	05633_P1602 relays for power supply Kl30_J318	100	2
18	16999_P0615 starter relay control Break	100	2
19	18358_P1950 fan for heat center v7 flame wheelchairs / blocked	100	2
20	06480 No display	100	2
21	19461 Relays for large heating capacity interrupted	100	2
22	00274_P0112 no signal G42 Air temperature	100	1
23	00280 Donor G62 signal implausible	100	1
24	00275_P0113 no signal G42	100	1
25	16955_P0571 stop lights switch F implausible Signal	100	1
26	006 Spa final after plus	100	1
27	16610_P0226 donor G79_185 implausible signal	100	1
28	17909 P1501 fuel pump relay short circuit to ground	100	1

29	000_Keine display	100	1
30	18080 P1672 cooler fan control 1U SC	100	1
31	18352 P1944 Temperature too high ECU for cooler fan	100	3
32	18353 P1945 ECU for cooler fan defective	100	3
33	18965_P2533_ignitor SC to ground	100	3
34	18020_P1612 SG incorrectly encoded	100	3
35	16994_P0612 SG encoded wrong Variant	100	3
36	19502_P0346 Starter relay 1 electric Error in the circuit	100	3
37	19506_P3051 feedback signal except tolerance	100	3
38	18008_P0101 implausible signal	100	3
70	16627_P0243 solenoid valve for Shop pressure bounding N75	100	2
71	02272_Ruhestrom Stufe1	100	2
72	00992 Load circuit active	100	3
73	19556_P3100 engine for V157_SC to plus	100	1
74	17949 P1541 Relay J17 break	100	1
75	004 No signal communication	100	1
76	G419_004 implausible signal	100	1
77	G419_45 implausible signal	100	1
78	G518_009 interruption trip after mass	100	1
79	00532 Power supply	100	1
80	02253 Generator high temperature off control	100	1
81	19505_P3047 no signal	100	1
82	17090 Step sensor F125 implausible signal	100	1
83	17094 Error in circuit G93_38	100	1
84	17234_P0850_000 no Signal	100	1
85	Starter does not rotate 19510 P3054	100	1
86	18042_P1634 donors 2 for Acceleration pedal position G185 Signal too large	100	1
87	18009_P1601 relay for Spannungsversorgung_Kl30_J317	100	1
88	Gear SG variants not fit 18249 engine together	100	1
89	17904 Sensor G93 signal too	100	1

	large		
90	01283_037 Relay J360 SC to plus	100	1
91	12293_P3005 Relay JK360 SC final to ground	100	1
92	16611_P0227 signal G79 too small	100	1
93	16864 Management cooling system - electrical failure	100	1
94	16894 Signal F60 interrupted	100	1

Table B-26: Relevant nodes of the fault code tree of vehicle C

ID	From Symptom	From FaultCode	Origin	Ending	Rank
1	TRUE	FALSE	11	0	1
2	TRUE	FALSE	11	6	1
3	TRUE	FALSE	21	6	1
4	FALSE	TRUE	4	26	1
5	FALSE	TRUE	5	42	1
6	FALSE	TRUE	6	45	1
7	FALSE	TRUE	7	27	1
8	FALSE	TRUE	7	48	1
9	FALSE	TRUE	7	42	1
10	FALSE	TRUE	8	6	1
11	FALSE	TRUE	8	54	1
12	FALSE	TRUE	8	43	1
13	FALSE	TRUE	9	6	1
14	FALSE	TRUE	9	47	1
15	FALSE	TRUE	10	51	1
16	FALSE	TRUE	10	45	1
17	FALSE	TRUE	11	44	1
18	FALSE	TRUE	11	45	1
19	FALSE	TRUE	12	44	1
20	FALSE	TRUE	12	43	1
21	FALSE	TRUE	13	42	1
22	FALSE	TRUE	13	50	1
23	FALSE	TRUE	13	45	1
24	FALSE	TRUE	14	53	1
25	FALSE	TRUE	14	50	1
26	FALSE	TRUE	15	52	1

27	FALSE	TRUE	15	47	1
28	FALSE	TRUE	16	52	1
29	FALSE	TRUE	17	52	1
30	FALSE	TRUE	17	50	1
31	FALSE	TRUE	18	48	1
32	FALSE	TRUE	18	52	1
33	FALSE	TRUE	19	49	1
34	FALSE	TRUE	19	47	1
35	FALSE	TRUE	19	46	1
36	FALSE	TRUE	20	50	1
37	FALSE	TRUE	21	53	1
38	FALSE	TRUE	22	51	1
39	FALSE	TRUE	22	42	1
40	FALSE	TRUE	23	28	1
41	FALSE	TRUE	23	43	1
42	FALSE	TRUE	24	42	1
43	FALSE	TRUE	25	29	1
44	FALSE	TRUE	26	51	1
45	FALSE	TRUE	27	45	1
46	FALSE	TRUE	27	42	1
47	FALSE	TRUE	27	53	1
48	FALSE	TRUE	28	47	1
49	FALSE	TRUE	29	43	1
50	FALSE	TRUE	30	49	1
51	FALSE	TRUE	30	53	1
52	FALSE	TRUE	31	48	1
53	FALSE	TRUE	32	45	1
54	FALSE	TRUE	32	42	1
55	FALSE	TRUE	33	50	1
56	FALSE	TRUE	34	51	1
57	FALSE	TRUE	35	50	1
58	FALSE	TRUE	36	48	1
59	FALSE	TRUE	37	47	1
60	FALSE	TRUE	38	44	1
61	FALSE	TRUE	70	52	1
62	FALSE	TRUE	70	43	1

63	FALSE	TRUE	71	52	1
64	FALSE	TRUE	71	53	1
65	FALSE	TRUE	72	47	1
66	FALSE	TRUE	72	50	1
67	FALSE	TRUE	72	51	1
68	FALSE	TRUE	73	42	1
69	FALSE	TRUE	73	45	1
70	FALSE	TRUE	74	45	1
71	FALSE	TRUE	74	53	1
72	FALSE	TRUE	75	44	1
73	FALSE	TRUE	75	46	1
74	FALSE	TRUE	76	47	1
75	FALSE	TRUE	77	51	1
76	FALSE	TRUE	78	45	1
77	FALSE	TRUE	79	48	1
78	FALSE	TRUE	80	47	1
79	FALSE	TRUE	81	51	1
80	FALSE	TRUE	82	46	1
81	FALSE	TRUE	83	46	1
82	FALSE	TRUE	84	44	1
83	FALSE	TRUE	85	50	1
84	FALSE	TRUE	85	47	1
85	FALSE	TRUE	86	45	1
86	FALSE	TRUE	87	48	1
87	FALSE	TRUE	87	42	1
88	FALSE	TRUE	88	46	1
89	FALSE	TRUE	89	45	1
90	FALSE	TRUE	89	50	1
91	FALSE	TRUE	90	53	1
92	FALSE	TRUE	90	52	1
93	FALSE	TRUE	91	52	1
94	FALSE	TRUE	91	49	1
95	FALSE	TRUE	92	51	1
96	FALSE	TRUE	92	45	1
97	FALSE	TRUE	93	52	1
98	FALSE	TRUE	93	43	1

99	FALSE	TRUE	93	42	1
100	FALSE	TRUE	94	52	1
101	FALSE	TRUE	94	53	1

Table B-27: Relevant suspicion rules for vehicle C

4.2 Cases description

Case	Vehicle km reading	Active Symptoms ID	Active fault codes ID	Test NotOK	Broken Node_ID
Case 1	50000	11	4;5 ;6 ;7 ;8 ;9 ;10 ;11 ;12 13;14;15;16;17;18;19;20 21;22;23;24;25;26;27;28 29;30;31;32;33;34;35;36 37;38	26;6	26
Case 2	63000	21	8;9;10;11;12;13;14;15;16; 17;70;71;72;73;74;75;76; 77;78;79;80;81;82;83;84; 85;86;87;88;89;90;91;92; 93;94	27;6	27

Table B-28: Cases caracteristics for vehicle C

4.3 Results of feedback evaluation engine

ID	Origin	Target	Weight
1	0	0	100
2	26	47	20
3	26	51	20
4	26	45	20
5	26	44	20
6	27	48	20
7	27	42	20
8	27	6	20
9	27	6	20
10	27	47	20
11	27	51	20
12	27	43	20
13	47	42	30
14	47	43	30
15	47	44	40
16	47	45	30

17	47	48	20
18	45	46	20
19	45	47	30
20	45	48	40
21	45	49	30
22	45	28	30
23	45	27	30
24	45	26	20
25	45	25	20
26	46	26	60
27	46	26	50
28	46	27	60
29	46	49	20
30	46	48	80
31	43	51	40
32	43	52	40
33	43	53	90
34	43	48	60
35	43	47	40
36	42	45	50
37	42	51	30
38	42	53	40
39	42	52	30
40	44	25	30
41	44	24	80
42	44	28	20
43	44	29	30
44	44	30	60
45	48	31	30
46	48	32	60
47	48	37	20
48	48	38	20
49	48	39	80
50	48	40	50
51	48	41	90
52	51	42	70

53	51	37	90
54	51	38	20
55	51	48	20
56	51	49	80
57	52	50	40
58	52	51	70
59	52	49	70
60	52	25	20
61	53	26	90
62	53	27	90
63	26	42	20
64	26	45	20
65	26	27	20
66	26	48	20
67	26	42	20
68	26	6	20
69	26	54	20
70	26	43	20
71	26	6	20
72	26	47	20
73	26	51	20
74	26	45	20
75	26	44	20
76	26	43	20
77	26	42	20
78	26	50	20
79	26	45	20
80	26	53	20
81	26	50	20
82	27	43	30
83	27	52	30
84	27	53	30
85	27	47	30
86	27	50	30
87	27	50	30
88	27	42	30

89	27	45	30
90	27	45	30
91	27	53	30
92	27	44	30
93	27	46	30
94	27	47	30
95	27	51	30
96	27	45	30

Table B-29: Causal links created through feedback engine for vehicle C

ID	Weight
4	10
5	10
6	10
7	10
8	10
9	10
10	10
11	20
12	30
13	20
14	40
15	60
16	40
17	10
18	80
19	40
20	30
21	30
22	20
23	40
24	40
25	10
26	10
27	20
28	20

29	20
30	20
31	20
32	20
33	40
34	40
35	40
36	40
37	30
38	50
39	10
40	10
41	20
42	40
43	20
44	20
45	10
46	10
47	10
48	40
49	40
50	40
51	40
52	60
53	60
54	60
55	60
56	60
57	20
58	20
59	10
60	60
61	20
62	20
63	10
64	50

65	50
66	50
67	50
68	10
69	10
70	10
71	50
72	50
73	50
74	30
75	50
76	30
77	10
78	10
79	20
80	20
81	60
82	40
83	20
84	20
85	20
86	20
87	20
88	10
89	20
90	10
91	10
92	30
93	30
94	10

Table B-30: Updated fault code weight for vehicle C

ID	From Symptom	From FaultCode	Origin	Ending	Rank
8	FALSE	TRUE	7	48	2
9	FALSE	TRUE	7	42	3

10	FALSE	TRUE	8	6	1
11	FALSE	TRUE	8	54	2
12	FALSE	TRUE	8	43	3
13	FALSE	TRUE	9	6	2
16	FALSE	TRUE	10	45	2
18	FALSE	TRUE	11	45	2
22	FALSE	TRUE	13	50	3
23	FALSE	TRUE	13	45	2
24	FALSE	TRUE	14	53	2
26	FALSE	TRUE	15	52	2
29	FALSE	TRUE	17	52	2
31	FALSE	TRUE	18	48	2
34	FALSE	TRUE	19	47	2
39	FALSE	TRUE	22	42	2
40	FALSE	TRUE	23	28	3
41	FALSE	TRUE	23	43	3
45	FALSE	TRUE	27	45	2
46	FALSE	TRUE	27	42	2
51	FALSE	TRUE	30	53	2
53	FALSE	TRUE	32	45	2
61	FALSE	TRUE	70	52	2
64	FALSE	TRUE	71	53	2
67	FALSE	TRUE	72	50	2
68	FALSE	TRUE	73	42	2
70	FALSE	TRUE	74	45	2
72	FALSE	TRUE	75	44	2
84	FALSE	TRUE	85	47	2
87	FALSE	TRUE	87	42	2
90	FALSE	TRUE	89	50	2
91	FALSE	TRUE	90	53	2
93	FALSE	TRUE	91	52	2
96	FALSE	TRUE	92	45	2
97	FALSE	TRUE	93	52	2
98	FALSE	TRUE	93	43	3
100	FALSE	TRUE	94	52	2

Table B-31: Updated suspicion rule weight for vehicle C

Appendix C

This appendix is a summary in french of the PhD report.

1 Introduction

L'industrie automobile est actuellement face a un changement global du climat économique. L'un des défis est de satisfaire les demandes spécifiques des clients qui conduisent à des multiples variantes d'un même modèle de véhicule. Afin de distinguer leur image de marque, les constructeurs sont forcés d'innover spécialement dans le domaine de la sécurité, du confort et de l'intégration des technologies de l'information et de communication. Dans ce contexte, 90% des innovations sont dans le domaine de l'électronique. Aujourd'hui les équipements électroniques d'un véhicule constituent près de 30% de la valeur du véhicule contre 0.5% dans les années 80. Une étude de l'AAMA [AAMA, 1998] suggère que la partie de la valeur d'un véhicule neuf dédié au service après vente avoisine les 300$ en moyenne mais que ce pourcentage ne cesse d'augmenter, tendance qui illustre que les constructeurs accorde de plus en plus d'importance au réseau après vente. C'est dans ce contexte que la société Siemens est entrée dans le marché des outils d'aide au diagnostic avec leur produit "Diagnose Entwicklungs-System" (DES) il 'a 20 ans et actuellement avec SIDIS Enterprise qui intègre un système d'information et de diagnostic multimarques pour la gestion complète des réseaux après-vente des constructeurs.
Il y'a cependant un grand nombre de difficultés scientifique et de contraintes économiques pour les systèmes d'aide au diagnostic tel que : le problème des variantes de véhicules, l'interaction de plusieurs domaines physique, des sous-systèmes dynamiques, la mesurabilité limitée, le temps d'une session de diagnostic et le coût de développement des modèles. Le but de cette thèse est d'étudier les concepts théoriques et de réaliser une nouvelle méthodologie du diagnostic off-board qui relie pragmatisme et environnement économique. Les 3 objectifs fondamentaux retenus par une analyse des contraintes industrielle sont :

- **Objectif 1**: Réduire le temps d'une session de diagnostic.
- **Objectif 2**: Améliorer la qualité du diagnostic off-board.
- **Objectif 3**: Réduire le coût de développement des modèles.

Ce résumé de thèse présente les différents modules développés au sein d'une plateforme pour un diagnostic off-board appliqué au domaine automobile. La seconde partie de ce résumé détaille la structure et les réseaux de connaissances dans SIDIS Enterprise. La troisième partie examine le traitement automatique des observations ou symptômes. De plus ce chapitre examine aussi un système d'analyse et d'importation automatique des données de description des calculateurs des véhicules. La quatrième partie examinera la réalisation d'un nouvel algorithme de diagnostic et l'influence des différents paramètres sur ses performances. Une étape de vérification et validation est menée dans la partie V de ce résumé sur des pannes simulées sur des véhicules réels, suivie d'une conclusion générale.

2 Situation actuelle

2.1 L'électronique dans les véhicules d'aujourd'hui

Au cours des dernières années les systèmes électroniques et accessoires dans l'automobile ont constamment augmenté pour couvrir les besoins toujours croissants dans les domaines de l'innovation, du confort, loisir, sécurité, communication et environnement (ex : ABS, ESP, BCS). Aujourd'hui la valeur des micro-processeurs et autres composants semi-conducteurs a atteint $1800 par véhicule [Leen, Heffernan, 2002]. La Figure C-1 présente cette tendance sur la base d'étude statistique de [DaimlerChrysler, 2004].

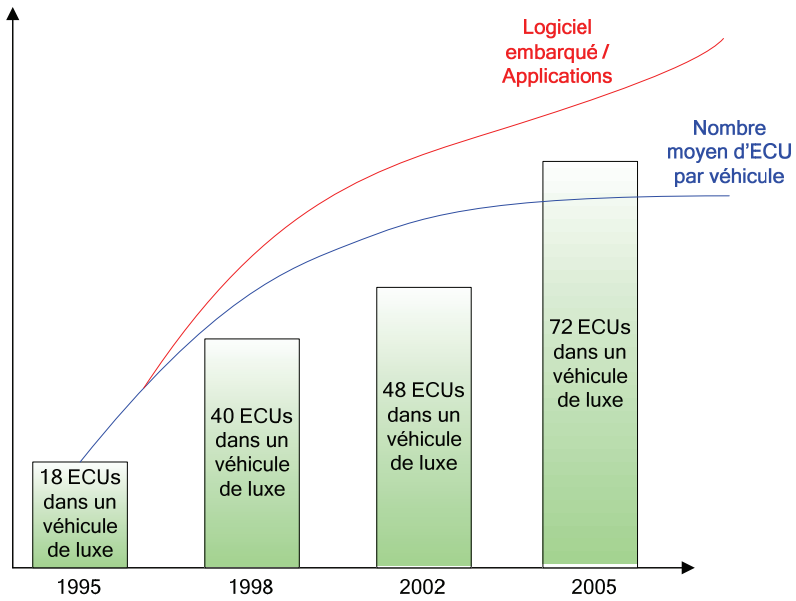

Figure C-1: ECUs dans les véhicules [DaimlerChrysler, 2004]

En parallèle à cette étude, le centre pour recherche dans l'automobile d'Allemagne [ADAC, 2005] a publié une étude (cf. Figure C-2) qui démontre la part croissante des panes électronique. Ce pourcentage atteint 50.5% en 1998 et la tendance est à la hausse.

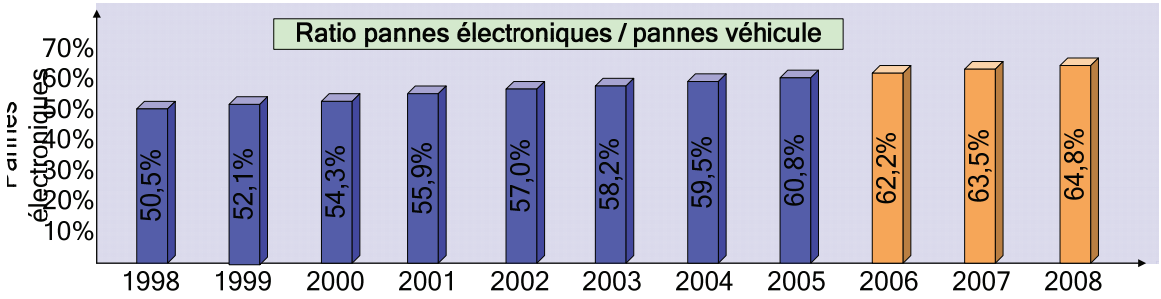

Figure C-2: Statistics of electronics failures [ADAC, 2005]

2.2 Les réseaux après-ventes

Le marché de l'après vente en Europe pour le service et les réparations de véhicules représente 100 milliards d'euros en 2004. Il s'agit du segment le plus important marché secondaire après la vente de véhicule neuf [London Economics, 2006]. Cette même étude souligne que la tendance du ratio garages indépendants sur garage autorisé a tendance à baisser. Cela est essentiellement du à la nature du métier qui a fondamentalement changé. Comme mentionné auparavant la nature des pannes est aujourd'hui essentiellement d'origine électronique ce qui implique l'utilisation de système d'aide au diagnostic capable de communiquer avec les calculateurs des véhicules. Dans les années 80 General Motors a distribué le système CAMS (Computerized Automotive Maintenance System) [William, 2003]. Même si ce système est aujourd'hui obsolète, il est représentatif du changement du cœur métier des mécaniciens. Le CAMS était capable de détecter, analyser et isoler les fautes dans les véhicules de GM équipée d'un calculateur de management du moteur. Cette tendance s'est vérifiée chez d'autres constructeurs qui ont développés leur propre outil de diagnostic. D'autre part Toyota a pu conquérir le marché de l'automobile de luxe aux Etats-Unis grâce à un modèle vendu avec un contrat de service très avantageux ce qui prouve que le service prend de plus en plus d'ampleur pour l'image de la marque dans la chaîne de valeur automobile.

2.3 Vue d'ensemble de SIDIS Enterprise

SIDIS Enterprise est un système modulaire divisé en deux concepts basiques [Ripplinger, 2006] :

- Le système auteur (AS): pour éditer les données des véhicules
- Le client garage (WS): utilize par les mécaniciens dans les garages pour le diagnostic.

La Figure C-3 présente l'architecture des composants de SIDIS Enterprise basés sur le mode client/serveur.

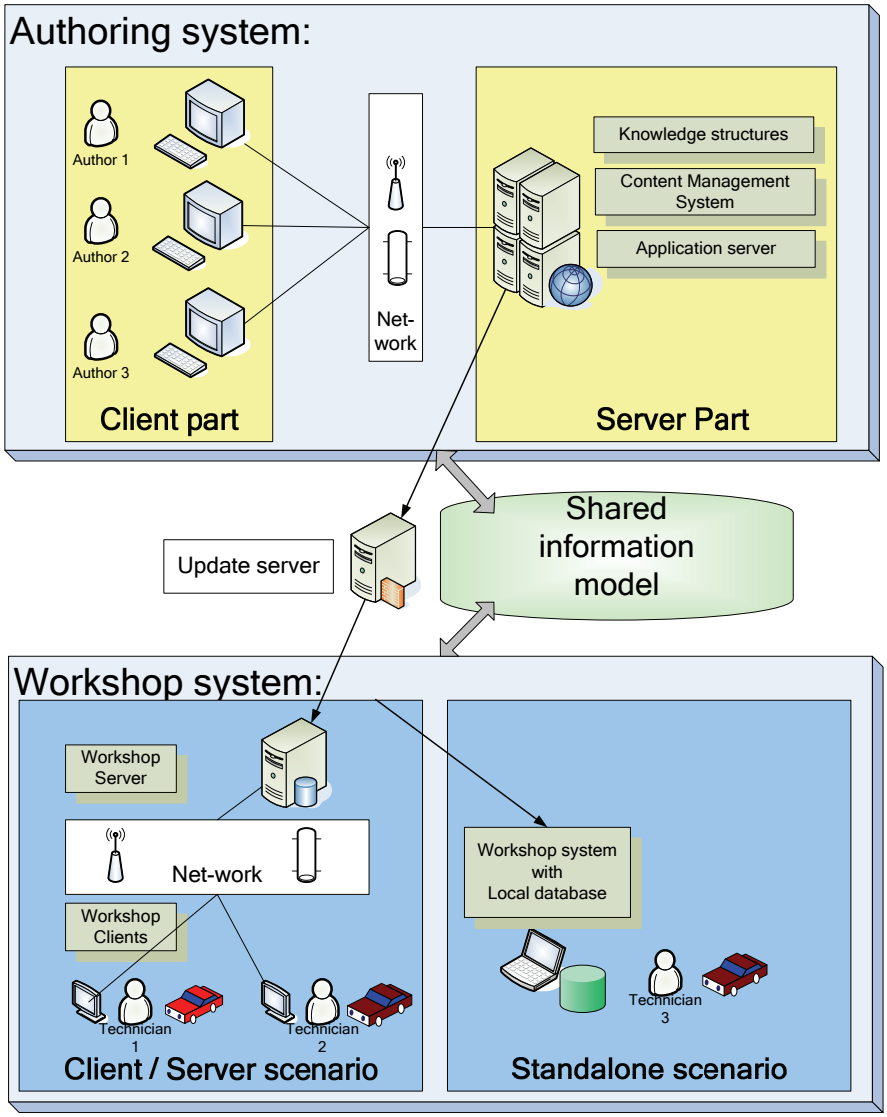

Figure C-3: Composant principaux de SIDIS Enterprise

Les caractéristiques principales sont listées ci-dessous :

- Le système est orienté-objet et l'information est organisé dans des structures arborescentes avec le principe d'héritage des propriétés des classes générales aux classes spécifiques.
- Un concept de référencement ou « *linkage* » permet de relier différent type d'objet (ex : des documents type notice de montage avec un composant).
- Un mécanisme de publication flexible permettant des mises à jours incrémentielle des mises à jour logicielles et des données.
- Le modèle d'information du système auteur et du client garage est identique.

2.4 Modélisation des connaissances de diagnostic dans SIDIS Enterprise

La Figure C-4 présente les structures relatives au diagnostic de SIDIS Enterprise. Outre l'arbre des variantes des véhicules et des tests, 3 structures sont fondamentales à la réalisation de la procédure de diagnostic guidé.

- L'arbre des objets de diagnostic : Il s'agit d'une décomposition hiérarchisée des composants du véhicule

- L'arbre des codes défauts : Il s'agit des différentes erreurs que peuvent remonter les calculateurs
- L'arbre des symptômes de perception : cet arbre est un ensemble de description qualitative de panne

Ces deux dernières structures sont liées par le biais de « lien de suspicion » vers les objets de diagnostic.

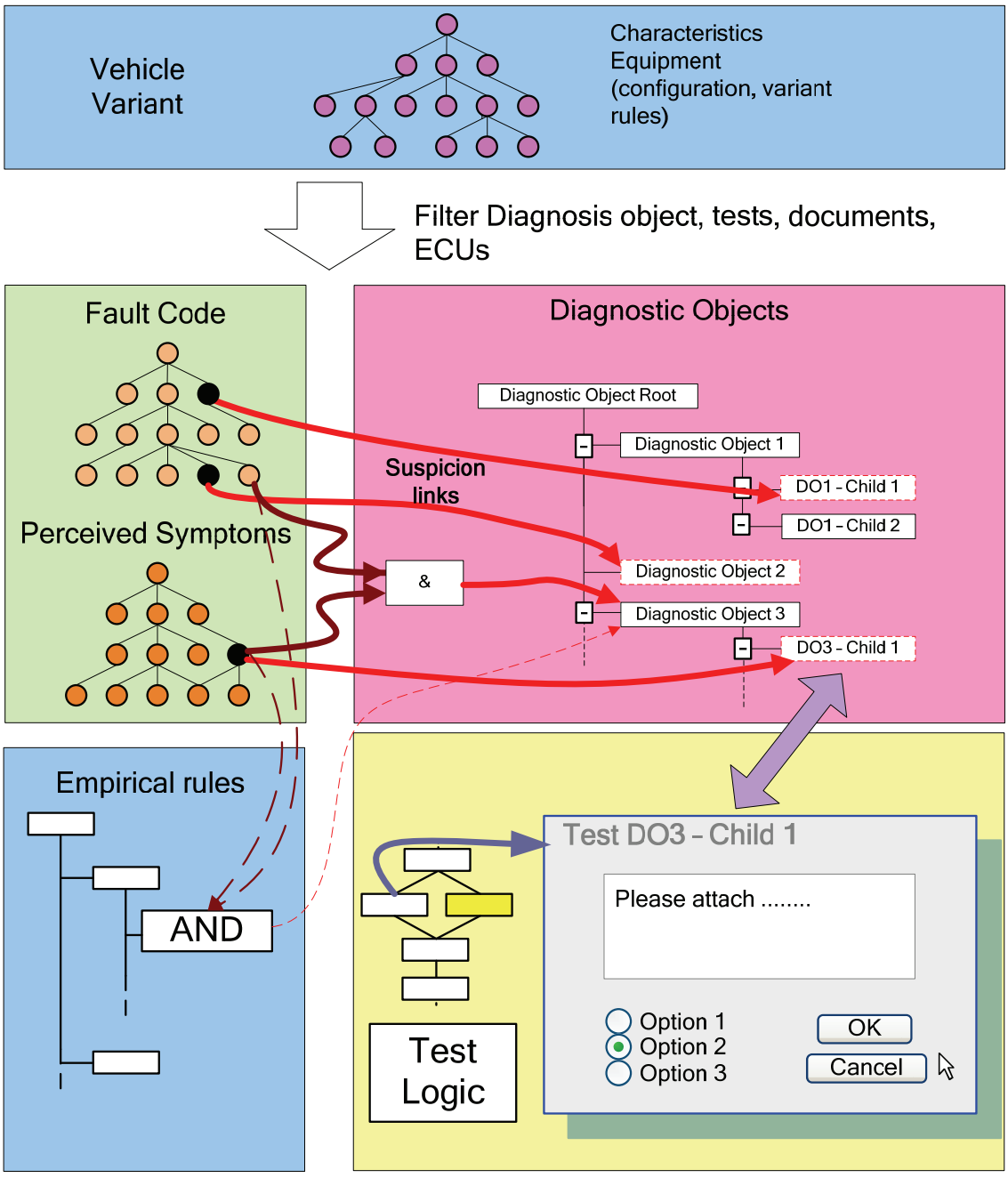

Figure C-4: Données relative au diagnostic dans SIDIS Enterprise

La procédure de diagnostic guidée dans le client garage se décompose dans les étapes représentées en Figure C-5.

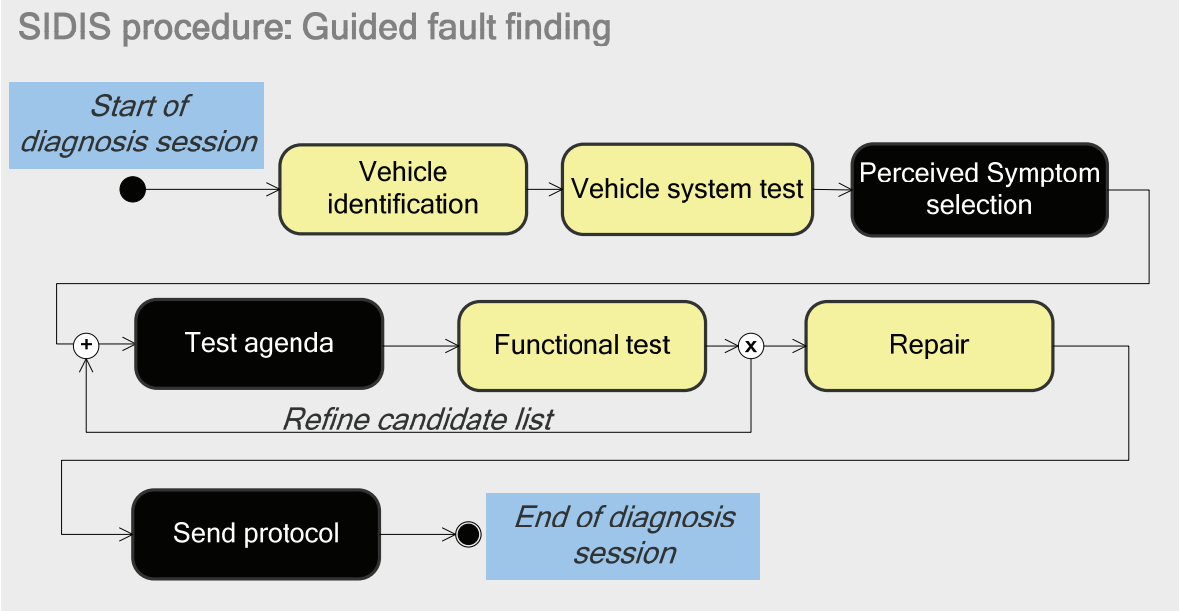

Figure C-5: Procédure de diagnostic guidée

Après avoir effectuée une identification véhicule et une acquisition des codes défauts, les symptômes de perceptions sont sélectionnées et les liens de suspicions initialisés. Ensuite en fonction de la pondération de ces liens les candidats sont classés dans un « Test Agenda » et le mécanicien peut choisir quel composant il souhaite examiner plus en profondeur. Cette boucle de test sert à réduire le champ des hypothèses afin de localiser le défaut et de le réparer. A la fin de la session de diagnostic un protocole est envoyé au serveur de retour d'expérience.

2.5 Conclusion

Compte-tenu que les objectifs de la thèse sont d'améliorer la procédure de diagnostic guidé en minimisant les coûts d'édition des données ainsi que les changements à implémenter dans la structure de SIDIS Enterprise 4 modules ont été développés. Le premier est dédié à la recherche des symptômes en langage naturel, le second au classement automatique des fichiers ODX. Le troisième se concentre sur l'algorithme de diagnostic et un module d'apprentissage automatique des données.

3 Traitement automatique des informations de diagnostic

3.1 Recherche des symptômes de perception

L'une des difficultés dans le processus de diagnostic guidée est de retrouver les symptômes de perceptions du type « le moteur vibre fortement lors du démarrage du moteur ». Ces derniers sont classés dans une structure arborescente mais le nombre de symptômes est souvent très important et le mécanicien doit rechercher les symptômes manuellement avec le système actuel. Par conséquent la plupart des mécaniciens ne rentrent pas de symptômes de perception et il en résulte une perte de temps due à l'exécution de tests intermédiaires lors de l'affinage du diagnostic. Pour aider le mécanicien dans cette tâche, j'ai développé un moteur de recherche des symptômes de perception en langage naturel fonctionnant en allemand, anglais et français. Cette section détaille les recherches menées sur ce module.

3.1.1 Méthode classique de recherche de l'information

Le langage naturel possède toute une série de caractéristiques qui représentent autant de difficultés à l'interprétation des phrases par ordinateur. On peut citer en exemple : la graphie, les variantes grammaticales, morphologiques, la synonymie, la polysémie, l'hyponymie, la terminologie, les expressions composées, la segmentation, et l'ordre des mots dans une phrase.

Les méthodes les plus utilisées en recherche documentaire commencent par examiner un corpus de documents et en extraire tous les mots utilisés pour constituer la base des termes (BoT). Chaque document est alors indexé par un vecteur colonne représentant les fréquences des termes utilisées. Par concaténation des vecteurs colonnes on obtient la matrice de fréquence des termes appelées *TF*. Afin de favoriser les termes qui contribuent à la différentiation informationnelle des documents, on applique une formule de pondération. Les plus courantes sont détaillées dans le Tableau C-1.

Nom de la formule de pondération	Formalisation	Commentaire
tf.idf (Term frequency, inverse document frequency) [Salton, 1989] :	$W(T_i, S_j) = tf_{S_j}(T_i).\log\left(\frac{N}{df_i}\right)$ **(Eq. C-1)**	
Harman weight [Harman, 1993]:	$W(T_i, S_j) = \frac{\log(tf_{S_j}(T_i)+1)}{\log(Length(S_j))}\log\left(\frac{N}{df_i}\right)$ **(Eq. C-2)**	
Croft weight [Croft, 1983]:	$W(T_i, S_j) = K + (1-K)\frac{tf_{S_j}(T_i)}{Max_i\{tf_{S_j}(T_i)\}}$ **(Eq. C-3)**	Avect $K \in]0,1[$ Souvent K = 0.5
Okapi weight [Robertson et al., 1994]:	$W(t_i, S_j) = \frac{tf_{S_j}(T_i)}{tf_{S_j}(T_i) + A}B$ **(Eq. C-4)**	$A = \frac{\| S_j \|}{Average_{k\in[1,N]}\{\| S_k \|\}}$ $B = \log\left(\frac{N + df_i + 0.5}{df_i + 0.5}\right)$ **(Eq. C-5)**

Tableau C-1: Formules de pondération

On applique la même modélisation à la requête à laquelle on applique l'une des formules de pondération. La dernière étape consiste alors à calculer la similarité entre requête et symptôme. Pour cela on peut utiliser la formule du cosinus qui correspond au produit scalaire divisé par le produit des normes euclidienne du vecteur de poids de la requête et du symptôme. Une étape de filtrage est ensuite nécessaire pour retourner les symptômes les plus pertinents. L'indicateur de performance le plus commode est la précision qui est le rapport entre symptôme pertinent et non pertinent retourné par l'algorithme de recherche pour une requête donnée, bien que la notion de pertinence ne soit pas stable.
Les méthodes de recherche sémantique s'appuient sur une structure hiérarchisée du vocabulaire. La recherche est ensuite approfondie dans les collections réduites de symptômes accrochées aux termes les plus spécifiques de l'arbre. Ce qui revient à coupler deux moteurs de recherche par index, avec des poids importants pour les mots utilisées dans la labellisation des collections de symptômes.

3.1.2 Méthode basée sur un graphe de correspondance

Au cours des recherches pour améliorer le module, j'ai développé une méthode basée sur un réseau de neurone [Azarian, Siadat, 2009]. Une vue d'ensemble est représentée en Figure C-6. Le système est composé de deux couches de nœuds, la première est une série de concept, la seconde les symptômes. Selon qu'un concept est important ou non dans un symptôme le lien entre les deux objets sera important ou nul. Pour calculer l'activité des nœuds dans la première couche une fonction distance sur la base de la distance de Levenstein combinée à une fonction Tangente hyperbolique a été définie. La propagation se fait par multiplication et sommation des niveaux d'activités entrant dans un nœud de la seconde couche. Puis une normalisation est effectuée pour ne pas favoriser les symptômes longs par rapport au plus court. Cette probabilité de pertinence obtenue en sortie de chaque symptôme permet de les filtrer selon la requête.

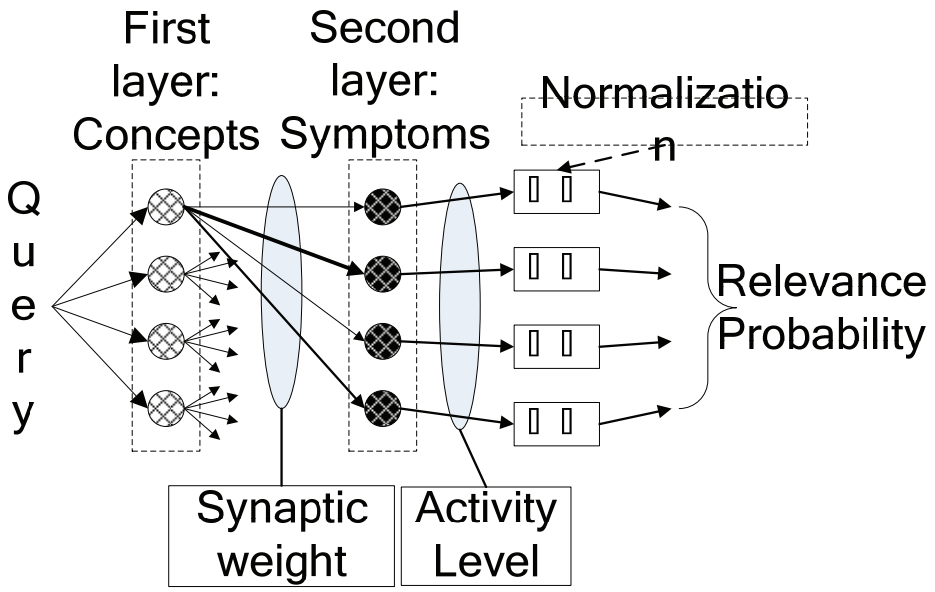

Figure C-6: Vue globale du système de recherché par graphe de correspondance

3.1.3 Résultat

Les résultats obtenus sur une base de tests sont données dans le Tableau C-2. On peut remarquer que le graphe par correspondance permet d'obtenir les meilleurs résultats. Cependant cela reste vrai pour la deuxième série de mesure qui implémente une fonction de filtrage heuristique en escalier dépendante du nombre de symptômes dont la similarité est supérieure à 0.3. Dans la première série un seuil arbitraire de filtrage à 30% a été fixé.

		Méthodes de recherche par index				Graphe de correspondance
		tf.idf	Harman	Okapi	Moyenne	
Serie 1	Pr	24%	25%	25%	25%	20%
Serie 2	Pr	44%	38%	43%	42%	60%

Tableau C-2: Résultat des méthodes de recherche de symptôme

L'avantage des méthodes par index réside dans leur automatisation du traitement et la facilité d'implémentation dans le système auteur. De plus elles sont multilingues alors que le graphe nécessite d'être reconçu pour chaque langue. Le steemming et thesaurus permette utilisé ici, permette d'autant plus d'améliorer la précision. Les bons résultats du graphe s'expliquent surtout sur la faculté de « perception » du réseau plutôt que le raisonnement booléen des méthodes par index.

3.2 Importation de fichier ODX

L'un des problèmes majeur face aux évolutions constantes dans le domaine de l'électronique des véhicules est d'actualiser les données des calculateurs. Ses fichiers de descriptions des fonctions de chaque calculateur se ressemblent d'un calculateur à la version supérieure, mais l'ensemble des fonctions doit être réimporté dans le fichier auteur à chaque fois. Afin d'économiser du temps pour cette opération, j'ai conçu un système multi-agents qui permet de classer automatiquement ces fichiers ODX.

3.2.1 Principe

Les objectifs principaux de ce module sont de réduire et simplifier le travail des auteurs autant que possible et d'aider les auteurs à classer et lier les nouvelles fonctionnalités des calculateurs. Pour cela un langage intermédiaire qui fonctionnalise l'information des fichiers ODX permettrait un

traitement efficace. La Figure C-7 représente les différents liens entre structure de connaissance de SIDIS Enterprise et fichier ODX.

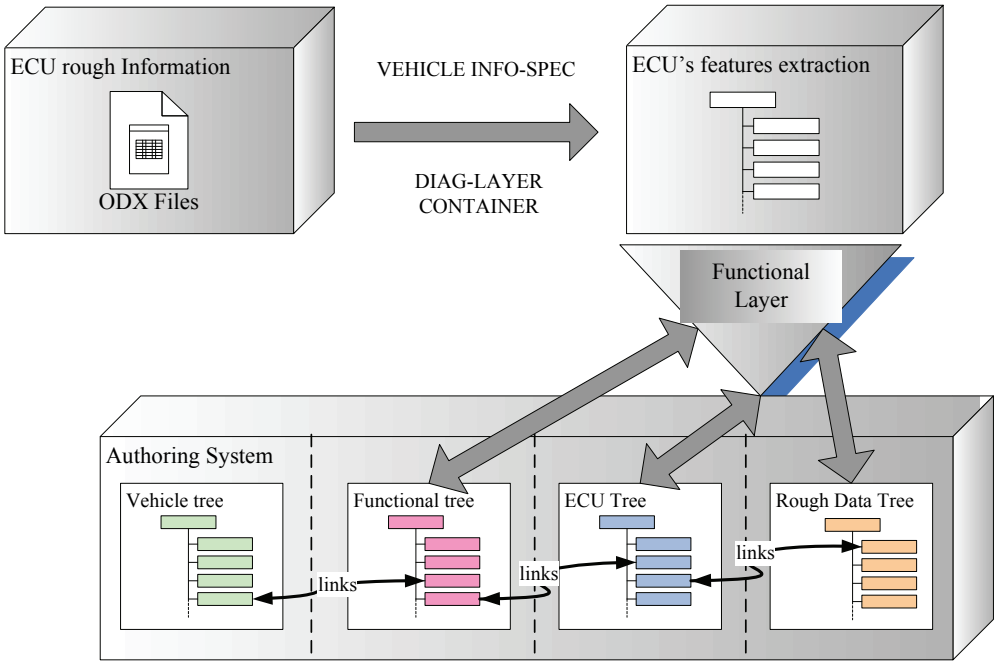

Figure C-7: Structure de connaissance de diagnostic relative au calculateur

3.2.2 Système multi-agent

Afin d'analyser le contenu des fichiers ODX, le système multi-agent présenté en Figure C-8 a été conçu.

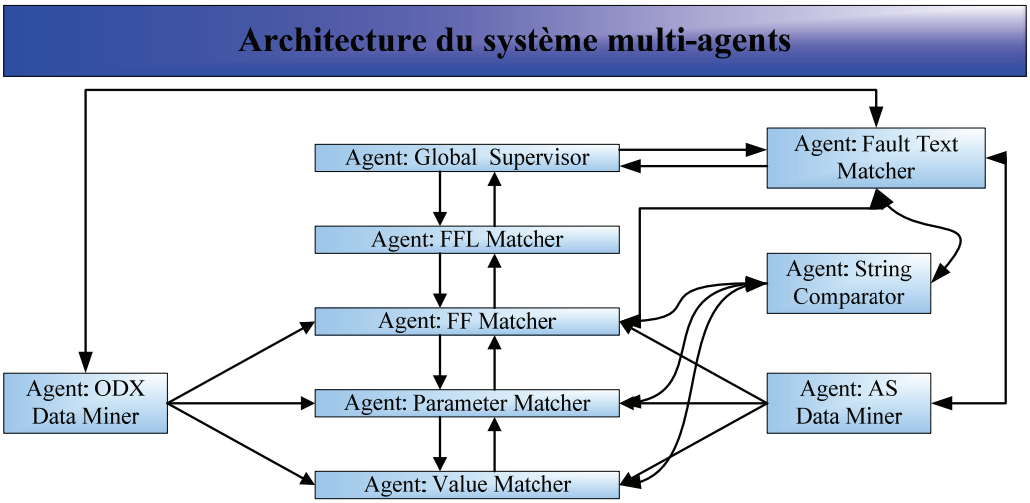

Figure C-8: Architecture du système multi-agent

Le rôle des différents agents est donné dans le Tableau C-3.

Agent	Rôle
ODX Data Miner	Chercher l'information dans le fichier ODX.
AS Data Miner	Trouver l'information dans le système auteur.
FFL Matcher	Déterminer si une FFL est liée à un ECU Déterminer la priorité de l'analyse des FF.

FF Matcher	Déterminer si les nouveaux DiagServices ODX concordent avec une FF. Déterminer la priorité de l'analyse des paramètres.
FF Parameter Matcher	Détermine quel paramètre ODX concorde le mieux avec le paramètre fonctionnel. Ordonner l'analyse des valeurs.
Value Matcher	Déterminer comment les valeurs d'un paramètre ODX et les paramètres fonctionnel peuvent être liés.
Fault Text Matcher	Déterminer si un texte d'un code défaut est lié à un ECU.
String Comparator	Donne une similarité entre deux chaînes de caractères.

Tableau C-3: Description des agents

L'agent FFL ordonne une recherche entre fonction, DiagService, paramètre et valeur entre le nouveaux fichier ODX et les fonctions déjà présente dans le système auteur. Chaque comparaison entre un paramètre et une valeur se fait par l'intermédiaire de l'agent String Comparator via des formules de type cosinus ou de recouvrement. Cette analyse descendante peut être interrompue si les résultats de similarité intermédiaire sont insuffisants.

4 Moteur de diagnostic

L'une des problématiques centrales de cette thèse a été de s'intéresser à l'amélioration du moteur de diagnostic de SIDIS Enterprise. Ce dernier est uniquement basé sur les liens de suspicions. En fonction du rang d'une relation (*RSL*) du poids de l'objet à la source du lien (*GS*) et du nombre de relation *k* pointant sur l'objet de diagnostic considéré (cf. Eq. C-4). Le moment partial de suspicion *PM* du lien est alors additionné avec celui des autres liens. Les objets sont ensuite classés par ordre décroissant du moment de suspicion divisé par le coût du test associé.

$$PM_n = GS\frac{(k+1)-RSL_n}{\sum_{i=1}^{k} RSL_i}$$

(Eq. C-6)

Si un test est effectué selon le résultat obtenu, le moment de suspicion est nul (réponse OK du test) ou les objets fils sont suspectés. L'algorithme parcoure ainsi récursivement le graphe en profondeur jusqu'à localiser la panne.

L'une des idées intuitives pour améliorer le diagnostic a été de modifier la métrique d'ordonnancement des tests. Le Tableau C-4 précise les différentes métriques utilisées ainsi que les résultats obtenus et une comparaison avec l'algorithme usuel de SIDIS Enterprise.

Nom de la métrique	**Expression littérale**	**Résultat**	**Comparaison avec l'orbite de SIDIS Enterprise**
Lissage entre suspicion et coût des tests	$\alpha SM + (1-\alpha)\frac{Tc_{\max}}{Tc}$ **(Eq. C-7)**	6.6 for $\alpha = 0.6$	+7%
Lissage entre suspicion et facteur coût des tests	$\alpha SM + (1-\alpha)\frac{Tc_{\max}}{Tc}100\%$ **(Eq. C-8)**	6.6 for $\alpha = 0.9999$	+7%

normalisés			
Contraintes légère entre coût des tests et suspicion	$\alpha^{\beta} SM + (1-\alpha)^{\beta} \frac{Tc_{\max}}{Tc}$ **(Eq. C-9)**	11.9 for $\alpha = 0.5$ and $\beta = 0.1$	+95%
Contraintes légère entre suspicion et coût des tests avec inspection en profondeur	$\alpha SM + (1-\alpha)\frac{Tc_{\max}}{2}\left(\frac{1}{Tc} + \frac{1}{\overline{Tc_{child}}}\right)$ **(Eq. C-10)**	5.8 for $\alpha = 0.5$	-6%

Tableau C-4: Différentes métriques d'ordonnancement des tests

Deux autres idées intuitives ont permis d'améliorer les performances de près de 15%. La première consiste à pondérer les codes défauts ou symptômes dynamiquement en fonction du pouvoir de discrimination des liens de suspicion qui leurs sont attachés [Gorny, Ligeza, 2001]. L'autre idée consiste à augmenter l'information disponible lors de l'initialisation de la session de diagnostic, ainsi si un lien entre symptôme de perception et objet de diagnostic permet de discriminer entre plusieurs candidats, la question de l'observation du symptôme est posée à l'utilisateur et le plan de test est recalculé.

4.1 Algorithme de diagnostic

4.1.1 Approche par méta-heuristique

L'une des approches développée au cours de la thèse a été de définir une méta-heuristique pour calculer le moment de suspicion des objets de diagnostics. Ces méthodes présentent l'avantage d'être facilement interprétable, implémentable, peuvent être combiné avec d'autres techniques de décision, ne demande pas de ressources de calcul. L'Eq. C-11 détaille le calcul du moment de suspicion.

$$SM_{node} = a_0 \log_2\left(o + \frac{Level(node)}{Depth(tree)}\right) + a_1 \log_2\left(o + \frac{Tc(node)}{Tc_{\max}}\right) +$$
$$+ a_2 \log_2\left(o + \frac{LT(vehicle)}{LT(node)}\right) + a_3 \log_2\left(o + \frac{\sum_i e_i GS_i}{\sum_k GS_k}\right) + a_4 \log_2\left(o + \frac{\sum_i e_i GC_i}{\sum_k GC_k}\right)$$

(Eq. C-11)

Le principal atout de cette méthode est qu'elle discrimine les candidats par une fonction logarithmique au lieu du comportement linéaire de SIDIS Enterprise (cf. Eq. C-4). Après la réalisation d'une étude de cas il s'avère que le les points de fonctions de cette méta-heuristique se situe dans l'intervalle (10%, [20%, 50%], 20%, 100%, 50%) pour les coefficients (a_0, a_1, a_2, a_3, a_4).

4.1.2 Approche hybride basée sur modèle et par règle experte

La seconde approche développée dans cette thèse repose sur l'entropie informationnelle de Réyni. La formule pour le moment de suspicion causal est donnée dans l'Eq. C-12 auquel s'ajoute un moment de suspicion induit par la durée de vie des composants modélisés ici par une fonction échelon (cf. Eq. C-13).

$$SM_{\alpha} = \left| \frac{1}{1-\alpha} \log_2\left(o + \frac{Level(node)}{Depth(tree)}\right)^{\alpha} + \frac{1}{1-\alpha} \log_2\left(o + \frac{\sum_i e_i GC_i}{\sum_k GC_k}\right)^{\alpha} \right|$$

(Eq. C-12)

$$PM_{LT} = Heaviside(LT(vehicle) - LT(node))$$

(Eq. C-13)

L'ensemble est linéarisé avec le moment de suspicion obtenu par SIDIS Enterprise par la formule donnée dans l'Eq. C-14.

$$SM_f = \beta^{\gamma} SM_h + (1-\beta)^{\gamma} SM_{\alpha} + c_{LT} PM_{LT}$$

(Eq. C-14)

Il faut noter que le comportement de cette approche favorise la « certitude » et que par conséquent si un objet de diagnostic est suspecté par SIDIS Enterprise et que ses prédécesseurs causaux sont également suspecté alors sont moment de suspicion sera d'autant plus élevés. Si le nombre de relation causale entrante active est égal au nombre de relation causale entrante alors la suspicion sera maximale d'où la « quasi-certitude » évoquée auparavant. Cette approche permet de d'augmenter les performances de diagnostic d'environ 9.6% tout en changeant que très peu le modèle d'information de SIDIS Enterprise. Cependant cette approche a l'avantage de fonctionner également sur la base du simple système expert de SIDIS Enterprise ce qui en fait une innovation incrémentale. Ce point est essentiel car les processus d'édition des données chez les constructeurs reposent essentiellement sur des systèmes experts et cette approche favorise une transition vers le diagnostic basé sur un modèle.

4.2 Apprentissage automatique

Les deux méthodes présentées ci-dessus ont le désavantage d'utiliser de nouvelles sources d'informations qui engendrent des coûts supplémentaires pour les compléter. Par conséquent le dernier module développé dans cette thèse concerne l'apprentissage automatique des données sur la base des protocoles des sessions de diagnostics établis dans le réseau après-vente. Les paramètres concernés sont listés ci-dessous :

- Optimisation du moteur de recherche des symptômes.
- Calibrage du coût des tests.
- Apprentissage de la durée de vie des composants.
- Optimisation des coefficients de poids des règles de suspicion et des relations causales.
- Apprentissage des dépendances entre composant.
- Apprentissage de la couverture et de l'indice de confiance des tests.

Pour réaliser l'apprentissage automatique 2 méthodes ont été exploitées : les réseaux Bayésien et les arbres de décisions. Le coût des tests et la durée de vie des composants sont exploité par les arbres de décisions qui génèrent des hypothèses sur la durée et fonction de l'exactitude de l'hypothèse sur le nombre de cas considéré et du facteur d'impureté de Giny l'hypothèse est reformulée et re-testée ou acceptée. Pour les autres paramètres, le tel que le poids des symptômes, une itération est réalisée sur l'ensemble des cas retourné au serveur de retour d'expérience et en fonction du ratio nombre de fois ou le symptôme est acute et nombre total de retour d'expérience un nouveau coefficient de poids est attribué. Ce réseau bayésien est basée sur une hypothèse naïve où tous les liens de suspicion et causaux existent mais implémentés dans l'architecture présentée Figure C-9, elle permet de limiter le traitement des données et surtout la préparation des données qui a lieu dans le client garage avant d'écrire et d'envoyer le protocole.

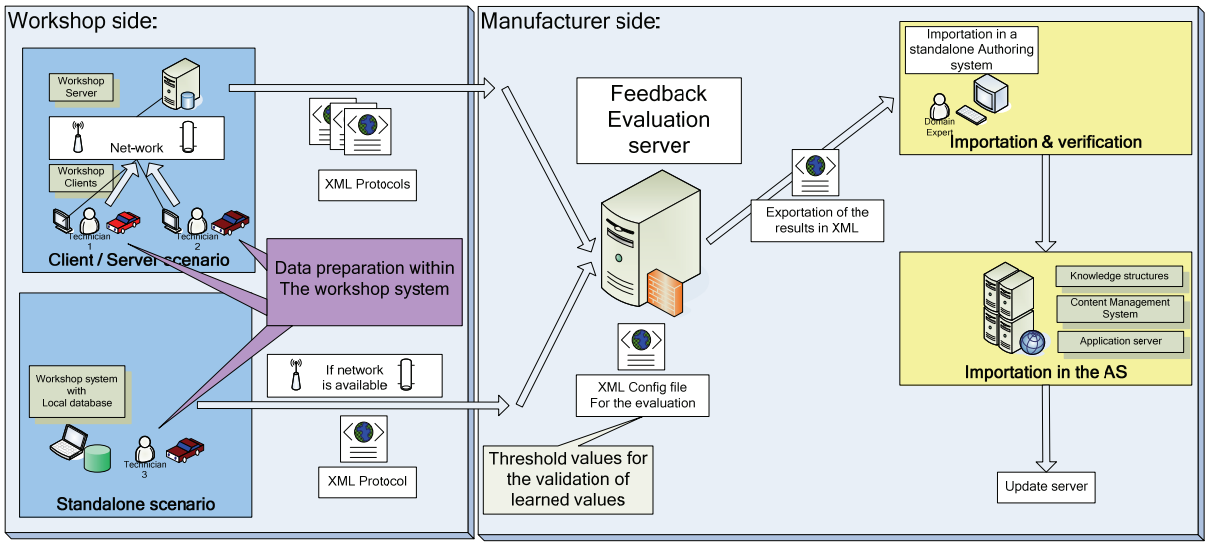

Figure C-9: Architecture pour le module de retour d'expérience

Le retour d'expérience pour l'algorithme de recherche des symptômes va étudier les requêtes, les symptômes retournés et surtout les symptômes sélectionné par le mécanicien. Les mots utilisés dans la requête permettent d'enrichir une matrice de popularité qui favorisera le retour de ces symptômes sélectionnés pour la même requête.

5 Vérification et validation

5.1 Prototype développé

Le prototype développé pour simuler des cas de panne automobile permet d'éditer les mêmes structures de connaissance que dans SIDIS Enterprise. De plus un mode démonstrateur a été implémenté qui permet de simuler pas à pas une procédure de diagnostic guidée.

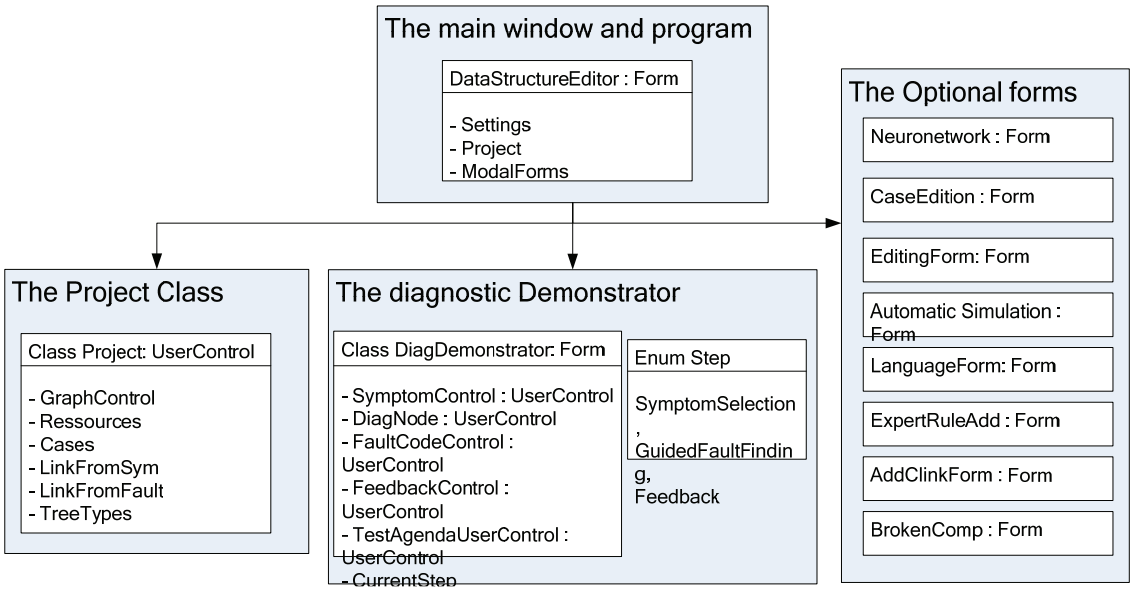

Figure C-10: Architecture du prototype

La classe « *Project* » (cf. Figure C-10) contient l'ensemble des champs et méthodes pour le moteur de recherche des symptômes, de diagnostic et de retour d'expérience. C'est la classe principale de l'application caractérisée par 40 champs et environ 120 méthodes (11.000 lignes de codes). De nombreuses méthodes d'optimisation en mémoire et de stockage ont du être développées pour la

stabilité du simulateur. La Figure C-11 représente l'interface principale du simulateur ainsi qu'une description des zones d'éditions des données.

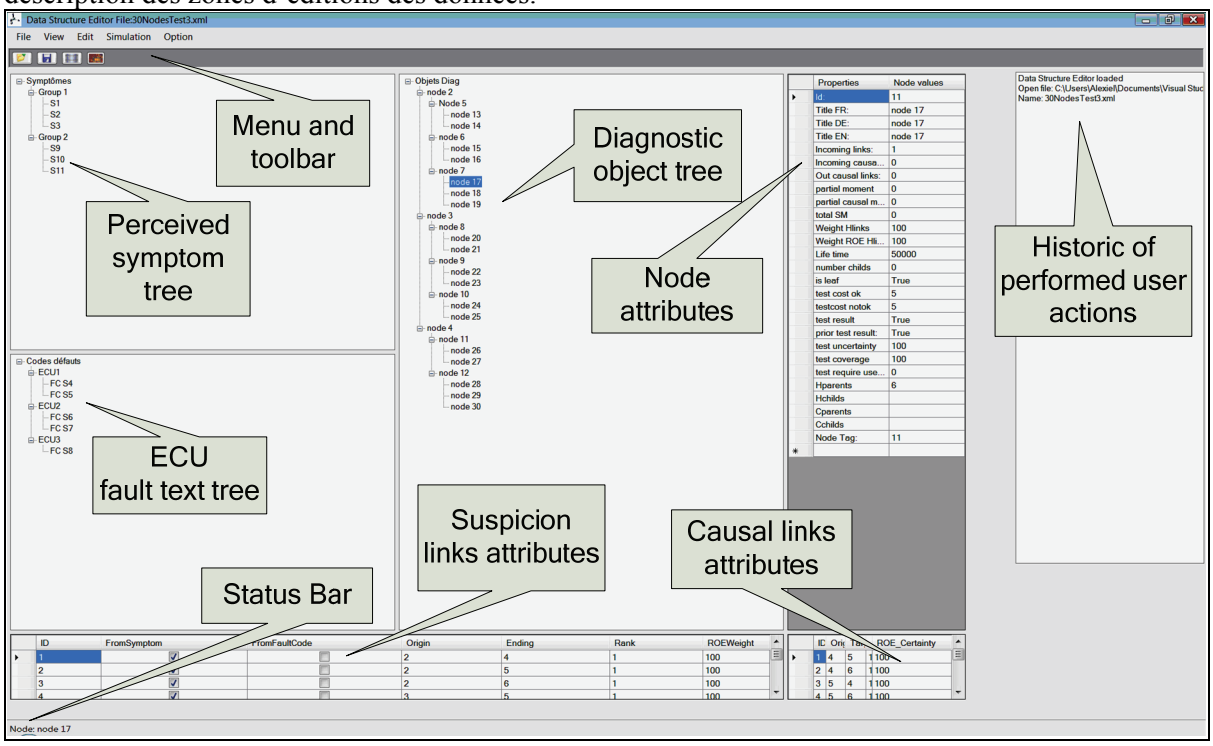

Figure C-11: Interface du simulateur

5.2 Vérification sur un véhicule réels

Le véhicule A appartient au haut de gamme. Il comprend environ 500 objets de diagnostic, 200 symptôme de perception, 500 codes défauts et 300 règle expertes. Les pannes simulées sur ce véhicule concernent le moteur et l'ABS. La recherche des symptômes de perceptions souligne les performances de la formule de pondération *tf.idf* avec une précision moyenne de 69% (avec des recherches en français, allemand, anglais). L'activation des symptômes retournés par le module de recherche permet d'économiser 2 tests en moyenne sur les 3 pannes simulées sur ce véhicule. Les pannes simulées appartiennent au domaine des pannes critiques qui génèrent plus de 60 codes défaus avec un haut degré de dispertion.

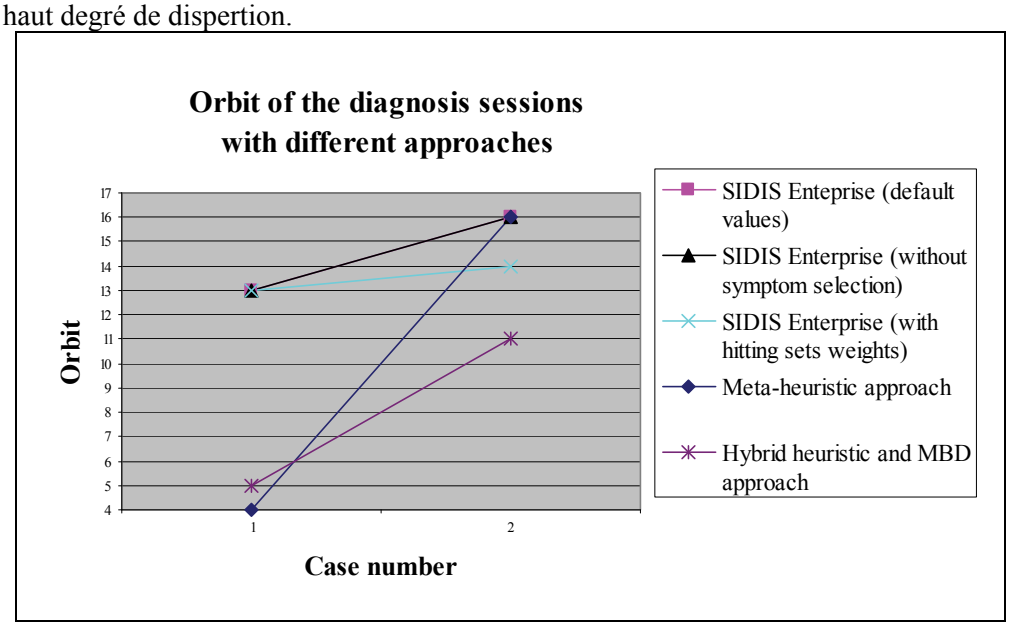

Figure C-12 : Orbite de la procedure de diagnostic guide

La Figure C-12 schématise les orbites des sessions de diagnostic avec les différentes approches sachant que les facteurs de poids dans les modèles ont tous étés initialisés à leur valeur par défaut (ce qui est le cas le plus courant dans l'industrie). La Figure C-13 contient le graphe des orbites sur les deux pannes simulées après évaluation du retour d'expérience. Dans le cas présent outre les deux cas simulées d'autres protocoles ont étés fournis au moteur d'évaluation du retour d'expérience avec des pannes très diverses. Ce fait explique pourquoi dans le premier cas, l'orbite des sessions de diagnostic a augmenté de deux tests para rapport à l'expérience de la Figure C-12.

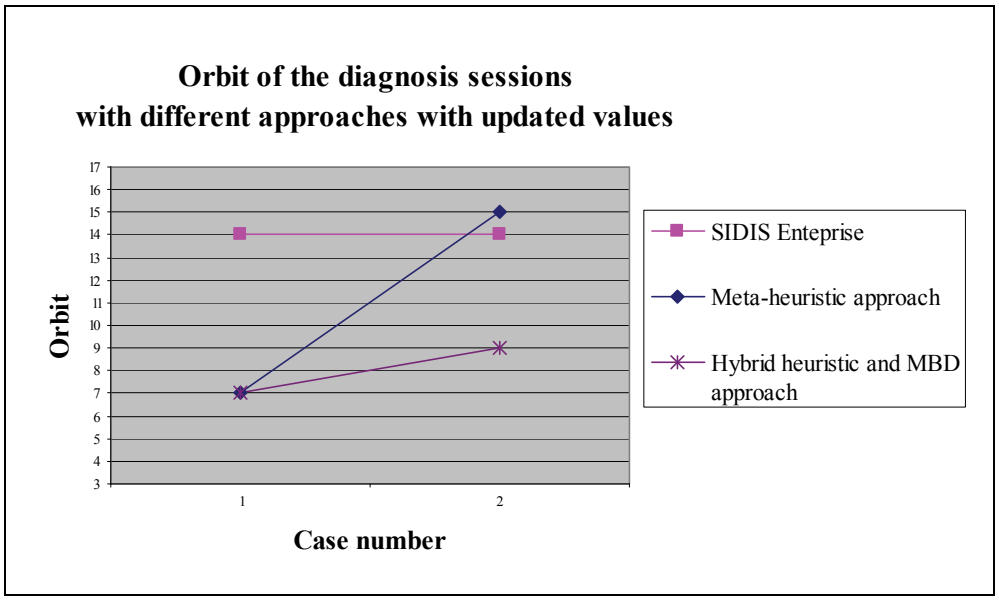

Figure C-13 : Orbite des sessions de diagnostic après evaluation du retour d'expérience

5.3 Discussion

La recherche en langage naturel des symptômes a permis une amélioration moyenne de 0.28 test/session (soit environ 1.4 minute/session). Ce module permet des économies importantes lorsque ce gain est ramené à l'échelle d'un constructeur : environ 28 heures d'économie par jour pour un réseau avec 10.000 garages. Par ailleurs en termes de précision la formule de pondération prédomine sur les autres méthodes avec un score d'environ 82% en français anglais et 51% en allemand (couplé à la métrique heuristique pour fixer le seuil d'acceptabilité). Cet effet est essentiellement du à la qualité du *stemming* en allemand, qui est trop sévère.

L'approche hybride pour le diagnostic guidé diminue l'orbite d'une valeur moyenne de 2 tests si les valeurs par défauts sont conservées. L'approche par méta-heuristique permet d'économiser en moyenne 1.72 tests par session. Par contre après injection des résultats obtenus par évaluation du retour d'expérience l'économie moyenne est de 1.75 tests avec l'approche méta-heuristique et de 3 test par l'approche hybride. Une étude des variations sur les paramètres β, γ, (cf. Eq. C-14) n'a pas permis d'améliorer les performances.

6 Conclusion

Dans cette thèse, le développement des 4 modules a permis de réaliser une plateforme modulaire dédié au diagnostic automobile. L'une des avancées significatives de cette plateforme réside dans le fait qu'elle combine efficacement le diagnostic basé sur un système expert et le diagnostic basé sur des modèles. La Figure C-14 présente la combinaison des différentes sources de connaissances utilisées pour le nouveau algorithme de diagnostic.

En moyenne une économie de 3.28 tests / sessions peu être réalisé grâce au module développée avec une réduction des coûts d'édition des données par le retour d'expérience et le classement automatique des fichiers ODX.

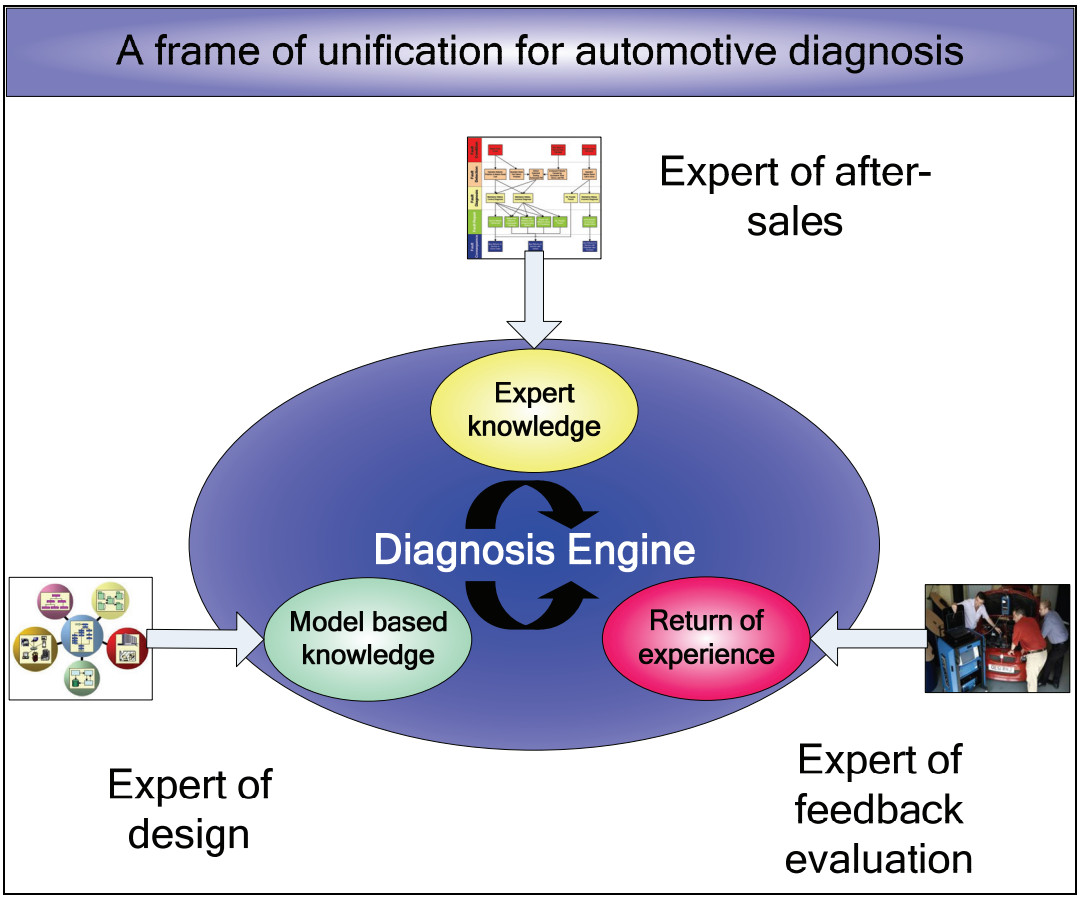

Figure C-14: Plateforme modulaire pour le diagnostic automobile

Cependant des évolutions techniques peuvent encore s'ajouter à la complexité actuelle du diagnostic. D'une part les véhicules électriques ou hybrides risquent de contenir beaucoup plus de système électronique afin de minimiser les effets sur l'environnement. De même l'évolution démographique de la société et une espérance de vie en constante augmentation créera des nouveaux besoins d'aide à la conduite tel que la détection automatique de présence dans les angles morts, la reconnaissance automatiques des panneaux de circulations et les technologies de communication entre véhicule...

Cette thèse a abordé en particulier la notion de causalité, centrale en intelligence artificielle car si elle est formalisée elle permettrait d'améliorer les algorithmes d'apprentissage automatique. Cependant cette thèse est concentrée sur les notions pratiques en vue d'application exclusive du diagnostic. Les études de cas menées sur le simulateur ont prouvé que la méthode peut s'appliquer dans un contexte industriel.

Abstract

This dissertation explores the field of diagnosis applied to the automotive industry based on SIDIS Enterprise developed by Siemens AG. At the beginning we analyzed the importance of the after-sale world for car manufacturers and in the automotive value chain, as well as the requirements of manufacturers for computer assisted diagnostic tools. One important requirement for manufacturers was to limit the post development cost of models for the diagnosis. Furthermore, the analysis shows that electronic failures increase as well as electronic equipment with the consequences that the failure localization becomes more and more complex. The need for computer assisted tools for car manufacturers was therefore essential in order to save time for the fault localization, for the client's satisfaction, and for the image of the brand.

This thesis leads to a global modular framework for automotive diagnosis composed by: a perceived symptom search engine, an automatic ODX exchange system, a hybrid heuristic and model based diagnostic strategy, a feedback engine. The first module is concerned with the interpretation and mapping of qualitative failure descriptions in natural language. The second module allows authors of the database and knowledge network to automatically import Electronic Control Units description files. The last ones are concerned with the development of a diagnosis algorithm, which combines all different knowledge sources and benefits from the return of experience for the auto completion of the models. All these modules contribute to the overall reduction of the costs of models for the diagnosis and a significant decrease of the orbit of guided fault finding procedure.

Résumé

Cette thèse explore le domaine du diagnostic appliqué à l'automobile basé sur le produit SIDIS Enterprise développé par Siemens AG. Au début, une analyse est conduite de l'importance du réseau après-vente pour les constructeurs et leurs places dans la chaine de valeur de l'automobile ainsi que les contraintes industrielles des outils d'aide au diagnostic. L'une d'elle est de limiter le coût de développement des modèles pour le diagnostic. De plus, une analyse approfondie démontre que la part des composants électroniques et des pannes d'origine électronique des véhicules à tendance à augmenter tout en complexifiant la localisation de leurs origines. Le besoin d'un outil moderne pour les constructeurs est essentiel pour la satisfaction client et pour l'image de marque.
Ces travaux ont conduit au développement d'une plateforme modulaire pour le diagnostic composée : d'un module de recherche des symptômes, un système automatisé d'échange de fichier ODX, une méthode de diagnostic hybride et d'un moteur d'évaluation du retour d'expérience. Le premier module est dédié à l'interprétation et au mappage de la description des symptômes en langage naturel. Le second module permet aux auteurs des bases de données et des connaissances d'importer automatiquement les fichiers de descriptions des calculateurs électroniques. Les deux derniers sont dédiés à un algorithme de diagnostic qui combine les différentes sources de connaissances et bénéficie du retour d'expérience pour compléter automatiquement les modèles. Ces modules contribuent à l'objectif de réduction des coûts de développements des modèles pour le diagnostic et diminue significativement l'orbite des sessions de diagnostic guidé.